Kompression, Pose-Tracking und Halftoning

Dissertation zur Erlangung des Grades des Doktors der Ingenieurwissenschaften der
Naturwissenschaftlich-Technischen Fakultäten der Universität des Saarlandes

vorgelegt von

Christian Schmaltz

Saarbrücken 2012

Teilweise unterstützt durch die DFG (We 2602/5-1, We 2602/5-2, We 2602/9-1) und die
Graduiertenschule der Universität des Saarlandes.

Mathematische Bildverarbeitungsgruppe, Fakultät für Mathematik und Informatik,
Universität des Saarlandes, 66041 Saarbrücken

Tag des Kolloquiums

23.03.2012

Dekan

Prof. Dr. Holger Hermanns

Prüfungsausschuss

Prof. Dr. Hans-Peter Lenhof (Vorsitz)
Universität des Saarlandes

Prof. Dr. Joachim Weickert (Gutachter und Betreuer)
Universität des Saarlandes

Prof. Dr. Luis Álvarez León (Gutachter)
Universidad de Las Palmas de Gran Canaria

Prof. Dr. Bodo Rosenhahn (Gutachter)
Leibniz Universität Hannover

Dr. Simon Setzer
Universität des Saarlandes

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Saarbrücken, 23.03.2012

Kurzzusammenfassung – Short Abstract

Kurzzusammenfassung – Deutsch

In dieser Arbeit werden die auf den ersten Blick vollkommen voneinander unabhängig erscheinenden Bereiche Bildkompression, 3D-Posenschätzung und Halbtonverfahren behandelt und im Bereich der Videokompression sinnvoll zusammengeführt. Unser erster Beitrag ist ein Bildkompressionsalgorithmus, der auf einem rechteckigen Unterteilungsschema basiert. Dieser Algorithmus speichert nur eine kleine Teilmenge der im Bild vorhandenen Punkte, während die restlichen Punkte mittels partieller Differentialgleichungen rekonstruiert werden. Danach stellen wir ein Posenschätzverfahren vor, welches die 3D-Position und Ausrichtung von mehreren Objekten anhand von Bilddaten gleichzeitig verfolgen kann. Unser Verfahren funktioniert bei verrauschten Videos und im Falle von Objektüberlagerungen. Auch Verdeckungen innerhalb einer kinematischen Kette werden natürlich behandelt. Unser dritter Beitrag ist ein Halbtonverfahren, das auf elektrostatischen Prinzipien beruht. Durch eine Reihe von Erweiterungen kann dieses Verfahren flexibel an verschiedene Szenarien angepasst werden. So ist es beispielsweise möglich, verschiedene Punktgrößen zu verwenden oder Schraffuren zu erzeugen. Der letzte Teil der Arbeit zeigt, wie man unseren Bildkompressionsalgorithmus, unser Posenschätzverfahren und unser Halbtonverfahren zu neuen Videokompressionsalgorithmen kombinieren kann. Die für jeden der vier Themenbereiche entwickelten Verfahren erzielen hervorragende Resultate, welche die Ergebnisse anderer moderner Verfahren übertreffen.

Short Abstract – English

In this thesis, we discuss image compression, pose tracking, and halftoning. Although these areas seem to be unrelated at first glance, they can be connected through video coding as application scenario. Our first contribution is an image compression algorithm based on a rectangular subdivision scheme which stores only a small subsets of the image points. From these points, the remained of the image is reconstructed using partial differential equations. Afterwards, we present a pose tracking algorithm that is able to follow the 3-D position and orientation of multiple objects simultaneously. The algorithm can deal with noisy sequences, and naturally handles both occlusions between different objects, as well as occlusions occurring in kinematic chains. Our third contribution is a halftoning algorithm based on electrostatic principles, which can easily be adjusted to different settings through a number of extensions. Examples include modifications to handle varying dot sizes or hatching. In the final part of the thesis, we show how to combine our image compression, pose tracking, and halftoning algorithms to novel video compression codecs. In each of these four topics, our algorithms yield excellent results that outperform those of other state-of-the-art algorithms.

Zusammenfassung

In der vorliegende Arbeit werden Algorithmen aus vier anspruchsvollen Forschungsgebieten betrachtet: Zunächst führen wir ein Bildkompressionsverfahren ein, das nur einige wenige Punkte des Bildes speichert und die restlichen Bildpunkte mittels einer partiellen Differentialgleichung (PDE) rekonstruiert. Zur Reduktion des zur Speicherung der Punktpositionen nötigen Speicherbedarfs werden die Punkte mittels einer adaptiven Unterteilung des Bildes in Rechtecke bestimmt. Unter anderem werden verschiedene PDEs, Methoden die gespeicherten Punkte zu wählen und Entropiekodierer evaluiert. Durch eine Reihe weiterer behutsamer Optimierungen ist es somit möglich, den Speicherbedarf eines Bildes deutlich zu reduzieren. Experimente mit Standard-Testbildern zeigen, dass dieser Ansatz bessere Kompressionsraten als JPEG erreicht und sogar dessen Nachfolger JPEG 2000 übertrifft.

Unser zweiter Beitrag ist ein Algorithmus zur Verfolgung der sogenannten *Pose* sich bewegender Objekte. Bei der Pose handelt es sich dabei um Lage, Ausrichtung und eventuell vorhandene interne Parameter, wie beispielsweise Gelenkwinkel. Die Pose wird durch die Minimierung eines Energiefunktionals gefunden, das direkt auf den Poseparametern arbeitet. Unser Posenschätzverfahren kann mehrere unterschiedliche Objekte gleichzeitig verfolgen und dabei sowohl Selbstverdeckungen als auch Verdeckungen durch andere Objekte bewältigen. Obwohl verschiedene Erweiterung das Einbringen von Vorwissen ermöglichen, liefert unser Verfahren auch ohne solche a priori Informationen gute Ergebnisse. Beispiele für mögliches Vorwissen sind bekannte Hintergrundbilder oder Einschränkungen der möglichen Posen.

Wir haben unser Posenschätzverfahren mittels einer Vielzahl von Sequenzen evaluiert, die insgesamt aus über 15000 Bilder und mehr als einem Dutzend verschiedener Objekte bestehen. Ein darunter befindlicher standardisierter Vergleichstest zeigt, dass die Ergebnisse unseres Verfahrens zu den Besten der Literatur gehören.

Im dritten Teil dieser Arbeit beschäftigen wir uns mit sogenannten *Halbtonverfahren*, welche Bilder mit kontinuierlichen Grautönen durch schwarze Punkte auf einem weißen Hintergrund annähern. Solche Verfahren werden insbesondere bei Druckern und Fax-Maschinen verwendet, die nur rein schwarze Punkte auf weißes Papier erstellen können.

Das von uns entwickelte Halbtonverfahren basiert auf der Idee, schwarze Punkte durch elektrisch geladene Partikel zu modellieren, die sich in einem vom Bild bestimmten Kraftfeld bewegen. Durch eine geschickte Kombination der abstoßenden Kräfte zwischen verschiedenen Partikeln und der durch das Bild gegebenen anziehenden Kräfte verteilen sich die Partikel derart, dass eine gute Näherung des Eingabebildes erreicht wird. Außerdem stellen wir eine Reihe von Erweiterungen dieses einfachen und flexiblen Ansatzes vor, um optimale Ergebnisse für verschiedene Anwendungen zu erhalten. Diese erlauben unter anderem die Behandlung von Punkten mit unterschiedlichen Größen oder auch von Linien.

Schließlich demonstrieren wir, wie unsere Verfahren der Themenbereiche Bildkompression, 3D-Posenschätzung, sowie unser Halbtonverfahren zu verschiedenen Videokompressionsverfahren vereint werden können. Moderne Videokompressionsverfahren der MPEG- oder H.26x-Familie verwenden zur Kompression der Videodaten Bewegungskompensation und Bildkompressionsverfahren, die auf Bildtransformationen beruhen. Da unser Verfahren auf

Anwendungen mit nahezu statischem Hintergrund spezialisiert ist, verwenden wir jedoch eine andere Herangehensweise, in der Vorder- und Hintergrund eines Videos unabhängig voneinander gespeichert werden. Dieses Verfahren eignet sich für praktische Anwendungen in vielen Bereichen, beispielsweise bei Videokonferenzen oder der Verkehrs- und Sicherheitsüberwachung.

Unser Videokompressionsverfahren verfolgt zunächst die sich bewegenden Vordergrundobjekte. Aus den so gewonnenen Posen und dem komprimierten Hintergrundbild wird zusammen mit weiteren Informationen wie den 3D-Modellen der Objekte und der Projektionsmatrix der Kamera das ursprüngliche Video näherungsweise wiederhergestellt.

Die Qualität des resultierenden Videos ist allerdings oft unbefriedigend, da es aus verschiedene Gründen zu deutlich sichtbaren Fehlern kommen kann. Solche Fehler entstehen beispielsweise durch ungenaue Modelle, falsche Posenchätzungen oder Beleuchtungsänderungen. Um solche Fehler zu kompensieren, wird zusätzlich der Unterschied zwischen dem ursprünglichem Video und seiner Rekonstruktion durch Informationen in ausgewählten Bildpunkten gespeichert. Ähnlich zur Speicherung des Hintergrundes wird hierzu das Differenzbild mittels partieller Differentialgleichungen aus diesen einzelnen Bildpunkten erzeugt. Unser Halbtonverfahren liefert hierbei passende Punktpositionen. Obwohl die Entwicklung der vorgestellten Videokompressionsverfahren gerade erst begonnen hat, liefern erste Experimente bereits jetzt Ergebnisse, die mit dem aktuellen Stand der Technik vergleichbar sind.

Somit führen wir in dieser Arbeit Bildkompressions-, 3D-Posenschätzungs- und Halbtonverfahren ein, und zeigen, dass sich diese zu Videokompressionsalgorithmen zusammenführen lassen. Die von uns für die vier betrachteten Themengebiete entwickelten Verfahren können nicht nur mit modernen Verfahren konkurrieren, sondern übertreffen diese sogar in vielerlei Hinsicht.

Compression, Pose Tracking, and Halftoning

Thesis for obtaining the degree of a doctor of the engineering sciences of the
natural-technical faculties of the Saarland University

by

Christian Schmaltz

Saarbrücken, Germany

2012

Partly supported by DFG (We 2602/5-1, We 2602/5-2, We2602/9-1) and the graduate school of Saarland University

Mathematical Image Analysis Group, Faculty of Mathematics and Computer Science
Saarland University, 66041 Saarbrücken

Abstract

This thesis deals with algorithms for four challenging research topics. First, we introduce an image compression algorithm which saves only a small set of the points in the image and reconstructs the remainder of the image using a partial differential equation (PDE). To reduce the amount of memory necessary to store the positions of these points, all points are determined from an adaptive rectangular subdivision of the original image. Among other things, we evaluate different PDEs, ways how to choose the saved points, and entropy coders. Using these and a number of other careful optimisations, the space required to store images is reduced significantly. Experiments on standard test images illustrate that this approach yields much higher compression ratios than JPEG, and even outperforms JPEG 2000.

Our second contribution is an algorithm to follow the so-called *poses* of moving objects within a video sequence, whereby the pose consists of the 3-D position, orientation, and any internal parameters – such as angles between joints – that might be present. This tracking step is achieved by minimising an energy functional that directly depends on the pose parameters. It can simultaneously follow the movement of multiple objects while being robust against both self-occlusions as well as occlusions between different objects. Even though various extensions allow to use prior knowledge, e.g. about the background image or positional constraints, our algorithm works well even if no such knowledge is available.

We evaluate our tracking algorithm using a large number of sequences consisting of more than 15000 images and more than a dozen different objects. This includes a sequence of a standard benchmark which shows that the results of our algorithm are among the best found in the literature.

In the third part of this thesis, we deal with *halftoning*, which is the task to approximate images with continuous grey values using black dots on a white background. Such algorithms are well-known from printers and fax machines, which can only generate pure black dots on white paper.

The halftoning algorithm we developed is based upon the idea to model black points as charged particles in a force field, which is created from the image. Using a mixture of repulsive forces between particles and attractive forces stemming from the image, the particles distribute in such a way that a good approximation of the original image is obtained. We also introduce a number of extensions to this simple and flexible approach to handle specific applications. Thus, we can create results containing varying dot sizes, different colours, or lines, among others.

Finally, we illustrate that our algorithms from the three topics image compression, 3-D pose tracking, and dithering can be combined to several video compression algorithms. Recent video compression algorithms, such as the members of the MPEG or H.26x family, use image transformations to store individual frames, and motion compensation between (not necessarily successive) frames. The video codec presented here is specialised for applications with static backgrounds, i.e. for applications such as traffic or security surveillance, or video conferencing. Hence, we can use an approach which stores fore- and background independently of each other.

Our video compression algorithm first tracks the moving foreground objects. Using the obtained poses and a compressed version of the background image, as well as other information such as 3-D object models and a projection matrix, an approximation of the original video is reconstructed. However, due to imprecise 3-D models, tracking failures, lighting changes, and other problems, the quality of the reconstructed video is often unsatisfying. Thus, an additional step is applied that accounts for these problems by storing an approximation of the difference image between the original frame and its reconstruction. Similar to our approach to store the background image, we save only a small set of points and reconstruct the remainder using a PDE. To this end, we use our halftoning algorithm to find appropriate point positions. Although the presented video compression algorithm is only at the beginning of its development, first experiments already show that the quality of videos compressed with this approach can yield a significantly improved quality compared to those obtained from state-of-the-art algorithms such as *MPEG-4*.

To recapitulate, we introduce algorithms for image compression, pose tracking, and halftoning, and combine these methods to video compression algorithms. Furthermore, we illustrate that the results of our methods do not only keep up with modern algorithms, but outperform them in many aspects.

To Sabine

Acknowledgements

"No man is wise enough by himself."

Titus Maccius Plautus (254 BC – 184 BC)
in *Miles Gloriosus*

To begin with, I would like to thank *Prof. Joachim Weickert* for encouraging me to start a PhD, giving me the possibility to do so, guiding me along the way, and for supervising my dissertation. Without him, I would probably not have started a PhD at all.

Furthermore, I want to thank *Prof. Bodo Rosenhahn* from the Leibniz University of Hannover for his constant support and help, and his various suggestions and propositions. He made my work in tracking possible.

Moreover, my thanks go to *Prof. Thomas Brox* from the University of Freiburg who, together with Prof. Bodo Rosenhahn, wrote the source code on which my tracking code is based, and who also provided valuable comments concerning my work and my paper drafts. Especially, our tracking papers would read much worse without his help.

Next, I would like to thank *Pascal Gwosdek*, with who I performed joint work in image halftoning. Without his suggestions, help and his implementation of our algorithms in CUDA, the experiments in Chapter 4 would probably still be running.

Further thanks got to *Andrés Bruhn* for his suggestions concerning our work in halftoning, and for his fast implementation of inpainting using EED. Without this code, the experiments shown in Chapter 3 could not have been finished so quickly.

Additionally, my thanks go to the *German Research Foundation* (*DFG*) for financing parts of my research. A large part of the work presented in this thesis has been done within the project "Verknüpfung von 3D-Formenwissen und Lageschätzung mit Bildsegmentierung". This project was done in cooperation with *Lennart Wietzke* from the the group of *Prof. Gerald Sommer* from the Christian-Albrechts-University of Kiel.

As last (but not least) co-authors, my thanks go to *Prof. Daniel Cremers* from the Technical University of Munich, *Prof. Hans-Peter Seidel* from the Max-Planck-Institute for Compute Science, Saarbrücken, and to *Prof. Gabriele Steidl* and *Tanja Teuber* from the University of Kaiserslautern for fruitful cooperation.

Many thanks also go to former and current researchers of our group (and the associated group of Andrés Bruhn) who created an atmosphere in which working was a pleasure, and who also helped me in one or the other way. Apart from those already mentioned above, these are: *Michael Breuß*, *Bernhard Burgeth*, *Oliver Demetz*, *Stephan Didas*, *Irena Galić*, *Gabriela Ghimpeteanu*, *Sven Grewenig*, *Kai Hagenburg*, *Laurent Hoeltgen*, *Sebastian Hoffmann*, *Yong Chul Ju*, *Markus Mainberger*, *Rodrigo Moreno*, *Nico Persch*, *Pascal Peter*, *Luis Pizarro*, *Karteek Popuri*, *Agustin Salgado*, *Christopher Schroers*, *Simon Setzer*, *Natalia Slesareva*, *Levi Valgaerts*, *Sebastian Volz*, *Martin Welk*, and *Henning Zimmer*. Further thanks go to our current and former secretaries *Inge Pilarski*, *Christiane Kosater*, and *Ellen Wintringer*,

and system administrators *Stefan Kiefer*, *Michael Krause*, and *Marcus Hargarter* for taking care of all the bureaucratic and technical problems.

Furthermore, I would also like to thank Prof. Joachim Weickert, Prof. Bodo Rosenhahn, Sabine Schmaltz, Pascal Gwosdek, Andrés Bruhn, Pascal Peter, Verena Kremer, Markus Mainberger, and Bernhard Burgeth for proofreading parts of this thesis, and Prof. Joachim Weickert, Prof. Bodo Rosenhahn, and Prof. Luis Álvarez León for reviewing my thesis.

Finally, I thank my parents *Robert Schmaltz* and *Monika Schmaltz*, as well as my brother *Thomas Schmaltz* for all their support before and during my thesis. My special thanks go to my wife *Sabine*, for too many things to mention them all.

Saarbrücken, Jan. 12, 2012 Christian Schmaltz

Contents

1 Introduction

"People often say that motivation doesn't last.
Well, neither does bathing – that's why we recommend it daily."

Hilary Hinton "Zig" Ziglar (*1926)

In image processing, visual computing, and computer vision, there are a huge number of unsolved problems. In this thesis, we discuss four of these problems, namely 3-D pose tracking, halftoning, as well as image and video compression.

We start with lossy *image compression*, which is the task to store an image using as few data as possible without deteriorating the image. Due to the huge amount and increasing resolution of images that are being created and viewed each day, image compression is still a topic of ongoing research. For example, more than 250 million photos are uploaded to the website "Facebook" on an average day [76].

In the closely related task of *video compression*, the amount of data necessary to represent a sequence of images is reduced. Only considering the website "YouTube", about two billion videos are watched each day, and 24 hours of video data is uploaded to this page every minute [302]. These numbers clearly indicate that there is a need for good image and video compression algorithms, as the amount of images and videos that can be stored and viewed simultaneously depends on their size.

The task occurring in 3-D pose *tracking*, namely to follow an object in a video sequence, is present in many applications ranging from object grasping in robotics over human computer interaction and content-based video retrieval, to traffic or security surveillance. However, due to the large variety of possible problems such as occlusions or changing illumination conditions, it is still a topic of ongoing research.

In *halftoning*, or in the closely related areas of *sampling*, *stippling*, and *dithering*, the task is to distribute points or other objects according to a given density distribution. This is necessary

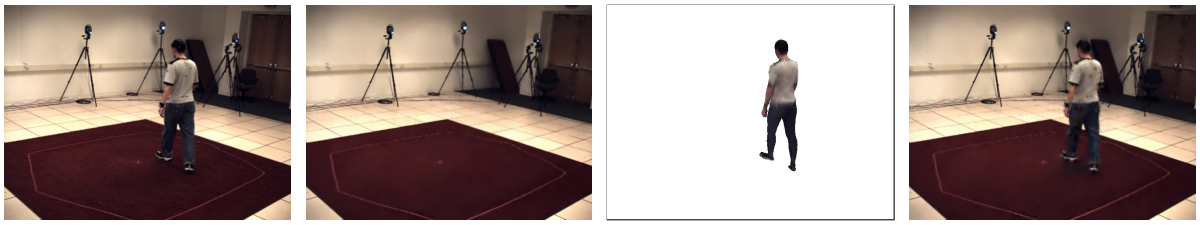


Figure 1.1.: Illustration of the different steps occurring in one of our video compression algorithms. **From left to right**: Initial image, compressed background image (compression ratio: 20 : 1), projected object model in tracked pose, and final result when using information from 2000 additional points where the locations are obtained by dithering.

in applications such as object placement, ray-tracing, or geometry processing. More example scenarios from tracking and halftoning are given in Chapter 4 and 9, respectively.

Let us now briefly sketch our algorithms, whose results are illustrated in Figure 1.2, 1.3, and 1.4 using the two example images shown in Figure 1.5.

Figure 1.2.: Compression results with compression ratios of 47:1 (left) and 95:1 (right).

Our image compression algorithm is based upon the idea already used in [88]: Store only a small subset of the pixels in the original image and reconstruct the remainder of the image using *partial differential equations*. To allow an efficient encoding of the pixel locations, only pixels on an adaptive regular grid are considered in our approach. Compared to [88], we introduced a number of new ideas and improvements including a novel brightness optimisation, an optimisation of free parameters occurring in the partial differential equations, and improved entropy coding schemes. More enhancements, details, and comparisons are given in Chapter 3.

The tracking algorithm we developed uses an uncoloured 3-D object model as input data. This model is projected onto the input image to segment the image into foreground and background region. By adapting the 2-D–3-D point correspondences obtained by this projection, we change the 3-D object position and orientation in such a way that the appearances of fore- and background differ as much as possible, or match known appearances estimated in previous frames. We explain various extensions, such as extensions to simultaneous tracking of multiple objects consisting of multiple internal parts, or new local ways to model the appearance of the background region.

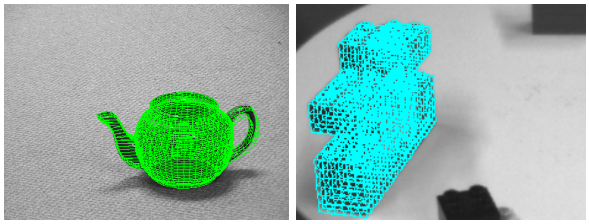


Figure 1.3.: 3-D pose tracking results illustrated by projecting the object in the estimated pose into the image.

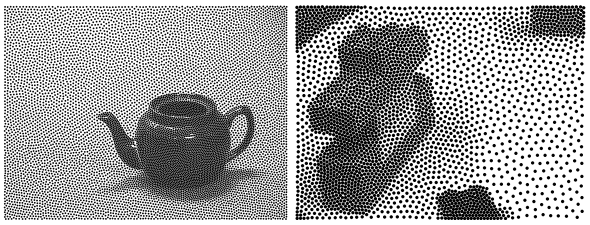


Figure 1.4.: Halftoning results with 10000 (left) and 3000 (right) dots.

Our halftoning algorithm is based upon the idea of electrostatic interactions of charged particles in an attractive force field computed from the input image. Thereby, each particle corresponds to one dot in the final image. This intuitive and flexible idea automatically binds the correct number of particles to each image region, independent of the size or shape of the region, and independent on the size of the dots in the output image. We further introduce several extensions of the simple but efficient idea to use electrostatic forces, which allow to adapt the algorithm to specific applications.

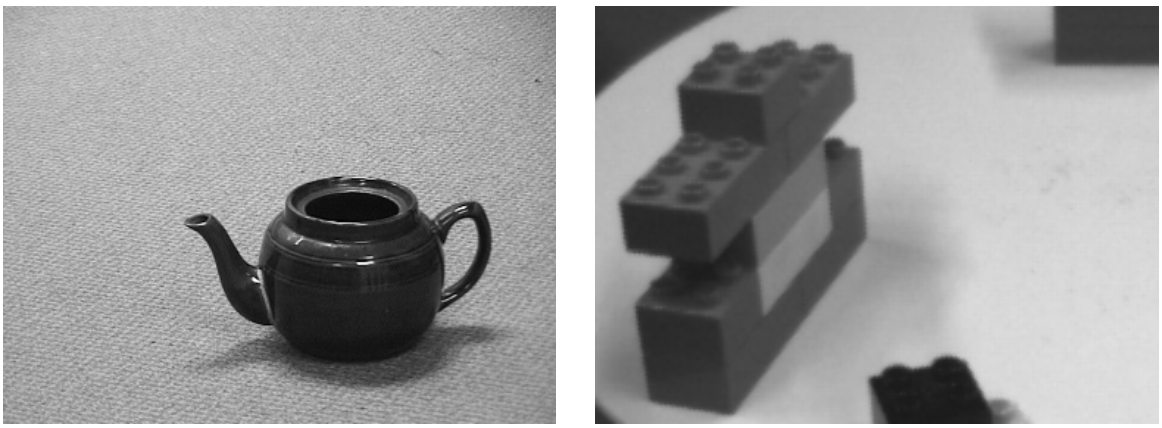


Figure 1.5.: Input images used for the illustrations in this chapter.

Our video compression algorithm is a so-called *model-based coding* scheme. These methods, which were introduced in [81], analyse the movement of one or several moving objects in the video sequence. The obtained information is then used to reconstruct an approximation of the video stream. Since object motions can typically be represented with few bytes, this approach allows very high compression ratios.

Thus, we first track objects moving in the video to reconstruct their 3-D position and orientation. This way, each image is split into fore- and background, which are compressed independently. We assume that the background remains approximately constant over the complete sequence. Thus, it is sufficient to compress the background once. To reconstruct the foreground region, the object model is projected onto the image using the pose determined by the tracking algorithm.

Additionally, we account for reconstruction inaccuracies due to inexact tracking, imprecise object models, shadows, or similar problems . This can be done by encoding the difference between original and reconstruction by a small set of image points whose positions slowly change from frame to frame. From this set, the complete difference image is reconstructed using image inpainting with homogeneous diffusion. To find the locations of the points in this set, we dither the magnitude of the *Laplacian* of the difference image. This choice was theoretically justified by Belhachmi *et al.* in [20]. An overview over the effects of these three parts is given in Figure 1.1.

As can be seen, our video compression approach combines our image compression, 3-D pose tracking, and dithering algorithms, which seemed unrelated at first glance. Detailed descriptions of our algorithms, as well as discussions of previous works, are given in the following chapters. Furthermore, we will present detailed evaluations which demonstrate that each of our novel methods yields results that are comparable to other state-of-the-art algorithms, and often even outperform them.

1.1. Outline

"A goal without a plan is just a wish."

Antoine de Saint-Exupery (1900 – 1944)

This thesis is organised as follows:

In Chapter 2, we introduce some background knowledge that is necessary to understand later chapters. The topics discussed in this chapter include PDE-based diffusion and image inpainting methods, the transformation of object models with Lie Groups and Lie algebras, an intuitive introduction to statistics, and a derivation of electrostatic forces in a pure 2-D plane.

As a first contribution, we present a detailed description of our compression algorithm for still images in Chapter 3. Thereby, we evaluate a number of different ways to select the stored points and test different inpainting operators, quantisation levels and entropy coders. As explained above, the algorithm we devised is based upon the ideas presented in [88], but clearly outperforms this approach by using several improvements discussed in detail in Chapter 3.

The next chapters deal with 3-D pose tracking. As our tracking algorithm uses many of the concepts introduced by Rosenhahn, Brox, and Weickert [219], we start by explaining their segmentation and tracking approaches in Chapter 4. The basic version of our novel pose tracker is introduced in Chapter 5, where we also sketch a number of extensions and show first experiments.

Chapter 6 extends this simple tracking approach such that a simultaneous tracking of multiple objects is possible. Since this allows to handle occlusions, tracking multiple objects even improves tracking results if only the 3-D position of one object is desired.

The ideas introduced in Chapter 6 are developed further in Chapter 7 to multiple parts of the same object model. The enhanced tracking algorithm uses internal object regions to improve tracking of single objects. This is especially useful when tracking kinematic chains.

As a last improvement to our tracking approach, we introduce a new region model in Chapter 8 to model locally varying appearances by using a mixture of densities. Our new model is spatially varying and uses a partitioning step guided by the tracking results. In particular when the background is unknown or moves, this model remarkably stabilised our tracking approach.

Chapter 9 introduces the novel halftoning algorithms we developed. They allow an accurate representation of a grey-scale image by few black dots on a white background. Although the algorithm is best suited for situations without restrictions on the locations of these dots, we will see that it also yields excellent results if such restrictions are present.

The algorithms described up to Chapter 9 are combined in Chapter 10 to three closely related video compression algorithms. Chapter 11 concludes this thesis and gives an outlook on possible directions of future research.

In addition, this thesis has two appendixes. Appendix A gives a detailed introduction to statistics. It accounts for the fact that this topic is not handled in detail in Section 2.7, where only an intuitive introduction is present. Appendix B contains a complete list of publications.

Finally, we have also added an index containing the most important terms, a glossary which can be used to look up the meaning of symbols, terms, and operators, and a list of figures.

2 Mathematical and Physical Foundations

"The mathematics is not there till we put it there."

Sir Arthur Eddington, English astronomer (1882 – 1944)
in *The Philosophy of Physical Science*

In this chapter, we introduce the mathematical and physical background necessary to understand the remainder of the thesis. We start by explaining diffusion and inpainting with partial differential equations, which will be used in our compression algorithm for single images. Then, the electrostatic particle interactions in a 2-D plane needed for our halftoning algorithm are derived in Section 2.2. After introducing the concepts of Lie groups and Lie algebras in Section 2.3, we illustrate in Section 2.4 how the movement of points can be described by elements of specific Lie groups and Lie algebras. This concept is extended to rigid objects and kinematic chains. Next, Section 2.5 give the representation of lines and planes used in this thesis. Finally, we introduce the camera model used throughout the thesis in Section 2.6, and illustrate the basic statistics necessary to explain how image regions are modelled in Section 2.7.

2.1. Diffusion and Inpainting With PDEs

"10^{50} *is a long way from infinity.*"

Daniel Shanks (1917 – 1996)
in *Solved and Unsolved Problems in Number Theory*

Diffusion is a widely-known everyday process responsible for balancing density or concentration differences, which occurs in a multitude of different branches of physics. Two often cited examples are the facts that a drop of ink added to a glass of water eventually colours the whole water, and that heat is distributed in a room or an object. However, one should be aware that convection has a stronger influence than diffusion in the first example, and that the second example illustrates heat conduction, which is described by the same equations as diffusion. More complex examples of diffusion include atomic diffusion or photon diffusion. The first is the process that allows helium atoms to travel through the walls of a balloon, resulting in a deflation of the balloon, while the second accounts for the fact that milk and chalk look different due to subsurface scattering.

Diffusion is also widely used in image processing: Here, we identify grey values in an image with concentrations. Given an initial image f with image domain $\Omega \subset \mathbb{R}^2$, we apply a diffusion process to compute the transport between different pixels, and thus new images.

Formally, we introduce a time variable t, and denote the image as a time-dependent function $u(\boldsymbol{x}, t)$: $\Omega \times \mathbb{R}_0^+ \mapsto \mathbb{R}$. This function is initialised as $u(\boldsymbol{x}, 0) = f(\boldsymbol{x})$, and diffusion is performed using some operator L. That is, the complete diffusion process is given by the *partial differential equation* (or *PDE*)

$$\partial_t u = Lu, \qquad u(\boldsymbol{x}, 0) = f(\boldsymbol{x}), \tag{2.1}$$

where L denotes some differential operator. In the simplest case, an isotropic linear diffusion operator $Lu = \Delta u$ is used. We will discuss this and many other differential operators later this section.

A related problem is given by *inpainting* or *interpolation*. In this scenario, the image is only given at some points while the missing parts of the image are to be reconstructed. One can imagine this scenario by a room in which some air conditioning units are placed. Given that each air conditioner heats or cools the surrounding air to a certain temperature, the task is to compute the temperature in each point in the room.

Again, let $f : \Omega \to \mathbb{R}$ be the original image to be reconstructed from a subset $K \subset \Omega$ of the image. The set K is called the *interpolation mask*. The task is to compute a reconstruction u of f such that u is equal to f in K,

$$u(\boldsymbol{x}) = f(\boldsymbol{x}) \quad \forall\, \boldsymbol{x} \in K, \tag{2.2}$$

and such that u is regular in some sense. For example, u can be required to be smooth. Both conditions are met by using the PDE

$$(1 - c_K)\, Lu - c_K\, (u - f) = 0 \tag{2.3}$$

for reconstruction with reflecting (i.e. homogeneous Neumann) boundary conditions. Again, L denotes some differential operator that ensures the requested smoothness properties, and c_K is the *characteristic function* of K, i.e. a function that is 1 at the specified pixel set K, and 0 elsewhere:

$$c_K(\boldsymbol{x}) := \begin{cases} 1 & \text{if } \boldsymbol{x} \in K \\ 0 & \text{else} \end{cases}. \tag{2.4}$$

Note that this PDE always keeps the given pixels fixed. Thus, the PDE in Equation (2.3) can also be written as the simplified PDE

$$Lu = 0 \tag{2.5}$$

on $\Omega \setminus K$, with Dirichlet boundary conditions (2.2) on K, and homogeneous Neumann boundary conditions on the boundary of Ω.

To solve Equation (2.5), we again introduce an artificial time parameter t and compute the steady state of the evolution equation

$$\partial_t u = Lu. \tag{2.6}$$

Note one important difference between PDE-based diffusion and inpainting: In diffusion, we are interested in the result after a finite time. For inpainting, however, we are looking for the *steady-state* obtained for $t \to \infty$.

It is also possible to derive Equation 2.3 as a generalisation of spline interpolation and variational regularisation. Details can be found in [290].

In the remainder of this section, we introduce several differential operators L that have been proposed in image processing. Each of these operators can also be used for image interpolation using the framework explained above.

2.1.1. Linear Isotropic Diffusion Operator

The simplest differential operator L that can be used for diffusion and inpainting is the linear operator already given as example in the last section. It is based on the *Laplacian* Δu:

$$Lu = \Delta u = \partial_{x_1}^2 u + \partial_{x_2}^2 u = \operatorname{div}(\boldsymbol{\nabla} u), \tag{2.7}$$

and was originally introduced in [119] in the context of scale-space evolutions [289]. It is well-known [108, 202] that diffusion with this operator corresponds to smoothing the initial image with a 2-D Gaussian K_σ with standard deviation $\sigma = \sqrt{2t}$ and mean $\mu = \mathbf{0}$:

$$K_\sigma(\boldsymbol{x}) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{|\boldsymbol{x} - \boldsymbol{\mu}|^2}{2\sigma^2}\right) \tag{2.8}$$

Consequently, this diffusion process is *linear*, i.e. the diffusion strength is independent of the image brightness, and *isotropic*, i.e. no direction is preferred. Due to these properties, this process is called *linear isotropic diffusion*. Another name often used in this setting is *homogeneous diffusion*.

This diffusion scheme is not only simple, but also fulfils useful properties the most important of which is the *maximum-minimum principle*:

$$\inf_{\boldsymbol{y}\in\Omega} f(\boldsymbol{y}) \leq u(\boldsymbol{x}, t) \leq \sup_{\boldsymbol{y}\in\Omega} f(\boldsymbol{y}) \quad \forall \boldsymbol{x} \in \Omega, t \in \mathbb{R}^+. \tag{2.9}$$

For inpainting, this means that the reconstructed image has pixels whose brightness values are between those of the specified data:

$$\inf_{\boldsymbol{y}\in K} f(\boldsymbol{y}) \leq u(\boldsymbol{x}) \leq \sup_{\boldsymbol{y}\in K} f(\boldsymbol{y}) \quad \forall \boldsymbol{x} \in \Omega. \tag{2.10}$$

This maximum-minimum principle is a strong stability result, since it guarantees that no over- and undershoots will appear.

2.1.2. Polyharmonic Operators

The linear operator explained in the last section can easily be extended to order p:

$$Lu = (-1)^{p+1} \Delta^p u. \tag{2.11}$$

The two higher-order representatives most widely used are the *biharmonic operator*

$$Lu = -\Delta^2 u, \tag{2.12}$$

one obtains for $p = 2$, and the *triharmonic operator*

$$Lu = \Delta^3 u \tag{2.13}$$

with $p = 3$. These two operators can be regarded as rotationally invariant multidimensional generalisations of cubic and quintic spline interpolation [68], respectively. Their corresponding Green's functions belong to the family of radial basis functions [42].

One big disadvantage of polyharmonic operators is the fact that they violate the maximum-minimum principle for $p \geq 2$. This is a typical observation for higher order differential operators.

2.1.3. Nonlinear Isotropic Diffusion

Homogeneous diffusion does not differentiate between different image parts. This space-invariance is a principal problem, since semantically important edges and the interior of a region are treated in the same way. Thus, it was proposed in [201] to reduce the diffusion at the edges of the developing image, i.e. where $|\boldsymbol{\nabla} u|$ is large. This is accomplished by multiplying $\boldsymbol{\nabla} u$ in Equation (2.7) with $g(|\boldsymbol{\nabla} u|^2)$ before the divergence operator is applied. Here, g is a so-called *diffusivity function*, which is a monotonically decreasing function in its argument $|\boldsymbol{\nabla} u|^2$ and satisfies $g(0) = 1$.

As a result, one obtains the following operator:

$$Lu = \operatorname{div}(g(|\boldsymbol{\nabla} u|^2)\, \boldsymbol{\nabla} u). \tag{2.14}$$

Here, we always use the *Charbonnier diffusivity* [48] given by

$$g(s^2) := \frac{1}{\sqrt{1 + \frac{s^2}{\lambda^2}}} = \frac{\lambda}{\sqrt{\lambda^2 + s^2}}, \tag{2.15}$$

where $\lambda > 0$ is a *contrast parameter*, and call the corresponding interpolation operator *Charbonnier diffusion*. There are other diffusivity functions that can be used, but in the interpolation settings used in this thesis, the Charbonnier diffusivity works best. Note that for small λ, the Charbonnier diffusivity is a scaled approximation to the *total variation* (TV) diffusivity [228] given by:

$$g(s^2) := \frac{1}{|s|}. \tag{2.16}$$

Thus, interpolation with the Charbonnier diffusivity approximates TV interpolation.

In some settings, including those used in this thesis, it is beneficial to regularise the Charbonnier operator by considering

$$Lu = \operatorname{div}(g(|\boldsymbol{\nabla} u_\sigma|^2)\, \boldsymbol{\nabla} u), \tag{2.17}$$

where $u_\sigma := K_\sigma * u$ is a smoothed version of the image u, obtained by convolving u with a Gaussian K_σ with standard deviation σ. We call Equation (2.17) the *regularised Charbonnier*

operator. This regularisation was first introduced in [44] in the context of diffusion filtering to stabilise the Perona-Malik filter [201] under noise, and to make this filter well-posed.

The operators given in Equations (2.14) and (2.17) are isotropic and *nonlinear*, i.e. they depend nonlinearly on the evolving image. Thus, they belong to the class of nonlinear isotropic diffusion filters, as they are space-variant but direction-invariant. As in the case of linear isotropic diffusion, the maximum-minimum principle is fulfilled for these filters.

2.1.4. Edge-Enhancing Anisotropic Diffusion (EED)

The isotropic nonlinear operators described in the last section adapt to image structures by changing the amount of diffusion at image edges. However, they still perform the same amount of diffusion in all directions. This is no longer the case for the anisotropic filter introduced in [287]. This filter, which is called *edge-enhancing (anisotropic) diffusion* (or *EED*) replaces the scalar-valued diffusivity $g(|\nabla u_\sigma|^2)$ in Equation (2.17) by the matrix-valued diffusion tensor $g(\nabla u_\sigma \nabla u_\sigma^\top)$. Again, we use the Charbonnier diffusivity given in Equation (2.15) because of its favourable performance, especially for interpolation tasks. The resulting operator is given by:

$$Lu = \operatorname{div}(g(\nabla u_\sigma \nabla u_\sigma^\top)\, \nabla u). \tag{2.18}$$

Since $\nabla u_\sigma \nabla u_\sigma^\top$ is a symmetric 2×2 matrix, applying the function g only changes the eigenvalues of the matrix: The first eigenvalue $|\nabla u_\sigma|^2$, which corresponds to the eigenvector parallel to the edge ∇u_σ, becomes $g(|\nabla u_\sigma|^2)$, resulting in little diffusion across the edge. The second eigenvalue 0 corresponding to an eigenvalue orthogonal to the edge becomes 1, resulting in full diffusion along the edge. Consequently, EED is designed in such a way that it also regards the direction of edges, and not only their location. Also note that EED fulfils the maximum-minimum principle in the continuous setting.

When EED is used for interpolation tasks, the anisotropy is even propagated to the neighbourhood of the given data. We will see shortly that this leads to improved interpolation results.

2.1.5. Absolute Minimal Lipschitz Extension (AMLE)

As a last operator, we introduce the *absolute minimal Lipschitz extension* (or *AMLE*) introduced in [10] and given by

$$Lu = u_{\boldsymbol{\eta}\boldsymbol{\eta}}, \tag{2.19}$$

where $\boldsymbol{\eta}$ denotes the normalised gradient:

$$\boldsymbol{\eta} = \frac{\nabla u}{|\nabla u|}. \tag{2.20}$$

This operator is anisotropic, as it diffuses only in the direction of the gradient. Although it was axiomatically justified for interpolation tasks in [43] and also fulfils the maximum-minimum principle, we will see that it usually gives worse results than the other operators.

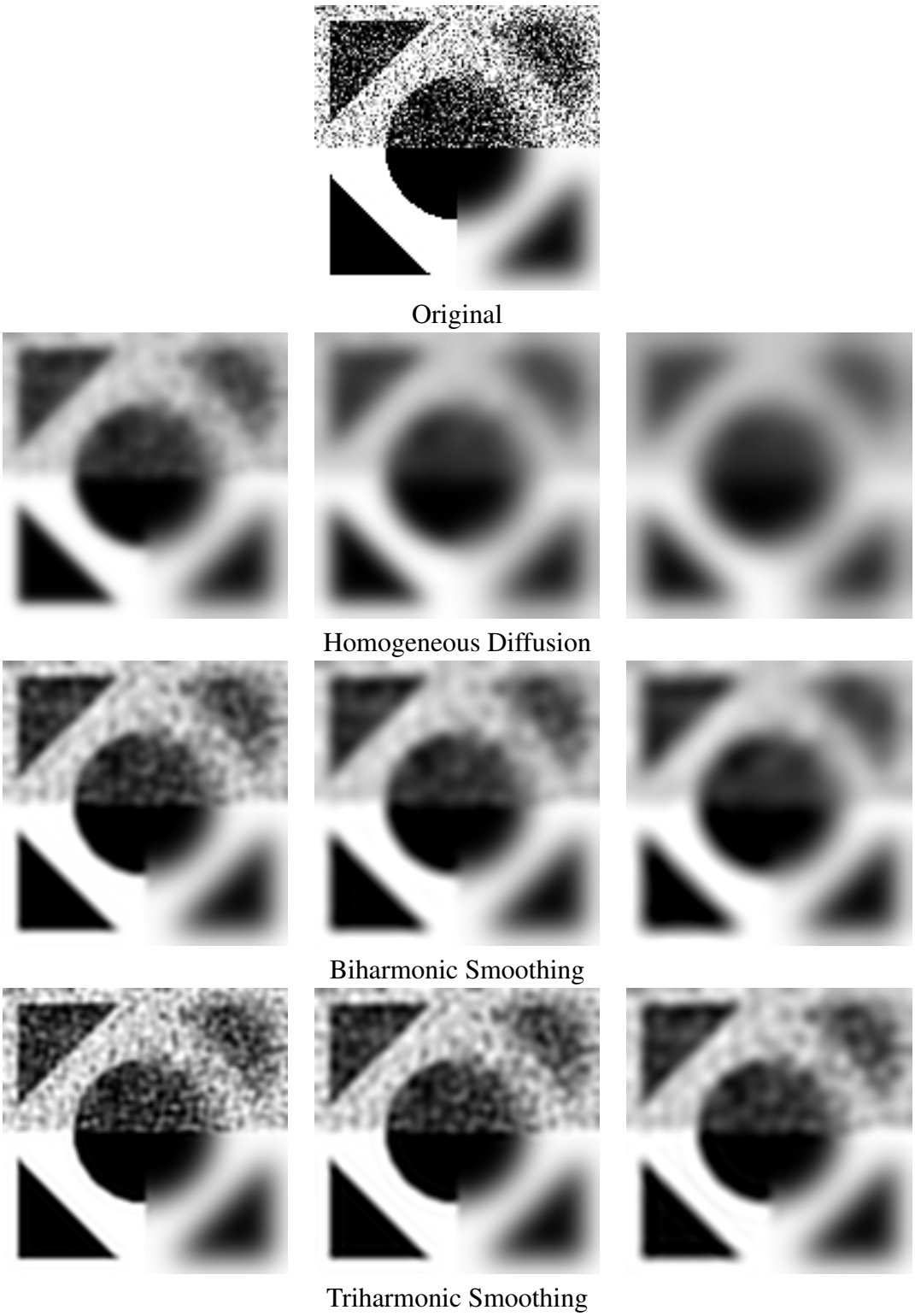


Figure 2.1.: Evolution of a partially noisy and partially blurred image under different diffusion operators. **First row**: Initial image. **Second row**: Homogeneous diffusion, $t = 3, 10, 20$. **Third row**: Biharmonic smoothing, $t = 1.2, 6, 30$. **Forth row**: Triharmonic smoothing, $t = 0.12, 0.6, 6$.

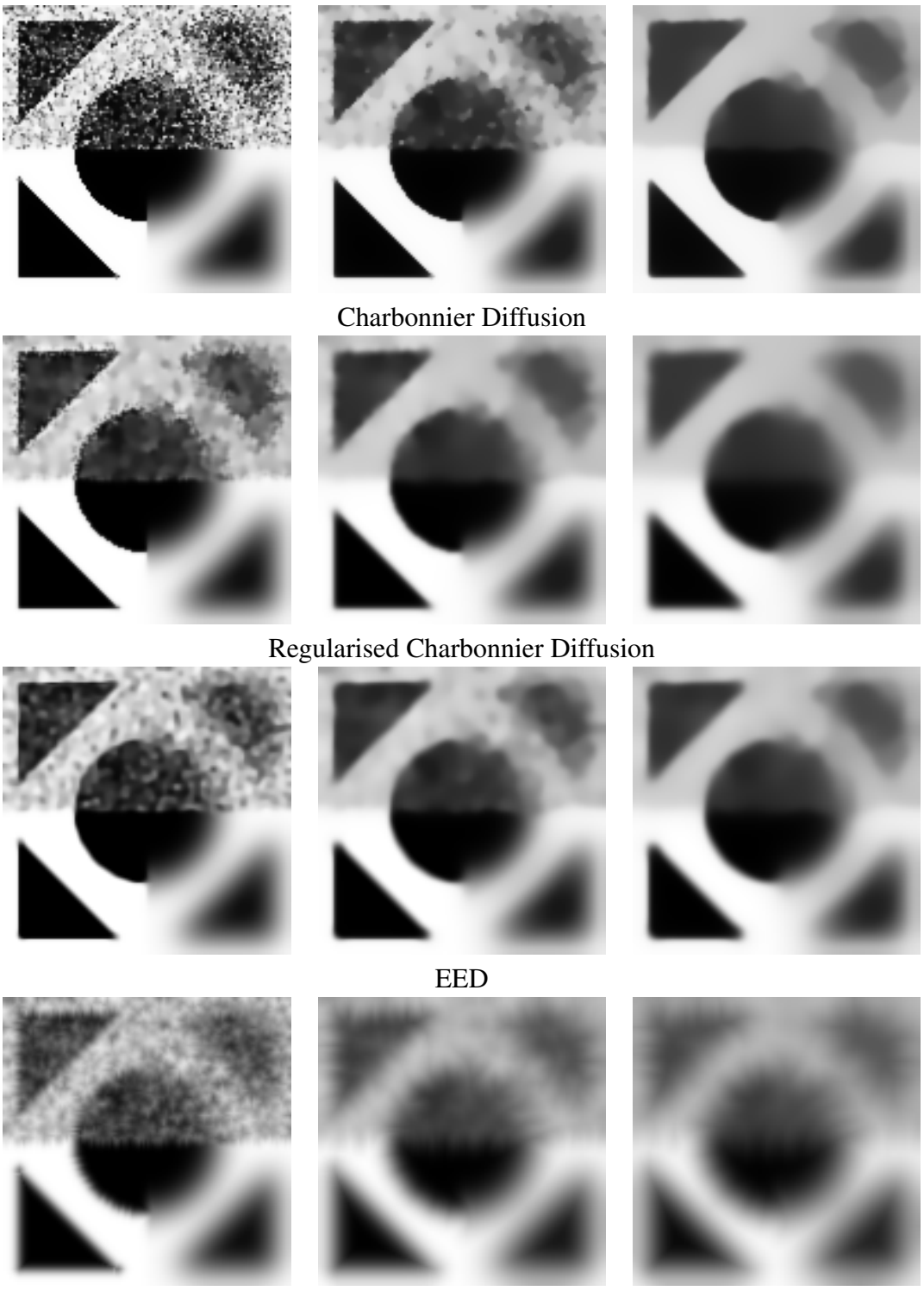


Figure 2.2.: Evolution of a partially noise and partially blurred image under different diffusion operators. The initial image is shown in Figure 2.1. **First row**: Charbonnier diffusion, $\lambda = 0.8$, $t = 8, 40, 100$. **Second row**: Regularised Charbonnier diffusion, $\lambda = 0.8, \sigma = 2.0$, $t = 8, 40, 100$. **Third row**: Edge enhancing diffusion, $\lambda = 3.0, \sigma = 0.8$, $t = 2, 10, 20$. **Forth row**: Absolute minimal Lipschitz extension, $t = 2, 10, 20$.

Figures 2.1 and 2.2 illustrate the effects of the different operators with different stopping times on a synthetic test image. The right part of the image is blurred, while noise was added to the upper part of the original image after blurring. Thus, it can be seen how the different operators handle noise, as well as sharp and blurry image regions.

2.2. Electrostatic Interactions in 2-D

"All science is either physics or stamp collecting."

Sir Ernest Rutherford (1871 – 1937)

It is common knowledge that a charged particle creates an electric field which exerts forces on other charged particles in this field. As the French physicist C. Coulomb published in 1783, the force acting on a charge q_1 at position $\boldsymbol{p}_1$ due to a particle q_2 at position $\boldsymbol{p}_2$ is given by

$$\boldsymbol{F} = \frac{1}{4\pi\varepsilon_0}\frac{q_1 q_2(\boldsymbol{p}_2 - \boldsymbol{p}_1)}{\|\boldsymbol{p}_2 - \boldsymbol{p}_1\|^3} = \frac{1}{4\pi\varepsilon_0}\frac{q_1 q_2}{\|\boldsymbol{p}_2 - \boldsymbol{p}_1\|^2}\mathbf{e}_{2,1}, \tag{2.21}$$

where $\boldsymbol{e}_{2,1}$ is the unit vector from $\boldsymbol{p}_2$ to $\boldsymbol{p}_1$,

$$\boldsymbol{e}_{2,1} := \frac{\boldsymbol{p}_1 - \boldsymbol{p}_2}{\|\boldsymbol{p}_1 - \boldsymbol{p}_2\|}, \tag{2.22}$$

and where ε_0 is the electric constant, which can be derived from the magnetic constant $\mu_0 = 4\pi \cdot 10^{-7}\frac{H}{m}$ and the speed of light in vacuum c_0 as

$$\varepsilon_0 = \frac{1}{\mu_0 c_0^2} \approx 8.85 \cdot 10^{-12}\frac{F}{m}. \tag{2.23}$$

According to Equation (2.21), the magnitude of the force is inversely proportional to the square of the distance between the particle. As always in physics, it is unclear if this really holds in the real world, as it cannot be proven to be true, but can only be observed by (repeated) experiments. For the real world, it was shown that the exponent lies in the interval $[2 - 0.4 \cdot 10^{-16}, 2 + 5.8 \cdot 10^{-16}]$ [296], indicating that 2 is at least a very good approximation to the actual behaviour.

However, all this is only true for a 3-D world. In the remainder of this section, we will derive the interaction between two charged particles in a pure 2-D world.

First of all, we consider the electric field generated by the first particle, i.e. by a particle with charge q_2 and position $\boldsymbol{p_2}$ and compute the *flux* Φ of an *electric field* $\boldsymbol{E}$ generated through a circular curve c_r of radius r [173]:

$$\Phi = \oint_{c_r} \boldsymbol{E} d\boldsymbol{A} \tag{2.24}$$

where $\boldsymbol{E}$ is the electrical field and $\boldsymbol{A}$ is a surface vector pointing outwards. Since $\boldsymbol{E}$ is parallel to $\boldsymbol{A}$ at each point on the circle, their scalar product is equal to the product of their magnitudes. Thus, we have:

$$\Phi = \oint_{c_r} \boldsymbol{E} d\boldsymbol{A} = \oint_{c_r} |\boldsymbol{E}| d|\boldsymbol{A}| = |\boldsymbol{E}| \oint_{c_r} d|\boldsymbol{A}| = 2\pi r |\boldsymbol{E}|, \tag{2.25}$$

Furthermore, by the theorem of Gauß-Ostrogradski [173], this electric flux is also equal to

$$\Phi = \frac{q_2}{\varepsilon_0}. \tag{2.26}$$

Combing the last two equations leads to

$$|\boldsymbol{E}| = \frac{q_2}{2\pi\varepsilon_0 r}. \tag{2.27}$$

Using the fact that the distance between the two particles is r, we can compute the repulsive force $F_{r,1,2}$ acting on the first particle as:

$$|\boldsymbol{F}_{r,1,2}| = q_1 |\boldsymbol{E}| = \frac{q_1 q_2}{2\pi\varepsilon_0 r} = \frac{k q_1 q_2}{\|\boldsymbol{p}_2 - \boldsymbol{p}_1\|}, \tag{2.28}$$

where we use $k := \frac{1}{2\pi\varepsilon_0}$ as abbreviation. Note that k equals twice Coulomb's constant. Furthermore, we know that this force acts in direction $\boldsymbol{e}_{2,1} = -\boldsymbol{e}_{1,2}$. Thus, we get:

$$\boldsymbol{F}_{r,1,2} = -\frac{k \cdot q_1 \cdot q_2}{\|\boldsymbol{p}_2 - \boldsymbol{p}_1\|} \boldsymbol{e}_{1,2}. \tag{2.29}$$

The negative sign is used to stress the repulsive character of this force. Note the difference to the 3-D world: In 3-D, we must use a sphere with surface area $O_3(r) := 4\pi r^2$ in Equation (2.25), while a circle with "surface area" $O_2(r) := 2\pi r$ (both with radius r) must be used in 2-D. This results in a linear instead of a quadratic dependency on the distance and a factor of 2, since $O_3(r)/O_2(r) = 2r$.

Obviously, it is impossible to check if the strength of the forces between charged particles is really given by Equation (2.29) in a pure 2-D world. However, the same equation can also be derived in 3-D when regarding infinitely long, thin, charged cylinders aligned parallel to each other. In such a scenario, the same forces act on each point of each cylinder. Thus, the cylinders will always stay parallel. Furthermore, it can easily be computed that the forces acting on the cylinders are given by Equation (2.29). Since the described 2-D scenario corresponds to a plane perpendicular to the cylinders, this alternative derivation shows that Equation (2.29) is sound.

2.3. Lie Groups and Lie Algebras

"In mathematics you don't understand things. You just get used to them."

Johann von Neumann (1903 – 1957)

Next, we give a short introduction to *Lie groups* and *Lie algebras*. For more detailed introductions, we refer to [183] and [242].

To introduce these advanced concepts, we need some additional definitions. We start with the definition of a *group*, which is well-known in mathematics. Nevertheless, we will repeat this very basic concept for the sake of completeness:

Definition 2.A: (Group) A *group* $(G, \cdot)$ is a set G together with a binary operation $\cdot : G \times G \mapsto G$ that fulfils the following axioms:

- *Associativity*: $\forall\, a, b, c \in G: \quad (a \cdot b) \cdot c = a \cdot (b \cdot c)$
- Existence of an *identity element*: $\exists\, e \in G\, \forall\, a \in G: \quad e \cdot a = a \cdot e = a$
- Existence of *inverse elements*: $\forall\, a \in G\, \exists\, b \in G: \quad a \cdot b = b \cdot a = e$, where e is the identity element.

A *manifold* is a more complex mathematical concept. Although we will use manifolds, we only give an intuitive "definition" here because the details are not of importance in the remainder of this thesis. However, manifolds are necessary to define Lie groups and their corresponding Lie algebras (see below).

Explanation 2.B: (Manifold) An n-dimensional *manifold* is a mathematical space in which every point has a neighbourhood that resembles the Euclidean space $\mathbb{R}^n$.

Lines and circles are examples for 1-dimensional manifolds, and the surface of any number of non-intersecting spheres is an example of a 2-dimensional manifold, because the surrounding of each surface point of each sphere can be mapped to a disk. Obviously, the Euclidean space $\mathbb{R}^3$ is a 3-dimensional manifold.

Now, we can define Lie groups:

Definition 2.C: (Lie group) A *Lie group* is a group that is also a smooth (infinitely differentiable) manifold in which the multiplication and inversion operators are smooth.

When dealing with a Lie group, one often considers the associated *Lie algebra*:

Definition 2.D: (Lie algebra) A *Lie algebra* g is a vector space over a field F, together with a binary operation (usually denoted with brackets $[\cdot, \cdot]$ and called the *Lie bracket* or *commutator*) with the following properties:

- *Bilinearity*: $\forall\, X, Y, Z \in g, \quad a, b \in F$:
  $[aX + bY, Z] = a[X, Z] + b[Y, Z], \quad [Z, aX + bY] = a[Z, X] + b[Z, Y]$
- *Skew-symmetry*[1] (or *Anticommutativity*): $\forall\, X, Y \in g : \quad [X, Y] = -[Y, X]$
- *Jacobi identity*: $\forall\, X, Y, Z \in g : \quad [X, [Y, Z]] + [Y, [Z, X]] + [Z, [X, Y]] = 0$

Although every Lie group has an associated Lie algebra, we will only describe how to construct those algebras corresponding to Lie groups G that are also matrix groups, since we will only consider such Lie algebras. In this case, the Lie algebra g consists of all matrices $\boldsymbol{X}$ for which the property $\exp(\theta \boldsymbol{X}) \in G$ holds for all real numbers θ:

$$g = \{\boldsymbol{X} : \quad \forall\, \theta \in \mathbb{R} \quad \exp(\theta \boldsymbol{X}) \in G\} \tag{2.30}$$

In this thesis, we will only consider two specific Lie groups, namely the group of rotation matrices $SO(3)$ and the group of rigid transformations $SE(3)$. Those two Lie groups are defined as follows:

Definition 2.E: ($SO(3)$ **and** $SE(3)$**)** The Lie groups $SO(3)$ and $SE(3)$ are given by

$$SO(3) := \{\boldsymbol{R} \in \mathbb{R}^{3\times 3} : \det(\boldsymbol{R}) = 1\} \tag{2.31}$$

and

$$SE(3) := \left\{ \boldsymbol{M} := \begin{pmatrix} \boldsymbol{R}_{3\times 3} & \boldsymbol{t}_{3\times 1} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix} \in GL(4, \mathbb{R}) : \boldsymbol{R} \in SO(3), \boldsymbol{t} \in \mathbb{R}^3 \right\}. \tag{2.32}$$

Elements of these groups can be used to describe rotations ($SO(3)$) and affine movements ($SE(3)$) in 3-D space. Thus, the movement of a 3-D point – and therefore also the movement of a rigid 3-D object – can be completely described by an element $\boldsymbol{\xi}$ of $SE(3)$. The Lie algebras corresponding to these two Lie groups are given in the following theorem:

Theorem 2.F: ($SO(3)$ **and** $se(3)$**)** For the two Lie groups $SO(3)$ and $SE(3)$, the corresponding Lie algebras are

$$so(3) := \{\boldsymbol{A} \in \mathbb{R}^{3\times 3} | \boldsymbol{A}^\top = -\boldsymbol{A}\}, \tag{2.33}$$

and

$$se(3) := \{(\boldsymbol{\nu}, \boldsymbol{\omega}) | \boldsymbol{\nu} \in \mathbb{R}^3, \boldsymbol{\omega} \in so(3)\}, \tag{2.34}$$

respectively. The elements of $se(3)$ are called *twists*.

[1] This condition must be replaced by the stronger condition $\forall X \in g : \quad [X, X] = 0$ if the characteristic of the field F is 2. However, this will never be the case in this thesis.

Depending on the situation, we will write the elements of these two algebras either as vectors $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3) \in so(3)$, $\boldsymbol{\xi} = (\boldsymbol{v}, \boldsymbol{\omega}) = (v_1, v_2, v_3, \omega_1, \omega_2, \omega_3) \in se(3)$ or we transform them to matrices by applying the so-called *wedge* operator $\wedge$:

$$\boldsymbol{\omega}^\wedge = \hat{\boldsymbol{\omega}} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \in so(3), \tag{2.35}$$

$$\boldsymbol{\xi}^\wedge = \hat{\boldsymbol{\xi}} = \begin{pmatrix} \hat{\boldsymbol{\omega}} & \boldsymbol{v} \\ \mathbf{0}_{1\times 3} & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 0 & -\omega_1 & v_2 \\ -\omega_2 & \omega_1 & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{pmatrix} \in se(3), \tag{2.36}$$

Note that, for all $\boldsymbol{a} \in so(3), \boldsymbol{b} \in \mathbb{R}^3$, the property $\boldsymbol{a} \times \boldsymbol{b} = \hat{\boldsymbol{a}}\boldsymbol{b}$ holds. Here, $\times$ denotes the cross product of two vectors.

The transformation from matrix to vector notation is done by the *vee* operator $\vee$. Thus, $(\hat{\boldsymbol{\omega}})^\vee = \boldsymbol{\omega}$ for all $\boldsymbol{\omega}$ in $so(3)$ or $se(3)$. Note that the vee operator is only defined for the special types of matrices shown in Equations (2.35) and (2.36).

As elements of $SO(3)$ denote rotations, it is not surprising that an element $\boldsymbol{\omega} \in so(3)$ can be thought of as a rotation as well; For $\boldsymbol{\omega} = (0, 0, 0)$, we have the identity. For $\boldsymbol{\omega} \neq (0, 0, 0)$, the unit vector $\frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}$ corresponds to the rotation axis and the scalar $\|\boldsymbol{\omega}\|$ to the magnitude of the rotation in radians. Thus, multiplying any $\boldsymbol{\omega} \in so(3)$ with a (not necessarily positive) multiple of 2π yields a new element corresponding to the same rotation.

Similarly, twists denote affine motions, i.e. rotations and translations. Thus, the movement of a rigid object can be described by a twist.

2.3.1. Conversion between Lie Groups and Lie Algebras

As stated above, it is necessary to compute the exponential of a matrix to convert elements of a Lie algebra to elements of the corresponding Lie group. Evaluating the exponential of a matrix is often cumbersome, though. For the Lie groups we are interested in, however, there are efficient formulas to evaluate this exponential function.

Formula 2.G: (Rodriguez formula) For all $\boldsymbol{\omega} \in so(3)$, the *Rodriguez formula* [183] states that

$$\exp(\hat{\boldsymbol{\omega}}) = \begin{cases} \boldsymbol{I} + \frac{\hat{\boldsymbol{\omega}}}{\|\boldsymbol{\omega}\|} \sin\|\boldsymbol{\omega}\| + \frac{\hat{\boldsymbol{\omega}}^2}{\|\boldsymbol{\omega}\|^2}(1 - \cos\|\boldsymbol{\omega}\|) & \text{for } \boldsymbol{\omega} \neq 0 \\ \boldsymbol{I} & \text{else,} \end{cases} \tag{2.37}$$

where $\boldsymbol{I}$ is the 3×3 identity matrix. Similarly, for elements $\boldsymbol{\xi} = (\boldsymbol{v}, \boldsymbol{\omega})$ of the Lie algebra $se(3)$, it can be shown that

$$\exp(\hat{\boldsymbol{\xi}}) = \begin{cases} \begin{pmatrix} \exp(\hat{\boldsymbol{\omega}}) & \left((\boldsymbol{I} - \exp(\hat{\boldsymbol{\omega}}))\hat{\boldsymbol{\omega}} + \boldsymbol{\omega}^\top\boldsymbol{\omega}\right)\frac{\boldsymbol{v}}{\|\boldsymbol{\omega}\|^2} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix} & \text{for } \boldsymbol{\omega} \neq 0 \\ \begin{pmatrix} \boldsymbol{I} & \boldsymbol{v} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix} & \text{else.} \end{cases} \tag{2.38}$$

To compute the length $\|\boldsymbol{\omega}\|$, we use the standard L^2-norm in $\mathbb{R}^3$, i.e.

$$\|(w_1, w_2, w_3)\| = \sqrt{w_1^2 + w_2^2 + w_3^2}\,.$$

The back transform of $\boldsymbol{R} \in SO(3)$, given by the *inverse logarithm*, can be computed as:

$$\log(\boldsymbol{R}) = \begin{cases} \left(\frac{\phi}{2\sin\phi}(\boldsymbol{R} - \boldsymbol{R}^\top)\right)^\vee & \text{for } \mathrm{tr}(\boldsymbol{R}) \in (-1, 3) \\ (0,0,0) & \text{for } \mathrm{tr}(\boldsymbol{R}) = 3 \Leftrightarrow \boldsymbol{R} = \boldsymbol{I} \\ \frac{\pi}{2}\left(\mathrm{diag}(\boldsymbol{R}) + (1,1,1)\right) & \text{for } \mathrm{tr}(\boldsymbol{R}) = -1 \end{cases} \tag{2.39}$$

where $\mathrm{tr}(\boldsymbol{R})$ is the *trace* of the matrix $\boldsymbol{R}$, $diag(\boldsymbol{R})$ is a vector containing the diagonal entries of the matrix $\boldsymbol{R}$, and $\phi \in \mathbb{R}$ is a scalar with $\cos\phi = \frac{\mathrm{tr}(\boldsymbol{R})-1}{2}$ and $|\phi| < \pi$. Note that, since $\boldsymbol{R}$ is a rotation matrix, $\mathrm{tr}(\boldsymbol{R}) \in [-1, 3]$.

For elements $\boldsymbol{M} = \begin{pmatrix} \boldsymbol{R}_{3\times 3} & \boldsymbol{v}_{3\times 1} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix}$ of $SE(3)$ with $\boldsymbol{R} \in SO(3), \boldsymbol{v} \in \mathbb{R}^3$, the back-transform $\log : SE(3) \to se(3)$ is given by

$$\log(\boldsymbol{M}) = \begin{cases} \left(\left(-\hat{\boldsymbol{\omega}}\sin\|\boldsymbol{\omega}\| - \hat{\boldsymbol{\omega}}^2(1 - \cos\|\boldsymbol{\omega}\|)\hat{\boldsymbol{\omega}} + \boldsymbol{\omega}^\top\boldsymbol{\omega}\right)^{-1}\|\boldsymbol{\omega}\|^2\boldsymbol{v}, \boldsymbol{\omega}\right) & \text{for } \boldsymbol{\omega} \neq \boldsymbol{0} \\ (\boldsymbol{v}, \boldsymbol{\omega}) & \text{for } \boldsymbol{\omega} = \boldsymbol{0} \end{cases} \tag{2.40}$$

where $\boldsymbol{\omega} = \log(\boldsymbol{R})$ is computed using Equation (2.39).

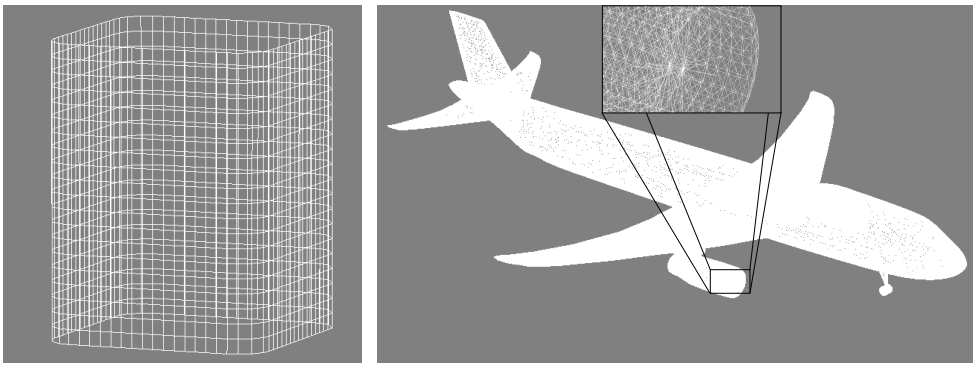
This concludes our introduction to Lie groups and Lie algebras. In the next section, we will show how to use these groups (and algebras) to store the 3-D pose and motion of rigid objects.

2.4. Object Models and Their Transformations

"Nothing is more revealing than movement."

Martha Graham (1894 – 1991)

In this section, we shortly explain how the object models we use for tracking are built up, and how they can be transformed using twists. This is first introduced for rigid objects. Afterwards, this concept is extended to kinematic chains.



(a) (b)

Figure 2.3.: Illustrations of a model that uses a patch representation (left) and of a model given as triangle mesh (right).

2.4.1. Rigid Models

This section deals with *rigid objects*, which are objects with an inflexible surface that cannot be deformed in any way. The complete information about the pose of such an object is given by its position and orientation in 3-D space. This can be stored in a single twist, since a twist corresponds to a rotation followed by a translation.

Note that the position and orientation of the model can always be changed by transforming each point of the model. Thus, the transformation needed to move a rigid object from its initial pose to any position is given by an element of the Lie group $SE(3)$ or a twist $\boldsymbol{\xi} \in se(3)$, respectively.

Hence, we start by explaining how to transform a single point $\boldsymbol{X} \in \mathbb{R}^3$ using a twist $\boldsymbol{\xi} \in se(3)$ to its new position $\boldsymbol{X}'$. For this transformation, we assume the point is given in *homogeneous coordinates*, i.e. $\boldsymbol{X} = (x, y, z, 1)^\top$. Then, it can be moved by first converting the twist to an element of the Lie Algebra $SO(3)$ (by computing $\exp(\hat{\boldsymbol{\xi}})$ using the Rodriguez formula (2.G)), followed by a simple matrix vector multiplication:

$$\boldsymbol{X}' = \exp(\hat{\boldsymbol{\xi}})\boldsymbol{X} \tag{2.41}$$

The transformed point is also given in homogeneous coordinates.

In order to use a rigid object, we first have to model it in some way. While algebraic expressions can be used for simple models, they are not suitable for complex ones. Thus, a general description by patches or triangle meshes will be used in this thesis.

In the *patch representation* (see Figure 2.3(a)), the model consists of some patches each of which consists of a 2-D array of points. Neighbouring points are connected such that strips of quadrangles are formed. If the first and last points in a row or column are identical, the strip is closed.

In the *triangle representation* (see Figure 2.3(b)), the model consists of a set of points and a set of triangles where every triangle is saved as a triple of points. In this representation, any object can be approximated arbitrarily well using a sufficiently large amount of triangles.

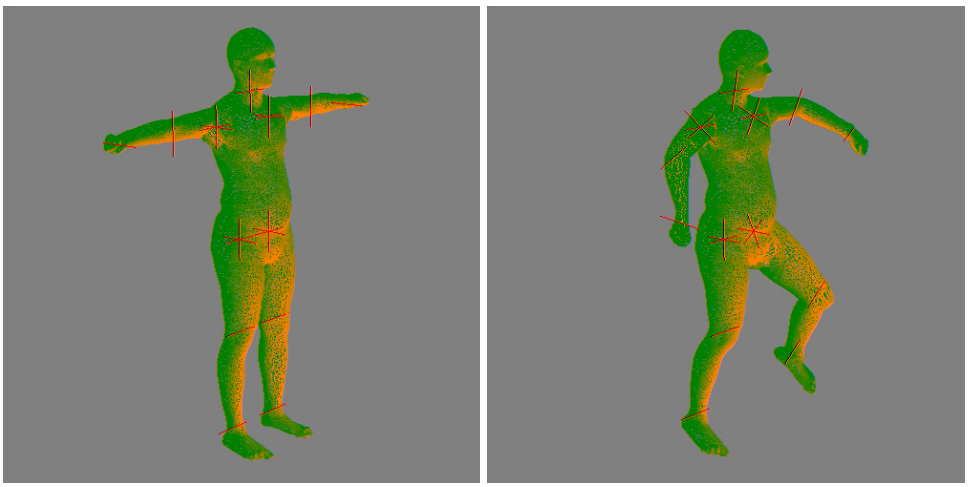


Figure 2.4.: Illustration of a model given as kinematic chain. In the left image, the object is in its initial pose while the right image illustrates the same model with different joint angles. The red lines illustrates the position and orientation of the joints. In these images, the triangles the model consists of have been lit such that the human is visible more clearly.

Other descriptions than those used in this thesis are also possible. For example, Dambreville *et al.*, who perform tracking by minimising an energy function nearly identical to one of the energy functions used in this thesis, uses point clouds in [58], amongst others.

For all representations, we will call a point on the surface of the 3-D model a *vertex*.

2.4.2. Kinematic Chains

The advantage of rigid objects is that six parameters are sufficient to describe its pose. Unfortunately, the object to be tracked is not always rigid. Thus, we will explain in this section the more general concept of kinematic chains, and give details how to model and transform them.

A *kinematic chain* is a system of $n + 1$ rigid bodies which are connected by n joints and represented in a tree structure, as explained in [29, 178]. Each joint is modelled as a twist with known rotation axis, but unknown rotation angle. Thus, there is one additional parameter that must be estimated for each joint in the model. We will denote the known rotation axis of the i-th joint by $\boldsymbol{\xi}_i$, the unknown joint angle by θ_i, and the vector containing all joint angles by $\boldsymbol{\Theta} := (\theta_1, \ldots, \theta_n)$. This vector is also called *joint configuration*.

In case of objects modelled with kinematic chains, the pose sought after in tracking no longer consists of a single twist $\boldsymbol{\xi} \in se(3)$, but of a vector $\boldsymbol{\chi} := (\boldsymbol{\xi}, \boldsymbol{\Theta}) = (\boldsymbol{\xi}, \theta_1, \ldots, \theta_n)$ containing the global rigid body motion and the joint configuration.

When using kinematic chains, each vertex is assigned to one of the rigid bodies. Consequently, a patch or triangle may belong to more than one rigid object in the kinematic chain, and the length of the line between points on different subobjects might be stretched or shrunk. A human modelled as kinematic chain is illustrated in Figure 2.4, which shows the human with two different joint configurations. Note that the joints have to be added manually in complex models created by first scanning the object using a 3-D scanner. Thus, there is often no clear

boundary between neighbouring rigid objects.

In order to transform a vertex $\boldsymbol{X}$ on a kinematic chain, one has to consider to which rigid part of the model the point belongs. If the point is on the rigid object at the root of the tree that describes the kinematic chain, it is transformed exactly as before (see Equation (2.41)). If the point lies on a rigid object which is connected to the root by the joints $\xi_{i_1}, \ldots, \xi_{i_j}$ with angles $\theta_{i_1}, \ldots \theta_{i,j}$, where the index i_1 denotes the first joint as seen from the root of the kinematic chain, the transformation is given by

$$\boldsymbol{X}' = \exp\left(\hat{\boldsymbol{\xi}}\right) \exp\left(\theta_{i_1}\hat{\boldsymbol{\xi}}_{i_1}\right) \exp\left(\theta_{i_2}\hat{\boldsymbol{\xi}}_{i_2}\right) \ldots \exp\left(\theta_{i_j}\hat{\boldsymbol{\xi}}_{i_j}\right) \boldsymbol{X}, \tag{2.42}$$

as the relative movement of each rigid part must be considered.

2.5. Representing Lines and Planes

"One geometry cannot be more true than another; it can only be more convenient. Geometry is not true, it is advantageous."

Robert M. Pirsig (*1928)

As we will show later, 3-D lines will be used in the estimation of the pose by our tracking approach. However, there are different ways how to represent lines in 3-D space. This section will introduce the representation used in our tracking approach, and also explains the related representation of planes necessary for the extension introduced in Section 5.4.3.

2.5.1. Plücker Lines

For the point-based pose estimation algorithm introduced later, 3-D lines will be used. There are different ways to represented 3-D lines, e.g. by two points on the line or by one point and a direction. Here, we will use the *Plücker form* [245], as it simplifies some computations.

A 3-D line in Plücker form $\boldsymbol{L}= (\boldsymbol{n}, \boldsymbol{m})$ is given by a normalised vector $\boldsymbol{n}\in \mathbb{R}^3$ (pointing in the direction of the line) and a *momentum* $\boldsymbol{m}\in \mathbb{R}^3$ that can be computed as $\boldsymbol{m} := \boldsymbol{y} \times \boldsymbol{n}$ for any point $\boldsymbol{y}$ on the line. Note that the momentum is independent of the choice of $\boldsymbol{y}$: If $\boldsymbol{y}$ and $\boldsymbol{y}'$ are two points on the line $\boldsymbol{L}$ (see Figure 2.5) with

$$\boldsymbol{y}_1 := \boldsymbol{y}' - \boldsymbol{y} = \lambda \boldsymbol{n} \tag{2.43}$$

for some $\lambda \in \mathbb{R}$, it follows that

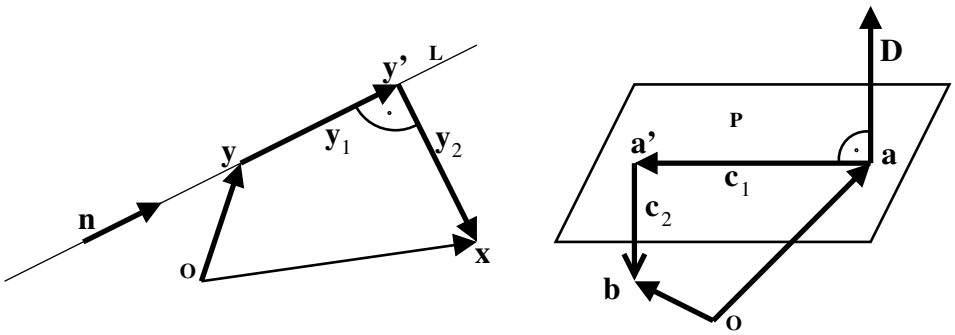


Figure 2.5.: **Left**:Identifiers used in the proofs that the momentum is independent of the point $\boldsymbol{y}$ chosen, and that the distance between a point $\boldsymbol{x}$ and a Plücker line $\boldsymbol{L} = (\boldsymbol{n}, \boldsymbol{m})$ is given by $\|\boldsymbol{x} \times \boldsymbol{n} - \boldsymbol{m}\|$. **Right**: Identifiers used in the corresponding proofs for planes.

$$\begin{aligned} \boldsymbol{y}' \times \boldsymbol{n} &= (\boldsymbol{y} + \boldsymbol{y}_1) \times \boldsymbol{n} \\ &= (\boldsymbol{y} + \lambda \boldsymbol{n}) \times \boldsymbol{n} \\ &= \boldsymbol{y} \times \boldsymbol{n} + \lambda \underbrace{\boldsymbol{n} \times \boldsymbol{n}}_{=0} \\ &= \boldsymbol{y} \times \boldsymbol{n} \, . \end{aligned} \tag{2.44}$$

Thus, the momentum is unique, i.e. well-defined. Since there are two possible normalised vectors $\boldsymbol{n}$ to a given line, there are always exactly two ways to represent a line in Plücker form.

One advantage of the Plücker form is that the distance of a point $\boldsymbol{x}$ to a line $\boldsymbol{L} = (\boldsymbol{n}, \boldsymbol{m})$ given in Plücker form can easily be computed as $\|\boldsymbol{x} \times \boldsymbol{n} - \boldsymbol{m}\|$. To understand this, we write $\boldsymbol{x}$ as sum of $\boldsymbol{y}$, the vector $\boldsymbol{y}_1 = \lambda \boldsymbol{n}$, which is parallel to $\boldsymbol{n}$, and a vector $\boldsymbol{y}_2$ perpendicular to $\boldsymbol{n}$, i.e. $\boldsymbol{x} = \boldsymbol{y} + \boldsymbol{y}_1 + \boldsymbol{y}_2$ (see Figure 2.5). Then, we have

$$\begin{aligned} \|\boldsymbol{x} \times \boldsymbol{n} - \boldsymbol{m}\| &= \|(\boldsymbol{y} + \lambda \boldsymbol{n} + \boldsymbol{y}_2) \times \boldsymbol{n} - \boldsymbol{m}\| \\ &= \| \underbrace{\boldsymbol{y} \times \boldsymbol{n}}_{=\boldsymbol{m}} + \lambda \underbrace{\boldsymbol{n} \times \boldsymbol{n}}_{=\boldsymbol{0}} + \boldsymbol{y}_2 \times \boldsymbol{n} - \boldsymbol{m}\| \\ &= \|\boldsymbol{y}_2 \times \boldsymbol{n}\| \\ &= \|\boldsymbol{y}_2\|. \end{aligned} \tag{2.45}$$

This equation also shows that $\boldsymbol{x} \in \boldsymbol{L}$ is equivalent to $\|\boldsymbol{x} \times \boldsymbol{n} - \boldsymbol{m}\| = 0$, i.e. a point $\boldsymbol{x}$ lies on the line $\boldsymbol{L} = (\boldsymbol{n}, \boldsymbol{m})$ if and only if $\boldsymbol{x} \times \boldsymbol{n} = \boldsymbol{m}$.

2.5.2. Hessian Form of a Plane

Similarly to lines, there are different ways how to encode a plane, e.g. by three points, or by a point and two (linearly independent) vectors in the plane.

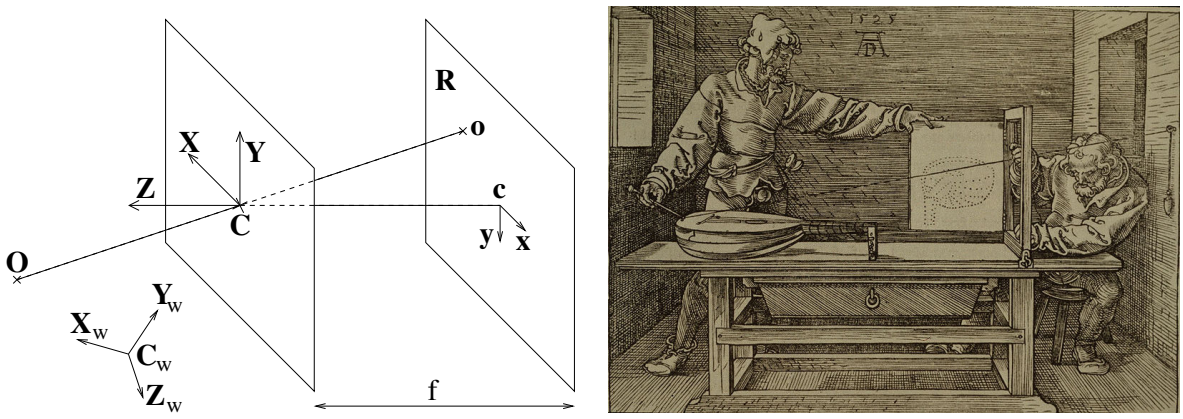


Figure 2.6.: **Left**: Schematic illustration of the pinhole camera model. **Right**: The image of the woodcut "Man drawing a Lute" from Albrecht Dürer, drawn in 1525, illustrates the idea behind the pinhole camera with a virtual image plane.

In this thesis, we use planes in the *Hessian form*. That is, a plane is implicitly given by the pair $(\boldsymbol{D}, d)$ consisting of a normalised vector $\boldsymbol{D} \in \mathbb{R}^3$, the normal of the plane, and the (signed) distance $d \in \mathbb{R}$ of the plane to the origin, which can be computed as $d = -\boldsymbol{a} \cdot \boldsymbol{D}$ for a point $\boldsymbol{a}$ on the plane.

It is easy to show that d is independent of the choice of $\boldsymbol{a}$: If $\boldsymbol{a}'$ is another point in the plane, i.e. the vector $\boldsymbol{c}_1 = \boldsymbol{a}_1 - \boldsymbol{a}$ is in the plane (see Figure 2.5), we have:

$$\begin{aligned} -\boldsymbol{a}' \cdot \boldsymbol{D} &= -(\boldsymbol{a} + \boldsymbol{c}_1) \cdot \boldsymbol{D} \\ &= -\boldsymbol{a} \cdot \boldsymbol{D} - \underbrace{\boldsymbol{c}_1 \cdot \boldsymbol{D}}_{=0} \\ &= -\boldsymbol{a} \cdot \boldsymbol{D} \end{aligned} \tag{2.46}$$

If the additional constraint $d \geq 0$ is added, we get the *Hessian normal form* of the plane. Although there are exactly two different normals, the requirement $d \geq 0$ makes the Hessian normal form unique.

To compute the distance of a point $\boldsymbol{b}$ to the plane $\boldsymbol{P} = (\boldsymbol{D}, d)$, the following formula can be used:

$$\text{dist}(\boldsymbol{b}, \boldsymbol{P}) = |\boldsymbol{b} \cdot \boldsymbol{D} + d|. \tag{2.47}$$

Especially, the point $\boldsymbol{b}$ is on the plane $(\boldsymbol{D}, d)$ if this distance is zero,i.e. $\boldsymbol{b} \in \boldsymbol{P} \Leftrightarrow \boldsymbol{b} \cdot \boldsymbol{D} + d = 0$.

Again, the proof is easy (see Figure 2.5):

$$\begin{aligned} \|\boldsymbol{b} \cdot \boldsymbol{D} + d\| &= \|(\boldsymbol{a} + \boldsymbol{c}_1 + \boldsymbol{c}_2) \cdot \boldsymbol{D} + d\| \\ &= \|\underbrace{\boldsymbol{a} \cdot \boldsymbol{D}}_{=-d} + \underbrace{\boldsymbol{c}_1 \cdot \boldsymbol{D}}_{=0} + \boldsymbol{c}_2 \cdot \boldsymbol{D} + d\| \end{aligned}$$

$$= \|\boldsymbol{c}_2 \cdot \boldsymbol{D}\| = \|\boldsymbol{c}_2\| \underbrace{\|\boldsymbol{D}\|}_{=1} |\underbrace{\cos(\theta)}_{=\pm 1}| \tag{2.48}$$
$$= \|\boldsymbol{c}_2\|$$

Here, θ is the angle between $\boldsymbol{c}_2$ and the normal vector $\boldsymbol{D}$ of the plane $\boldsymbol{P}$, which is zero or 180 degrees by construction of $\boldsymbol{a}'$. Thus, the sign of the term $\boldsymbol{x} \cdot \boldsymbol{D} + d$ determines the half-space in which the point lies: If it is positive, θ is zero, i.e. the point lies in the half-space determined by the direction $\boldsymbol{D}$.

2.6. Pinhole Camera Model

"Science is nothing but perception."

Plato ($\approx$ 428/427 – 348/347 BC)

In Section 2.4, we explained how to describe 3-D models as set of points with some underlying structure. In order to show these models in 2-D images, we have to project them in the same way a camera projects parts of the real world. There are different ways to model such a camera, ranging from a simple orthographic projection to models that incorporate different aberrations such as spherical aberrations or barrel/pincushion distortion.

In this thesis, we use the *pinhole camera model*, which is probably the most common camera model. It performs a perspective projection without any aberrations. The remainder of this section gives a short introduction about this camera model. More detailed introductions can be found in [78] or [107].

The basic idea behind the pinhole camera model is that each 3-D object point $\boldsymbol{O}$ is projected to an image position $\boldsymbol{o}$ computed as intersection point between the *image plane* (or *retinal plane*) $\boldsymbol{R}$ and the line through $\boldsymbol{O}$ and the so-called *optical centre* (also *focal point* or simply *camera centre*) $\boldsymbol{C}$, as illustrated in Figure 2.6. The line passing through $\boldsymbol{C}$ that is orthogonal to the image plane is called the *optical axis*. The intersection point between the optical axis and the image plane is denoted by $\boldsymbol{c}$. This point is called *principal point*, and the distance of $\boldsymbol{c}$ and $\boldsymbol{C}$, i.e. the distance between the point $\boldsymbol{C}$ and the plane $\boldsymbol{R}$ is called the *focal length* (or *focal distance*) $f > 0$.

One drawback of this camera model is that the orientation of the image is flipped. That is, an arrow pointing upwards in the real world will point downwards after projecting it. This problem can be "fixed" by using a virtual image plane in front of the optical centre, as shown in the right image in Figure 2.6.

When dealing with cameras, there are three different coordinate systems that must be considered (see Figure 2.6):

- A 2-D *image coordinate system* in the image plane with origin $\boldsymbol{c}$ and unit vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

- A 3-D *camera coordinate system* with origin $\boldsymbol{C}$ and axes $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ such that $\boldsymbol{Z}$ is parallel to the optical axis. The axes are orthogonal to each other and have length 1.
- A 3-D orthonormal *world coordinate system* $\boldsymbol{X}_W, \boldsymbol{Y}_W, \boldsymbol{Z}_W$ with origin $\boldsymbol{C}_W$.

To compute the image position x_i, y_i of a point given in world coordinates, we have to find a transformation from world coordinates into image coordinates.

Let us first consider how to get from a point $\boldsymbol{O} := (X_O, Y_O, Z_O)$ in camera coordinates to the image coordinates x_i, y_i. With the intercept theorem, we get

$$-\frac{f}{Z_O} = \frac{x_i}{X_O} = \frac{y_i}{Y_O} \quad \Leftrightarrow \quad x_i = -\frac{fX_O}{Z_O}, \quad y_i = -\frac{fY_O}{Z_O}. \tag{2.49}$$

Using homogeneous coordinates on both sides, this can be written as:

$$\begin{pmatrix} x_i \\ y_i \\ Z_i \end{pmatrix} = \begin{pmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X_O \\ Y_O \\ Z_O \\ 1 \end{pmatrix} \tag{2.50}$$

Remember that, since the vector (x_i, y_i, Z_i) is given in homogeneous coordinates, this results in the 2-D point $(\frac{x_i}{Z_i}, \frac{y_i}{Z_i})$

In order to transform the world coordinate system to the camera coordinate system, a rigid transformation must be done. That is, the necessary transformation is an element of the Lie group $SE(3)$ (see Definition 2.E). The necessary rotation $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$ are called the *extrinsic camera parameters*.

While the extrinsic camera parameters depend only on the position of the camera, there are also parameters that depend on the camera itself. These parameters are the position of the principal point $c= (x_0, y_0)$, the pixel dimensions k_x and k_y in x and y direction, and the angle θ between the coordinate axes x and y. To transform a point from the camera coordinate system to the image coordinate system, the following matrix can be used:

$$\begin{pmatrix} k_x & -k_x \cot(\theta) & x_0 \\ 0 & \frac{k_y}{\sin(\theta)} & y_0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.51}$$

Thus, the projection of a 3-D point $\boldsymbol{W} = (X_W, Y_W, Z_W, 1)^\top$ from world coordinates into image coordinates is given by

$$\begin{pmatrix} \frac{x_i}{Z_i} \\ \frac{y_i}{Z_i} \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ Z_i \end{pmatrix} = \begin{pmatrix} k_x & -k_x \cot(\theta) & x_0 \\ 0 & \frac{k_y}{\sin(\theta)} & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{R}_{3\times3} & \boldsymbol{t}_{3\times1} \\ \boldsymbol{0}_{1\times3} & 1 \end{pmatrix} \begin{pmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{pmatrix}. \tag{2.52}$$

Combining these three matrices into a single matrix results in the so-called 3×4 *projection matrix* $\boldsymbol{P}$. In total, there are 12 free parameters: Three for the global position $\boldsymbol{t}$, three for the

global rotation $\boldsymbol{R}$, one for the focal length f, two for the principal point $c = (x_0, y_0)$, two for the pixel dimensions k_x, k_y, and one for the angle between the coordinate axes θ. Since scaling the complete matrix results in the same projection function, there are eleven parameters that must be found to estimate an unknown projection matrix $\boldsymbol{P}$. The process of estimating the camera parameters is called *camera calibration*. One distinguishes between *explicit camera calibration*, which is the process of finding the individual extrinsic and intrinsic camera parameters, and *implicit camera calibration*, which is the (simpler) task to find the complete projection matrix P. In this thesis, only the complete projection matrix is used. Thus, implicit camera calibration is sufficient.

For implicit camera calibration, we create a set of 2-D–3-D point correspondences, i.e. we specify the 3-D positions of some 2-D image points. This can be done by taking an image of a calibration cube, as distances on the cube are known. Two such cubes used for camera calibration in sequences tracked in later chapters are shown in Figure 2.7. From these correspondences, a system of equations is set up and solved to obtain the projection matrix. However, this is only one of several possible approaches to camera calibration. In [8], for example, a set of ellipses is used for camera calibration.

The inverse task of determining the 3-D position $\boldsymbol{O}$ of a point from a given 2-D image point $\boldsymbol{o}$ is obviously underdetermined, since there are infinitely many points that are projected to each image point. The set of all such points is given by the *projection ray*, which is a line passing through the 2-D image point $\boldsymbol{o}$ and the camera centre $\boldsymbol{C}$.

2.7. Statistics

"There are three kinds of lies: lies, damned lies, and statistics."

Benjamin Disraeli (1804 – 1881)

In order to track an object, it is necessary to decide whether a point in the image belongs to the object to be tracked or to the background. This is often done by estimating the appearance of the fore- and background in some way, followed by a comparison to see to which appearance a point fits better. As in many other works [219, 58], we will use probability density functions to model the appearance of different image parts.

In this section, we introduce the basic concepts behind probability density functions, and explain some methods how they can be estimated. More detailed introductions into probability theory can be found in [130, 145, 257].

2.7.1. Intuitive Introduction to Probability Density Functions

For the proper definition of random variables and their probability density functions, some rather abstract concepts ranging from σ-algebras over measurable spaces up to probability spaces have to be considered. Such an axiomatic approach is not really helpful for beginners to understand the intuition behind the concepts used. Thus, we will give an intuitive and very

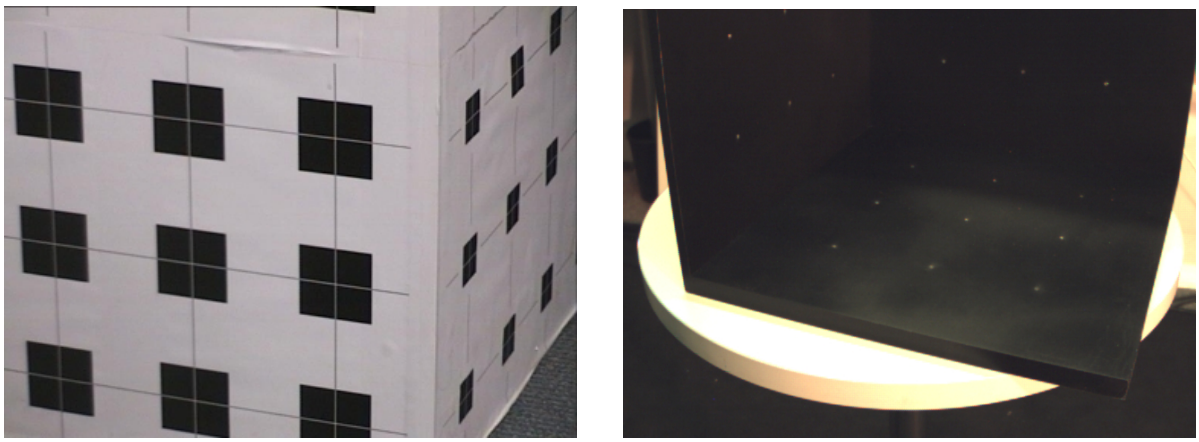


Figure 2.7.: Two cubes used for camera calibration in some sequences shown in this thesis.

short introduction in this section. For those readers interested in an axiomatic introduction, we refer to Appendix A, or to one of the books mentioned above.

To understand the idea behind probability theory, visualise a *random variable* as numerical result of an non-deterministic process or experiment. Simple (discrete) examples include the results from rolling a die, drawing lottery numbers, or flipping a coin. Note that, in the last example, we have to assign numbers to each outcome, e.g. by identifying "head" with zero and "tail" with one, since we only consider random variables as mappings into the real numbers in this intuitive introduction.

Well-known continuous examples of random variables are the time until a certain event occurs, or the *canonical random variable*, which results in a random number in the half-open interval $[0, 1)$ such that each number is equally likely.

The function P which assigns to each set the likelihood that the result of one experiment is in the set is called the *probability distribution*. When tossing a normal die, for example, we have $P(\{3\}) = \frac{1}{6}$, and $P(\{1, 4, 5\}) = \frac{1}{2} = 50\%$.

For tracking, we are especially interested in so called (absolutely) continuous random variables with values in $\mathbb{R}$. For such random variables X, there exist (by definition) an integrable function $p : \mathbb{R} \to \mathbb{R}^+$ such that for any set $S \subset \mathbb{R}$ over which one can integrate p, the property

$$P(X \in S) = \int_S p(x)dx \tag{2.53}$$

holds. The function p is called the *probability density function*. We will use the common abbreviation *PDF* in this thesis.

It is obvious that $P(\{x\}) = 0$ holds for all continuous random variables and all possible values of x, since an integral over a single point is always zero. Furthermore, we have $\int_{-\infty}^{\infty} p(x)dx = 1$, as $P(X \in \mathbb{R})$ must be equal to one. In fact, any nonnegative, integrable function f with $\int_{-\infty}^{\infty} f(x)dx = 1$ is a probability density function for some random variable.

The last concept we need is the (in)dependence of two (or more) random variables. As the name already implies, random variables are called *independent* if the results of the experiments measured by the random variables do not depend on each other. For example, when rolling

a die twice and noting the individual numbers, the two numbers are independent. However, the two random variables that measure "the first number" and "the sum of both numbers" are obviously dependent, i.e. not independent. If several random variables have the same underlying PDF and are independent, they are called *independent and identically distributed* (or short *i.i.d.*). Such random variables can be used to estimate the underlying PDF. This is the topic of the following section.

2.7.2. Estimating PDFs

To separate fore- and background points, we will model the appearance of those regions by PDFs and see which image points fit better to which PDF. However, the tracking approach presented in this thesis does not use predefined appearances of object or background. Thus, the required PDFs are unknown, and must be estimated.

In this section, we introduce some of the methods used to estimate the necessary PDFs, namely a simple *Parzen density estimation* [197] (also called *Parzen-Rosenblatt estimator* or *Rosenblatt-Parzen estimator* due to the initial work by Rosenblatt [215]), a global Gaussian model, and the local Gaussian model from [219].

Parzen Window Method

The *Parzen window method* or *kernel density estimation* is a non-parametric method to estimate the PDF of a sequence of i.i.d. random variables. Note that, in this context, *non-parametric* means that no prior knowledge about the PDF to be estimated is used, and not that the Parzen window method has no parameters.

Let $f(x, y) : \Omega \to \mathbb{R}$ be a grey-scale image from the image domain Ω to the real numbers and let $\Omega_1 \subset \Omega$ be a part of the image for which a PDF should be estimated. Since the Parzen density estimator does only consider the brightness of the pixels and not their positions, the information in the region Ω_1 is completely given by a sequence consisting of all brightness values in Ω_1. Let us denote this sequence by $x_1, \ldots, x_n$. Note that the ordering within the sequence is not important.

Intuitively, a kernel density estimation places some function around every observation of the random variable as shown in Figure 2.8: The red crosses indicate the observations $x_1, \ldots, x_n$, the blue lines the functions placed around each observation, and the green function is the estimated PDF obtained by adding the blue functions. As function, shifted and scaled versions of the same *kernel* must be used:

Definition 2.H: (Kernel) A *kernel* is a function $K\colon \mathbb{R} \to \mathbb{R}$ with

- $K(x) \geq 0$ for all $x \in \mathbb{R}$
- K is integrable
- $\int_{-\infty}^{\infty} K(x)\mathrm{d}x = 1$
- K is symmetric around zero, i.e. $K(x) = K(-x)$ for all $x \in \mathbb{R}$

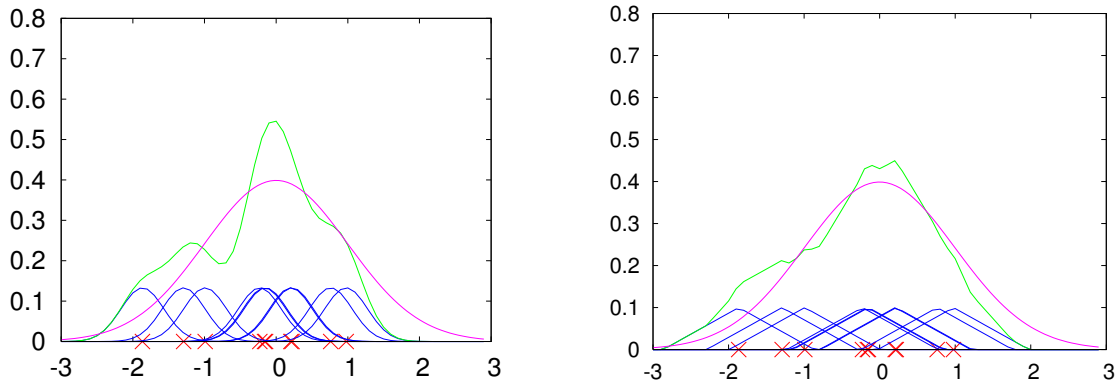


Figure 2.8.: Illustration of a Parzen estimation for a PDF with a Gaussian (left) and a triangle (right). The red crosses indicate the random events drawn from the PDF shown in violet and the blue curves the contribution of each point to the complete estimation (green).

In the examples in Figure 2.8, a Gaussian and a triangle were chosen as kernels. The resulting estimations p_h are given by

$$p_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) = \frac{1}{n} \sum_{i=1}^{n} K_h\left(x - x_i\right) \tag{2.54}$$

where

$$K_h(x) := \frac{1}{h} K\left(\frac{x}{h}\right), \tag{2.55}$$

and where the parameter h is the so-called *bandwidth* that serves as a smoothing parameter. The choice of h is very important, and wrong choices lead to estimation that are too smooth, or too noisy, respectively.

Global Gaussian Model

As mentioned above, a kernel density estimator does not use any prior knowledge about the PDF to be estimated. If prior knowledge is available, the estimator should use it, though. Thus, a kernel density estimator is obviously not optimal in such a situation.

One common way to use prior knowledge is to assume that the PDF to be estimated is a member of a parameterised class of PDFs. For example, the (global) *Gaussian model* explained in this section assumes that the PDF to be estimated is a Gaussian with unknown mean μ and standard deviation σ. That is, the PDF p is given by

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.56}$$

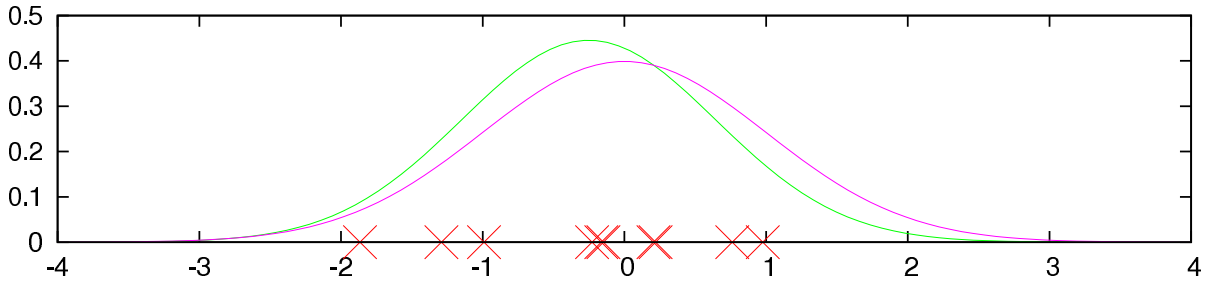


Figure 2.9.: Estimation using a Gaussian model: The observations are shown in red, the estimated PDF in green, and the actual PDF used to generate the random events in violet.

Consequently, instead of estimating an infinite-dimensional PDF, only two parameters must be estimated in this setup.

A common (but not optimal) way to estimate approximations $\tilde{\mu}$, $\tilde{\sigma}$ of μ, σ from the sequence $x_1, \ldots, x_n$ (as defined in the last section) is by using the following formulas [145]:

$$\tilde{\mu} = \frac{\int_{\Omega_1} f(x) \mathrm{d}x}{\int_{\Omega_1} 1 \, \mathrm{d}x}, \qquad \tilde{\sigma} = \sqrt{\frac{\int_{\Omega_1} (f(x) - \tilde{\mu})^2 \mathrm{d}x}{\int_{\Omega_1} 1 \, \mathrm{d}x}} \tag{2.57}$$

in a continuous setting and

$$\tilde{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i, \qquad \tilde{\sigma} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \tilde{\mu})^2} \tag{2.58}$$

in the discrete setting, respectively.

Figure 2.9 shows the estimated PDF when using the (global) Gaussian model and the same random variables as in Figure 2.8.

Local Gaussian Model

Both the Parzen window method as well the global Gaussian model ignore the ordering of the random variables since they are assumed to have the same distribution. However, in the situation described in the beginning of Section 2.7.2, i.e. if the observations are image points, this assumption usually does not hold.

Thus, it was proposed by [136] not to use a single PDF for the whole region Ω_1, but to use PDFs that vary depending on the position inside the region. That is, a PDF for each image pixel inside Ω_1 is estimated in such an approach. This is advantageous if the object's appearance varies strongly, e.g. due to shadows and highlights, or if the appearance of the object observable in the region Ω_1 is complex.

While it is possible to use the pixels within a certain neighbourhood to perform a kernel density estimation [136], the estimations can become inaccurate due to the low amount of information given. The advantage of a non-parametric method, namely their ability to model

multimodal densities, is usually not necessary within a local window, because multimodal situations are rarely encountered there.

Thus, it was proposed in [219] to assume local Gaussian probability densities

$$p(s,x) = \frac{1}{\sigma(x)\sqrt{2\pi}} \exp\left(\frac{(s-\tilde{\mu}(x))^2}{2\tilde{\sigma}(x)^2}\right) \tag{2.59}$$

that vary with the position x in the region. Here, $\tilde{\mu}(x)$ and $\tilde{\sigma}(x)$ are estimated in a local neighbourhood of each point as

$$\tilde{\mu}(x) = \frac{\int_{\Omega_1} K_\rho(t-x) f(t) \mathrm{d}t}{\int_{\Omega_1} K_\rho(t-x)\ \mathrm{d}t}, \qquad \tilde{\sigma}(x) = \sqrt{\frac{\int_{\Omega_1} K_\rho(t-x)(f(t)-\tilde{\mu}(x))^2 \mathrm{d}t}{\int_{\Omega_1} K_\rho(t-x)\ \mathrm{d}t}}, \tag{2.60}$$

where ρ is the standard deviation of the Gaussian used as window function. This *local Gaussian model* is widely used in segmentation nowadays [179, 151], and will also be employed for some experiments in this thesis.

3 Image Compression

"People forget how fast you did a job – but they remember how well you did it."

Howard Newton (*1982)

For many years, *image compression* has been a field of ongoing research due to the constantly increasing amount and resolution of images. Since lossy image compression algorithms can reach much higher compression ratios than lossless algorithms, we decided to develop a lossy image compression algorithm. This algorithm, which is based upon the work presented in [232] and [240], is the topic of this chapter.

One of the most prominent and widely used lossy image compression algorithms today is JPEG [200], which is based on the *discrete cosine transform* (*DCT*). Its successor JPEG 2000 [270], which uses Cohen-Daubechies-Feauveau wavelets, performs noticeably better but is not yet widely supported. Still, JPEG 2000 is likely to replace JPEG in the near future.

In contrast to these popular approaches, which are transformation-based, there are also recent approaches that store only a subset of all image points, and build upon *partial differential equations* (*PDE*s) to reconstruct the remainder of the image, as explained in Section 2.1.

The general problem to reconstruct missing image parts is called *inpainting*, or *disocclusion* [170, 21]. Note that upsampling, the task to increase the resolution of images, can also be regarded as inpainting, such that the same approaches can be used [290, 227].

Only a few selected points of the initial image are stored in the extreme situation considered in [87]. The remainder of the image is then reconstructed using edge-enhancing anisotropic diffusion (EED, see Section 2.1.4). These points are located on an adaptive triangulation, which is stored as binary tree [65]. However, even with the improvements introduced in [88], the performance of this compression approach only lies between that of JPEG and JPEG 2000 for medium to high compression ratios, and can only surpass JPEG 2000 at very high compression ratios.

The approach presented in this chapter builds upon the basic ideas presented in [88]. By changing a number of concepts and improving some parts of the algorithm [240], our novel

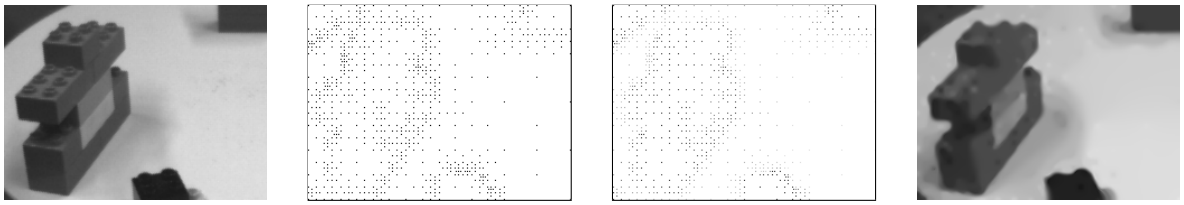


Figure 3.1.: Illustration of PDE-based image compression. **From left to right**: Initial image, inpainting mask, brightness values at the inpainting mask, and result after reconstruction with EED. In this example, a compression ratio of $95:1$ was used.

image compression codec yields much better results, which even surpass those of JPEG 2000 for basically all compression ratios in images with little texture.

It should be noted that there are a number of related approaches which also store only a small subset of the initial image and reconstruct the remainder. These methods differ both in the way the points are placed (regular, on specific structures, or arbitrarily) and in the way in which the image is reconstructed.

In [59], for example, points can be placed arbitrarily. When loading the image, the Delauney triangulation of the known points is computed, and the remainder is reconstructed using linear splines in each triangle. This approach was extended to anisotropic geodesic triangulations in [27].

While these approaches use triangulations to approximate the image, the image is approximated on rectangles using linear functions in [258]. The necessary recursive plane decomposition is embedded in a quadtree data structure. Another approach that uses quadtrees is given in [263], in which Lagrange multipliers are used to find optimal rate allocations.

Approaches in which images are reconstructed using information about image edges are also very common: In [117], different methods to reconstruct images from zero-crossings in scale-space are compared. Acar and Gökmen proposed to use an energy function based on a weak membrane model [3]. In [13], an edge-preserving filtering followed by a nonlinear difference filter is applied to obtain a ternary image which encodes edges in the smoothed image. Together with a subsampled version of the smoothed image, this information is used to reconstruct an approximation of the original image. In [62], an edge map and colour intensities of pixels left and right of each edge are stored. To reduce errors obtained by non-closed contours, the mean of each 10×10 block of the original image is additionally saved. In [73], blur and gradient direction is used as additional information to edge locations and intensity change. Recent work shows that approaches based on edges can even beat JPEG 2000 for cartoon-like images [168].

Another class of image and video compression algorithms also uses PDEs and related variational methods. In contrast to the works by Galić *et al.* [87, 88] and our approach [240], which use PDEs within the compression framework, these works mainly perform preprocessing [45, 144, 278] or postprocessing [6, 70, 98, 182, 298, 299] steps using PDEs. An exception is the work of Chan and Zhou [46] in which a variational approach using total variation is employed to minimise oscillations when using wavelet decompositions.

Furthermore, there has been work in which images are reconstructed using top points in scale-space [135, 138], i.e. from points in scale-space where both gradient and Hessian determinant vanishes. In [156], a set of scale-space blobs and edges is used to reconstruct images, where a first order structure is used for edges, and a second-order structure for blobs. Another classical technical report using points and their derivatives for global image reconstructions is given by [39], in which it is proposed to use a standard Tichonov regularisation method and directional filtering for stabilisation.

This chapter is organised as follows: In the next section, we explain our image compression codec, which includes a detailed description of our compression and decompression algorithms as well as an explanation what information is stored in a compressed image. Section 3.2 shows several experiments that demonstrate the high quality of our compression algorithm. The chapter is concluded in Section 3.3 with a summary.

3.1. Codec

"One's first step in wisdom is to question everything - and one's last is to come to terms with everything. "

Georg C. Lichtenberg (1742 – 1799)

In this section, we give a detailed description of our image compression framework. We start by discussing how to choose the pixels whose brightness will be stored in the compressed file, and how their location is encoded. Next, we investigate how to store the corresponding brightness values, followed by several possible optimisations. After explaining a simple extension to colour images and clarifying the file format used, we also show how to decompress a file compressed with our algorithm.

3.1.1. Selecting Pixels and Encoding Their Localisation

Our image compression algorithm stores only a small subset of all points of the image, whereby the number of points saved directly corresponds to the compression ratio. The remainder of the image is then reconstructed using PDE-based inpainting, as explained in Section 2.1. Since the reconstruction quality depends on the choice of points, it is necessary to find a good subset of points to save. However, there is little theoretical knowledge about how to find such a subset. For linear isotropic diffusion (see Section 2.1.1) in a continuous setting, this question has been answered using the theory of shape optimisation [20]. Still, it is not clear how to carry over these results to the discrete setting, especially if there are additional model assumptions. Additionally, it may pay off to use more but less optimal points if those points can be encoded more efficiently.

A naive approach to find good point positions is to try all subsets of the whole image with a given number of points, and to choose the set with yields the best approximation to the original image. Even for a very small image of size 64×64 from which only 64 points are to be kept, this results in

$$\binom{4096}{64} \approx 7.5 \cdot 10^{141} \tag{3.1}$$

different subsets that must be checked. However, even when regarding the whole universe as a big physical system, the total number of elementary (quantum) logical operations the universe can have performed since the big-bang is merely around 10^{120} [157]. Moreover, saving the pixel locations tends to be quite expensive in this general setting since there usually is no simple pattern behind the points.

Another extreme approach is to store points on a regular grid. Encoding the point locations with such an approach only requires very few bits. However, unnecessarily many points are stored in homogeneous image regions, while regions with a more complicated appearance are sampled too sparsely. Thus, the reconstruction quality is often poor.

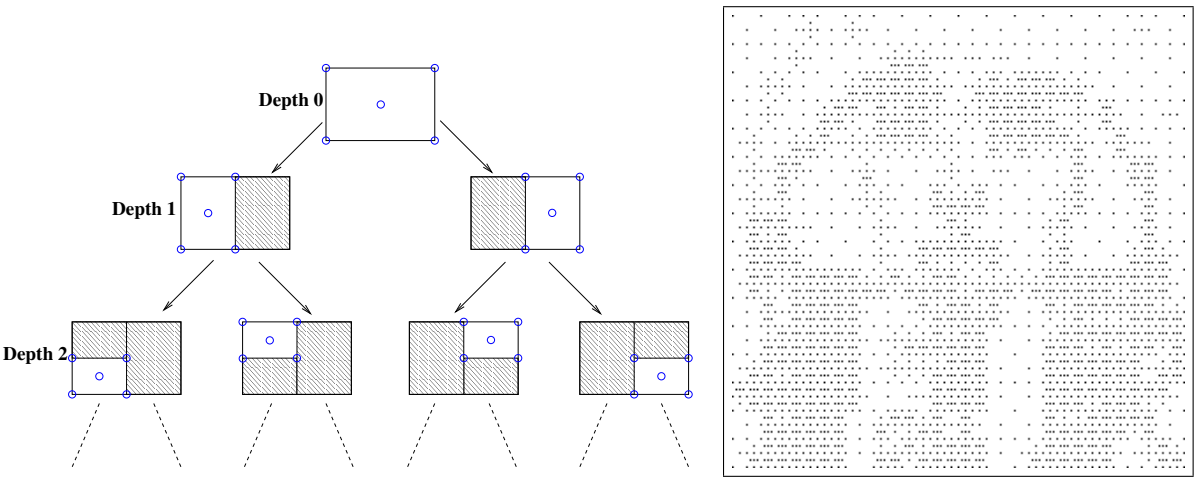


Figure 3.2.: **(a) Left**: Illustration of the binary tree structure used to find point positions restricted to an adaptive rectangular grid. The white parts are the (sub)images being processed, while the blue circles show example points that might be used to represent this part of the image. **(b) Right**: Example of a complete interpolation mask used for the image "trui". The original image is shown in Figure 3.6(a).

Here, we use a hybrid approach that only considers point positions on an adaptive rectangular grid. This way, all point locations can be encoded in a binary tree, similar to the tree used in [65]. This allows to store the point locations in an inexpensive way.

To find out which pixels should be kept, we simulate the decompression step by reconstructing the sub-image from those points that have already been chosen. We will give details which points these are later in this section. Then, original and reconstruction are compared. If the reconstruction is sufficiently close to the original, we stop. Otherwise, the image is split along the middle of the x or y direction. Thereby, we always split along the largest dimension. As illustrated in Figure 3.2, these two image parts are then saved recursively.

To judge how accurate the image is reconstructed, we use the *mean square error* (or *MSE*) between reconstruction and original image. The MSE is a widely-used error measure defined as

$$\mathrm{MSE}(f, g) := \frac{1}{|\Omega|} \sum_{i=1}^{|\Omega|} (f(i) - g(i))^2, \tag{3.2}$$

where Ω is the common image domain of the images f and g, and where $|\Omega|$ is its size. The same error measure will be used later in the experiments to compare different compression algorithms.

If the MSE between reconstruction and original image part exceeds the threshold T, the image is split by our algorithm. As in [88], this threshold depends on the recursion depth d of the sub-image being processed. More precisely, the threshold is given by $T := al^d$, where a and l are free parameters in our compression algorithm. We discuss how to find these parameters in Section 3.1.4.

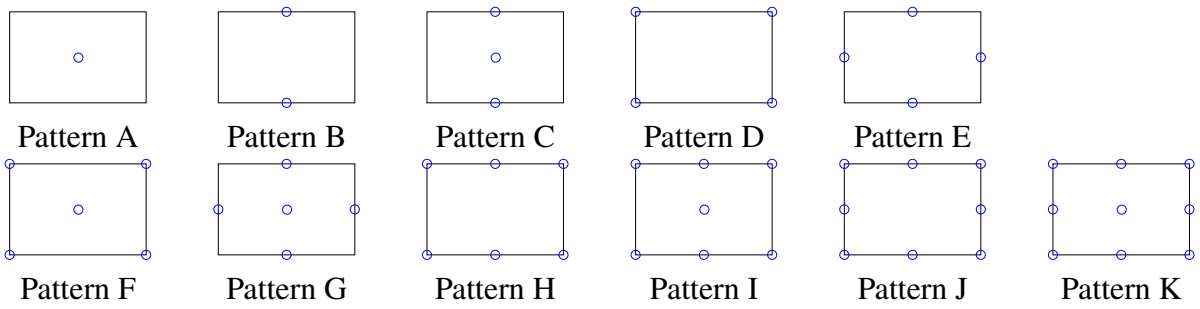


Figure 3.3.: The pixel selection patterns evaluated in our image compression framework. In pattern B, C, H, and I, the selected non-corner pixels are always on the longer side of the rectangle.

Table 3.1.: Number of points in each of the patterns shown in Figure 3.3, and mean square error (MSE) for each pattern when saving different images with a compression rate of $20:1$. Note that the brightness values are not compressed, yet. The best result in each line is highlighted.

Pattern	A	B	C	D	E	F	G	H	I	J	K
No. of points	1	2	3	4	4	5	5	6	7	8	9
MSE (trui)	47.75	46.49	43.98	38.08	47.53	**35.78**	46.94	37.60	39.68	44.09	41.86
MSE (walter)	23.61	25.13	22.51	20.64	23.27	**19.95**	24.58	20.27	22.30	22.44	20.90
MSE (peppers)	56.50	53.29	54.50	**44.13**	56.30	45.88	59.05	46.22	50.83	52.70	50.36
Average MSE	42.62	41.64	40.33	34.28	42.37	**33.87**	43.52	34.70	37.60	39.72	37.70

Once the splitting steps are completed, we know which points will be saved. Instead of storing the position of each pixel individually, it is sufficient to store a binary tree containing the splitting decisions. This is done by first storing the minimal depth up to which all sub-images are split, as well as the maximal depth at which no sub-image is split any more. In between, one bit for each node of the tree is saved. The same approach was used in [65] and [88] to store a triangular subdivision. In the decompression step, the positions of the saved points can be reconstructed from this tree. The points form the inpainting mask K used for reconstructing the image.

The amount of data necessary to store the tree described in the last paragraph depends on the maximal and minimal depth of the tree. Thus, it usually pays off to specify a minimal depth m up to which all images are split independent of T, and a maximal depth M at which no image is split no matter what T is. This way, more points can be saved without additional space, as less space is required to store the tree. These two parameters, which are not needed for decompression, can therefore improve the quality of a compressed image.

The last open question is which points of each sub-image should be stored. To answer this question, we have evaluated the performance of our compression algorithm when compressing the three images “trui”, “walter”, and “peppers” with a compression ratio of $20:1$. The eleven

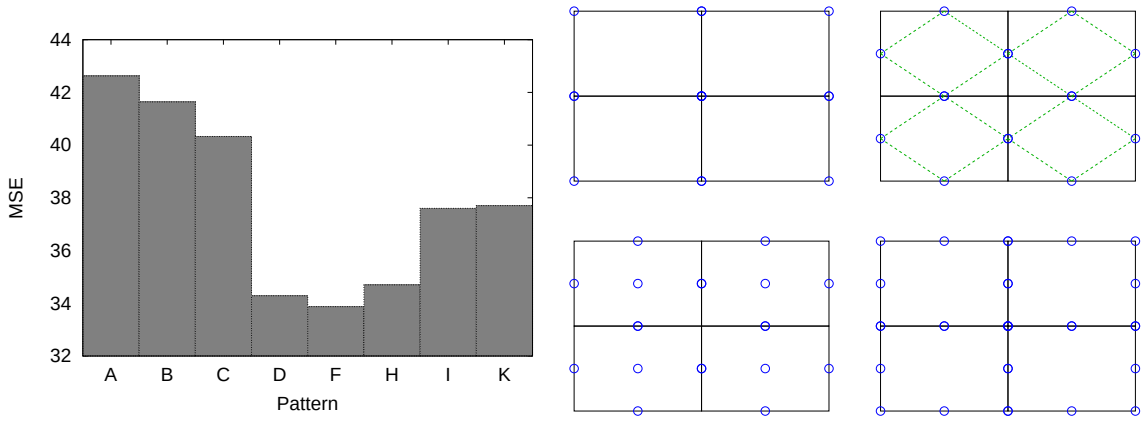


Figure 3.4.: **Left**: Average MSE obtained when compressing the three images "trui", "walter", and "peppers" with a compression ratio of $20 : 1$ with different point selection patterns. The patterns are shown in Figure 3.3, and are ordered with respect to the number of points used in each image part. Only a subset of all tested patterns is shown here. **Right**: Illustration of the resulting pattern if Pattern D (top left), E (top right), G (bottom left), or J (bottom right) is repeated several times.

patterns tested are illustrated in Figure 3.3, while the best MSE obtained with each pattern is shown in Table 3.1. This table also shows the average performance. As can be seen, the best results are usually obtained when using Pattern F. The results with Patterns D and H are also quite good, though. In the remainder of this section, we will explain why this is the case.

First of all, let use compare the results of the eight Patterns A, B, C, D, F, H, I, and K, i.e. we will ignore the Patterns E, G, and J for the moment. The MSEs obtained with these patterns are plotted in the graph in Figure 3.4. Note that the patterns are ordered according to the number of points in each pattern, as shown in Table 3.1. One can observe that there is an optimal number of points in each sub-image, namely five. This is due to the fact that the number of points in the pattern acts as a tradeoff between accurate point localisation and overhead necessary to store the binary tree. If there are many points in each sub-image, the overhead necessary to store a certain amount of points is quite low. However, the possible point positions are rather restricted, which results in a comparatively bad approximation quality. This can be seen at the right end of the graph. If there are few points in each image part, the drawback that a larger tree is necessary to store a certain amount of points outweighs the possibility to localise the points in a more precise way. This is visible in the left part of the graph.

Next, we compare the Patterns D and E. With both, the same amount of points is used in each image part. When several image parts with these patterns are next to each other (see right images in Figure 3.4), both form a regular grid. However, the two grid directions obtained with Pattern E (illustrated by the dashed green lines) are not orthogonal to each other in general. Furthermore, points on adjacent images are more likely to coincide, resulting in less points. Additionally, natural images often contain horizontal or vertical edges, which can be stored better using Pattern D.

Finally, we explain why Pattern G and J yield worse results than one would expect by only considering the number of points in the pattern. This is due to the fact that the interpolation points should be well distributed over the complete image domain. As shown in Figure 3.4, this is not the case for Pattern G and J, since there are "holes" in the otherwise regular patterns. Note that these holes also appear when using Pattern I, thus explaining the large jump between Patterns H and I, and the small difference between Patterns I and K. Similarly, the jump between Patterns C and D can be explained by the fact that the three points in Pattern C are very badly distributed, while the four points in Pattern D are very well distributed in the image parts considered.

Since Pattern F yields the best average performance, we use this pattern for the remainder of this thesis. An alternative is to use different patterns for different images. However, we have not done this in this thesis.

3.1.2. Quantisation and Encoding of Brightness Data

In the last section, we have shown how we encode the positions of the pixels stored in the compressed file. Next, we explain how to store the corresponding brightness values.

First of all, a list of all brightness values to be stored is created. This is done by scanning the interpolation mask row-wise from top left to bottom right. Whenever the interpolation mask K indicates that a pixel should be kept, the brightness value from the original image is added to the list.

In contrast to the pixel positions, which are stored in a lossless way, a lossy compression will be used for the brightness values. That is, quantised brightness values are stored. This reduces the number of possible brightness values, which improves the final entropy coding step explained later this section. Thus, the coarser the quantisation, the more pixels can be saved. It is easy to see that, for a fixed amount of space, there is a tradeoff between a fine quantisation, which allows to store less points very accurately, and a coarse quantisation, with which points can be stored with less accuracy.

Here, we use an (approximately) equidistant quantisation. Each of the 256 grey values possibly present in the original image is mapped to the closest of the q quantised values

$$q_i := \left\lfloor \frac{(i + \frac{1}{2}) \cdot 256}{q} \right\rfloor \qquad i \in \{0, \ldots, q-1\}, \tag{3.3}$$

where $\lfloor x \rfloor : \mathbb{R} \mapsto \mathbb{N}$ returns the largest integer smaller or equal to x. Note that q can have any value between 2 and 256. Further note that we store the index i instead of the quantised value q_i, as this results in smaller files with some of the entropy coders described in the next paragraph.

As a final step, the quantised brightness values are encoded using a general purpose entropy coder. Different entropy coders can be used in this step. Specifically, our implementation supports *Huffman coding* [116] with or without using *canonical codes*, *arithmetic coding* with static or adaptive model [214], *PAQ* (version paq8o8z-feb28, [165, 166]), *gzip* (version 1.3.5, [86]), *bzip2* (version 1.0.3, [243]), *Lempel-Ziv-Welch coding* [291], and *range coding* [169]. For gzip and bzip2, we used the standard implementations available in Linux. We built our

own version of PAQ from the source code available at [165] by removing its ability to save multiple files in one archive, and by removing the name of the stored file from the archive. For the remaining entropy coders, the implementations from [64] have been used.

In order to decide which entropy coder is suited best for quantised brightness values, we evaluate the performance of these coders using different quantisation levels and point sets with different sizes. Some of these experiments are shown in Table 3.2. It can be seen that different coders are optimal in different situations: For larger files, PAQ always performs best, independent of the quantisation used. For small files, arithmetic coding and gzip sometimes results in a stronger compression. In our codec, we use three additional bits in the uncompressed image header to decide which of the first eight methods shown in Table 3.2 was used to store the brightness data. This way, the entropy coder which yields the best compression for a specific file can be used.

3.1.3. Improved Diffusion Parameters

Some of the inpainting operators explained in Section 2.1 have free parameters, namely the (regularised) Charbonnier operator, and EED (see Sections 2.1.3 and 2.1.4).

One important parameter present in these nonlinear operators is the contrast parameter λ, which is used to distinguish between low contrast regions where homogeneous diffusion is approximated, and high contrast locations where less diffusion is appropriate.

In contrast to the approach from [88], in which a constant λ was used to create the inpainting mask K, we employ different values for λ for different images or compression ratios. That is, the (quantised) contrast parameter which yields the smallest reconstruction error is stored in the file header, and used when decompressing the image. A single byte is sufficient to store this parameter according to our experiments, since the optimal value has a limited range. This range, which depends on whether Charbonnier interpolation, regularised Charbonnier interpolation or EED interpolation is used, is quantised linearly into 256 values. For example, we used the interval $[0, 1]$ for EED. To find the optimal λ, we decompress the image once with each of the 256 possible values for λ. However, it is also possible to use a binary search without noticeably worse results.

The second parameter that can be optimised is the regularisation parameter σ, which occurs in the regularised Charbonnier operator and in EED. In our experiments, even strong variations only slightly change the results with our framework. We conclude that, possibly due to the specific point patterns or point densities used, the presence of regularisation itself is more important than its exact amount in the setting considered here. Thus, we use a constant σ of 0.8 for all experiments shown here.

3.1.4. Finding Good Parameters

For our compression framework, we have followed an idea often used in video compression. This idea is to specify only file format and decoder, while the encoder is not specified explicitly. This allows to improve the encoder without changing the decoder or the file format. For example, more than 6 megabits per second were necessary in 1995 to get a picture quality acceptable for television broadcast when using MPEG-2. In 2002, a third of that bandwidth

Table 3.2.: Size of brightness data after compression using different general purpose entropy coders. As initial data, brightness values obtained when encoding the image "trui" with different splitting thresholds are used. Shown are the size of the uncompressed pixel data in bytes (none), and the size of the compressed pixel data with *Huffman coding* (HC), with *Huffman coding* using canonical codes (HCc), with *arithmetic coding* with static (ACs) or adaptive (ACa) model, *PAQ*, *gzip* (version 1.3.5), *bzip2* (version 1.0.3), *Lempel-Ziv-Welch coding* (LZW), and *range coding* (RC). The best result for each ratio and quantisation level is highlighted.

Levels	Size									
	None	HC	HCc	ACs	ACa	PAQ	gzip	bzip2	LZW	RC
256	8001	8229	7566	7396	7785	**6520**	7380	7209	9625	7794
128	8001	6786	6578	6445	6545	**5498**	6322	6075	7703	6804
64	8001	5566	5588	5476	5428	**4483**	5251	4998	5996	5813
32	8001	4470	4607	4502	4377	**3475**	4257	4017	4555	4826
16	8001	3480	3672	3548	3387	**2546**	3251	3052	3329	3866
8	8001	2511	2728	2614	2436	**1758**	2321	2198	2296	2928
256	2002	2749	2086	1899	2328	**1810**	1911	2115	2406	2337
128	2002	2051	1843	1685	1836	**1547**	1645	1724	2148	2095
64	2002	1576	1598	1456	1464	**1287**	1393	1414	1760	1849
32	2002	1218	1355	1221	1156	**1033**	1148	1155	1374	1604
16	2002	933	1125	990	889	**787**	934	915	1024	1368
8	2002	676	893	760	643	**573**	715	694	746	1135
256	504	1269	706	**493**	898	510	527	626	577	962
128	504	845	652	454	641	455	**453**	510	559	908
64	504	573	595	407	465	390	**386**	415	500	851
32	504	396	533	353	341	**319**	**319**	340	398	790
16	504	286	478	299	**253**	259	270	273	317	733
8	504	203	420	244	**182**	201	228	222	247	675

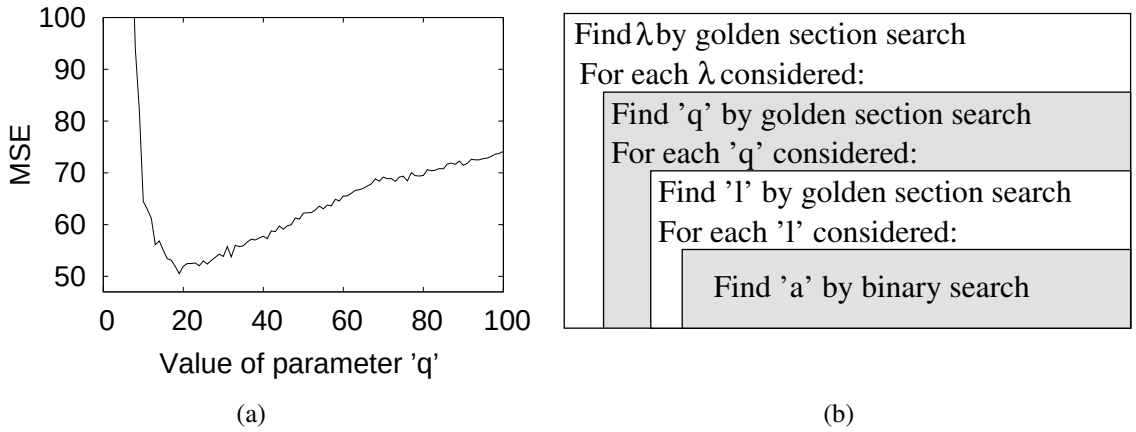


Figure 3.5.: **Left**: A graph showing the MSE when compressing the image trui with a different number of quantised values q. The parameter a has been optimised such that a compression ratio of $40:1$ was obtained using arithmetic coding. All remaining parameters are constant. **Right**: Overview of the recursive algorithm used to find a good set of parameters.

was sufficient to reach the same quality even though the same decoders were still used [181]. Similarly, we expect that improved encoders or faster computers will increase the compression quality of our framework in the future, as we only find a local minimum with the steps explained in this section.

There are several free parameters in our image compression algorithm explained so far: The parameters a and l used to compute the splitting threshold T, the number of possible quantised values q, the minimal and maximal allowed tree depths m and M, and possibly parameters occurring in the inpainting operator such as the contrast parameter λ described in the last section. In this section, we describe the method we have used to find good parameters for a specific compression ratio C.

First, we consider only the parameter a. Reducing the parameter a results in smaller thresholds T, with the consequence that more sub-images are split. Thus, we assume that, the smaller this parameter, the better the reconstruction and the larger the resulting files[1]. In order not to exceed the compression ratio C, we only have to perform a binary search in order to find the smallest parameter a for which the resulting file is at least compressed with this compression ratio. Thus, it is not necessary to perform the decompression step to find this parameter.

When searching the number of quantised values q, the parameter a is searched for each value of q considered. If q is chosen too high, few pixels are saved, as each pixel requires a lot of space. If q is too low, the pixels stored are very inaccurate. Both cases yield bad reconstructions, as can be seen in the graph in Figure 3.5(a). Thus, the optimal value is somewhere in between the extremal values $q = 2$ and $q = 256$. Here, we assume that there is exactly one minimum, and use the *golden section search* [209] to find it. Although this assumption is sometimes violated, the golden section search still results in a good approximation.

[1] Although it is counterintuitive, both assumptions can be violated: Due to entropy coding, storing more pixels can result in smaller files, and specifying more pixels in the interpolation step can result in a worse reconstruction.

The parameter l, which is used in the computation of the splitting threshold T, and the contrast parameter λ are also found using a golden section search in which other parameters are searched for recursively. Note that the order in which the parameters are searched for can make a difference. The ordering we used in our experiments is illustrated in Figure 3.5(b). Any other ordering could be used, though. To speed up the optimisation, we cache the computed inpainting results for each sub-image. As altering λ changes the inpainting result and thus invalidates the cache, we suggest to search for λ last.

The remaining two parameters m and M have a very limited range, and are thus found manually for all experiments shown in this thesis. However, searching them automatically can be done straightforwardly.

3.1.5. Brightness Optimisation

Instead of utilising the (quantised) brightness values of the original image, it is usually beneficial to use modified brightness values to reconstruct the image. This can significantly improve the quality of the reconstructed image, as already reported in [88].

Here, we use a simple and straightforward approach: For each pixel in the inpainting mask K, we replace its brightness if increasing or decreasing it to the next quantisation level reduces the reconstruction error of the image. Thereby, the ordering of the pixels in K is random. Once no single pixel can be optimised any more, the algorithm stops. This way, we reach only a local minimum. Since finding the global minimum is very expensive, we restraint from further optimising the brightness values. For alternatives to the simple approach employed here, we refer to [167] and [111].

3.1.6. Interpolation Swapping

Especially when using large compression ratios, it can happen that the error of the reconstructed image is larger in the inpainting mask K than in the inpainting domain $\Omega \setminus K$. There are several reasons why this can happen. First of all, the pixels in K are quantised quite coarsely, which is not the case for the interpolated pixels. Secondly, higher errors are accepted in the interpolation set K if this is improves the approximation quality within the inpainting domain $\Omega \setminus K$. Thirdly, data is inpainted by PDEs, which basically average information from K in the inpainting domain $\Omega \setminus K$. Even if the stored brightness values in K are erroneous due to quantisation or due to the brightness optimisation step explained in the last section, blending them averages the errors and can lead to less "noisy" results in $\Omega \setminus K$. A similar observation can be made in variational optic flow estimation where the results at locations with a large data term are worse than those in areas where the smoothness term with its inpainting effect dominates [40]. The fourth and last reason only applies when using an isotropic diffusion operator. In such a setting, inpainting results can become singular close to an isolated point in K. In the linear case, this can be explained with the well-known logarithmic singularity of the Green's function of the 2-D Laplacian.

This problem has already been reported in [15]. This paper also proposes a relatively simple solution. After the initial inpainting step is done, the role of known and unknown pixels is exchanged. That is, the reconstructed points in $\Omega \setminus K$ are assumed to be known, and the points

in K are reconstructed using the same inpainting algorithm as before. The resulting image is much smoother and propagates the high quality solution from $\Omega \setminus K$ to the more inaccurate data in K. This approach can be justified as a numerical consistent approximation of the inpainting PDE in the sense of so-called Hopscotch schemes [99].

In this thesis, we propose to further extend this idea: Instead of only recomputing the points in K, we even allow to recompute all points within a certain distance to the points in K. We denote this step by *interpolation swapping* in this thesis. According to our experiments, the larger the compression ratio, the larger this distance should be. The radius for which the best results are obtained is stored in the file header, and will be used in the decompression step.

It should be noted that the three optimisations introduced in the last three sections influence each other. For example, changing the radius used for interpolation swapping may change the optimal brightness values. Thus, the diffusivity optimisation from Section 3.1.3, the brightness optimisations from Section 3.1.5, and the search for the best radius for interpolation swapping explained in this section are interleaved (in this order) by our algorithm.

3.1.7. Colour Images

So far, we have only explained how to compress grey-scale images. However, as we will use our image compression algorithm to compress colour images in Chapter 10, we now introduce a version of our algorithm that can handle colour images.

As a first variant to handle colour images, we have implemented a rather simple approach: When a colour image is to be compressed, the R, G, and B value of each pixel indicated by the inpainting mask are stored in an interleaved way. The file containing all colour channels is then compressed together using the chosen entropy coder. An additional bit in the file header is used to indicate whether a grey-scale or colour image has been stored. In the decompression step, all three channels are reconstructed simultaneously using vector-valued inpainting operators [288]. Note that this yields different results than inpainting each channel separately when a nonlinear inpainting process is used, since the argument of the diffusivity function depends on all channels.

3.1.8. File Format

Before we discuss the decompression algorithm, we will quickly explain the exact file format we have used for the experiments in this thesis.

For the image size, we first store a single bit indicating if one or two bytes must be used to store the width and height of the image. Since width and height are positive, one byte is sufficient if width and height of the image are smaller or equal to 256. The next bit indicates if width and height are identical, i.e. if one or two numbers must be saved. Finally, between one and four bytes are necessary to store the actual width and height. That is, ten bits (one byte and two bits) are sufficient for a small square image, while 34 bits (four bytes and two bits) are necessary for a large image whose width and height differ.

The next three bits indicate which of the first eight entropy coders from Table 3.2 must be used to decompress the brightness values.

Then, four bits are used to indicate the radius used in the interpolation swapping step. Thereby, each of the sixteen possible values stands for one specific radius such that all discrete circles with radii between 0 to $\sqrt{26}$ can be used. When using the maximal radius of $\sqrt{26}$, 88 points around each given point will be reconstructed. This is much more than necessary for all our experiments.

Next, the quantisation level q of the brightness values is stored. This allows to store these values as numbers between 0 and $q-1$ instead of the actual brightness values, which improves the performance of some entropy coders. Especially when using PAQ, this additional byte can reduce the total file size by a few percent.

The next bit is straightforward and indicates whether a grey-scale or colour image is compressed.

The quantised contrast parameter λ that should be used in the reconstruction step is stored as a single byte, as explained in Section 3.1.3.

To store the maximal depth of the splitting tree, we use the knowledge that this number is nonnegative and bounded by a number that can be computed from the image size. Thus, one can easily compute the number of bits necessary to store the maximal tree depth. Similarly, the number of bits for the minimal tree depth, which is nonnegative and bounded by the maximal tree depth, can be computed.

Next, the tree containing the splitting decision (see Section 3.1.1) is saved by storing a single bit for each decision. Here, we traverse this tree in breadth-first order, but any other ordering results in the same file size.

In the remainder of the file, the quantised and entropy encoded brightness or colour values are stored. This is done by scanning the interpolation mask row-wise from top left to bottom right. Whenever the interpolation mask indicates that information at a certain position should be kept, the quantised brightness or colour values at that point are stored in a file, which is encoded afterwards.

The following list summarises our file format described in this section:

- image size (between 10 and 34 bits)
- type of entropy coder (3 bits)
- radius of interpolation swapping (4 bits)
- number of quantisation levels (1 byte)
- flag whether a colour or grey-scale image is compressed (1 bit)
- contrast parameter (1 byte)
- minimal and maximal depth of the tree (variable size)
- splitting information as a binary tree (variable size)
- compressed brightness or colour values (variable size)

Our current file format does not contain magic number or checksums, which are very common in standard file formats. However, they can be included easily, and only result in a constant overhead which does not invalidate the good compression performance of our codec. Similarly, other information such as the type of inpainting operator used can easily be added if necessary.

3.1.9. Encoding and Decoding Algorithms

In the decompression step, our algorithm first reads the file header explained in the last section. Using the tree containing the splitting data and the image size, the inpainting mask is created. Next, the compressed brightness or colour values are decompressed, mapped according to the number of quantisation levels, and put into an otherwise black image at those points indicated by the inpainting mask. Then, the remainder of the image is reconstructed by the same inpainting algorithm as in the compression step. As a last step, an interpolation swapping step (see Section 3.1.6) is done, if necessary.

3.1.10. Numerical Implementations

To solve the inpainting problem with different PDEs, we have used a large variety of numerical schemes. Most of them are based on finite difference discretisations of the evolution equation (2.6), and use (semi-)implicit time discretisations to allow using large time steps. In these approaches, SOR (successive overrelaxation) or the conjugate gradient method are used to solve the occurring systems of equations. Alternatively, fast explicit diffusion schemes [101], as well as multigrid methods have been employed. Exceptions are biharmonic and triharmonic smoothing, for which pseudospectral methods have been used, and the approach by Bornemann and März (see Section 3.2.1), which is not iterative and employs a fast marching approach. For the method by Tschumperlé, we used the result kindly provided by David Tschumperlé. However, an open source implementation is also available [277].

3.2. Experiments

"A fool is a man who never tried an experiment in his life."

Erasmus Darwin (1731 – 1802)

The experiments presented in this section have two purposes: First, we compare the performance of several differential operators in our image compression framework using the image "trui", see Figure 3.6(a). Additionally, we look at specifically chosen inpainting problems. In the second section, we compare our codec against JPEG, JPEG 2000, and the approach by Galić *et al.* [88] using the standard test images "trui", "walter", and a sub-image of "peppers". These three images are shown in Figure 3.6. The latter two images are available from the SIPI web page `http://sipi.usc.edu/database` of the University of Southern California.

(a) Trui, 257×257 (b) Walter, 256×256 (c) Peppers, 257×257

Figure 3.6.: Original images used to evaluate the performance of our compression algorithm.

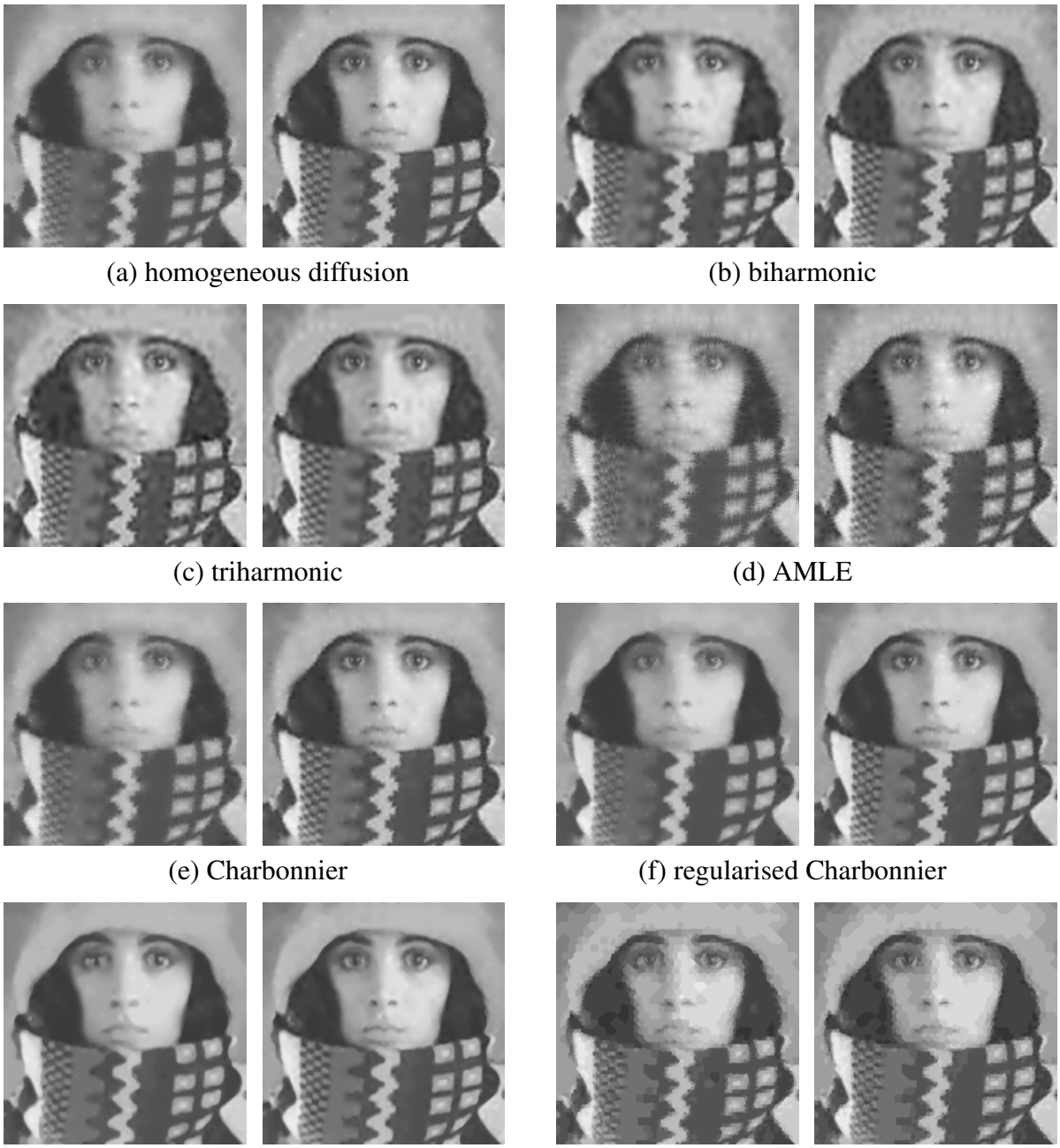
Table 3.3.: Performance of different inpainting operators in our image compression framework. This table evaluates the results using the operators introduced in Section 2.1, and the approach by Bornemann and März (see [26]), which is abbreviated as "BM". **First line**: MSEs obtained. **Second line**: Number of quantisation levels used. **Third line**: Total number of brightness values saved. **Fourth line**: Depth range of the adaptive tree used to store the pixel locations.

	Homogeneous	Biharm.	Triharm.	AMLE	Charb.	regul. Charb.	EED	BM
MSE	35.40	39.79	42.83	64.94	35.25	29.53	**28.29**	69.45
quantisation levels	11	20	17	11	11	11	24	17
points saved	3502	2858	3021	3469	3510	3516	2750	2991
depth of tree	9–13	10–13	10–13	8–13	9–13	9–13	9–12	9-13

Note that, strictly speaking, the approach by Galić *et al.* is only explained in [88] for square images whose width and height have the form $2^n + 1, n \in \mathbb{N}$. Despite the fact that the image "walter" does not have this form, we include this method in our comparison. This is possible since the implementation kindly provided by Irena Galić can in fact handle images with different resolutions.

3.2.1. Comparison of Differential Operators

In total, we evaluate eight different PDE-based inpainting operators in our image compression framework. These eight operators are: *homogeneous diffusion* (see Equation (2.7)), *biharmonic* smoothing (see Equation (2.12)), *triharmonic* smoothing (see Equation (2.13)), *AMLE* (see Equation (2.19)), Nonlinear isotropic diffusion (*Charbonnier*, see Equation (2.14)), Nonlinear isotropic diffusion with presmoothing (*regularised Charbonnier*, see Equation (2.17)), *EED* (see Equation (2.18)), and the method by Bornemann and März (see [26]). The latter method is a non-iterative algorithm that combines a fast marching method with a coherence transport model, and is based on Telea's algorithm [272]. As test setup, we use the image "trui"



(a) homogeneous diffusion (b) biharmonic

(c) triharmonic (d) AMLE

(e) Charbonnier (f) regularised Charbonnier

(g) EED (h) Bornemann and März

Figure 3.7.: Visual performance evaluation of different inpainting algorithms in our compression framework. The left image in each block shows the image before optimisation (brightness optimisation, interpolation swapping, improved diffusion parameters), while the right image shows the image after optimisation. For each operator, the point mask was optimised such that a compression ratio close to $45:1$ using Huffman coding was obtained. For a numerical evaluation, see Table 3.3.

(see Figure 3.6(a)) with a compression ratio of approximately $45 : 1$. All other parameters are optimised individually for each method such that the best results are obtained.

Figure 3.7 illustrates the visual quality of each of these eight methods with and without brightness optimisations, interpolation swapping, and improved diffusion parameters, while results of a numerical evaluation are shown in Table 3.3. If the differences between these results are not visible in the printed version, we suggest to look at the images in the PDF-file.

Considering these experiments, on can see several things. First of all, using EED yields the best results. This is not only true with respect to the *MSE* (see first line in Table 3.3), but can also be seen visually when zooming into the images. The results from all other methods have deficiencies such as the singularities visible with standard or regularised Charbonnier interpolation or homogeneous interpolation. Another example are the fluctuations at edges when using bi- or triharmonic interpolation. The latter is due to over- and undershoots, and appears since these higher-order differential operators do not fulfil the maximum-minimum principle. For the remaining second-order operators, the information in the selected pixels seems to be insufficient to create sharp edges, while the method by Bornemann and März creates too many edges.

Furthermore, one can see that EED needs less points and a less precise localisation of the point locations, resulting in a small depth range of the binary splitting tree. This results in a smaller file header, and less brightness values. Consequently, EED may use a larger number of quantisation levels for the same compression ratio to obtains a better approximation quality.

To illustrate why EED has such a good performance compared to other PDEs, let us consider the experiment depicted in Figure 3.8. For this experiment, information inside the three discs visible in the input image is given, while the remainder of the image is initialised with random noise. Due to the white wedges present in the discs, a human observer can see a triangle. Thus, this experiment resembles the well-known *Kanizsa triangle* [137].

When comparing the inpainting results with the different operators, one can see that EED yields an almost perfect reconstruction. This is due to the combination of two properties of EED: its anisotropy, which allows to create sharp edges, and its semi-locality obtained by presmoothing, which propagates information about preferred directions from the given data points to other image regions.

Note that the method by Bornemann and März also creates an almost perfect triangle, but still yields bad results in our compression framework. This is due to two reasons. First of all, this method is well suited for connecting level lines. This is sufficient to yield a good result in the experiment with the three circles since basically three level lines are to be reconstructed the direction of which is easy to detect. In natural images, however, it is unclear how the level lines look like when only single image points are given. The second reason is that the method by Bornemann and März is often quite sensitive to the choice of its parameters. Thus, slight variations can significantly alter the results, as shown in Figure 3.8. As a consequence, it is unlikely that there is one parameter set that yields a good reconstruction in all parts of a natural image.

For this experiment, we additionally evaluated a method by Tschumperlé, which uses a tensor-driven PDE that regularises images while respecting some curvature constraints [276]. Independent of the parameters used, results are not satisfactory. Moreover, the method is rather slow and did not converge even after 5000 iterations in most cases even though this

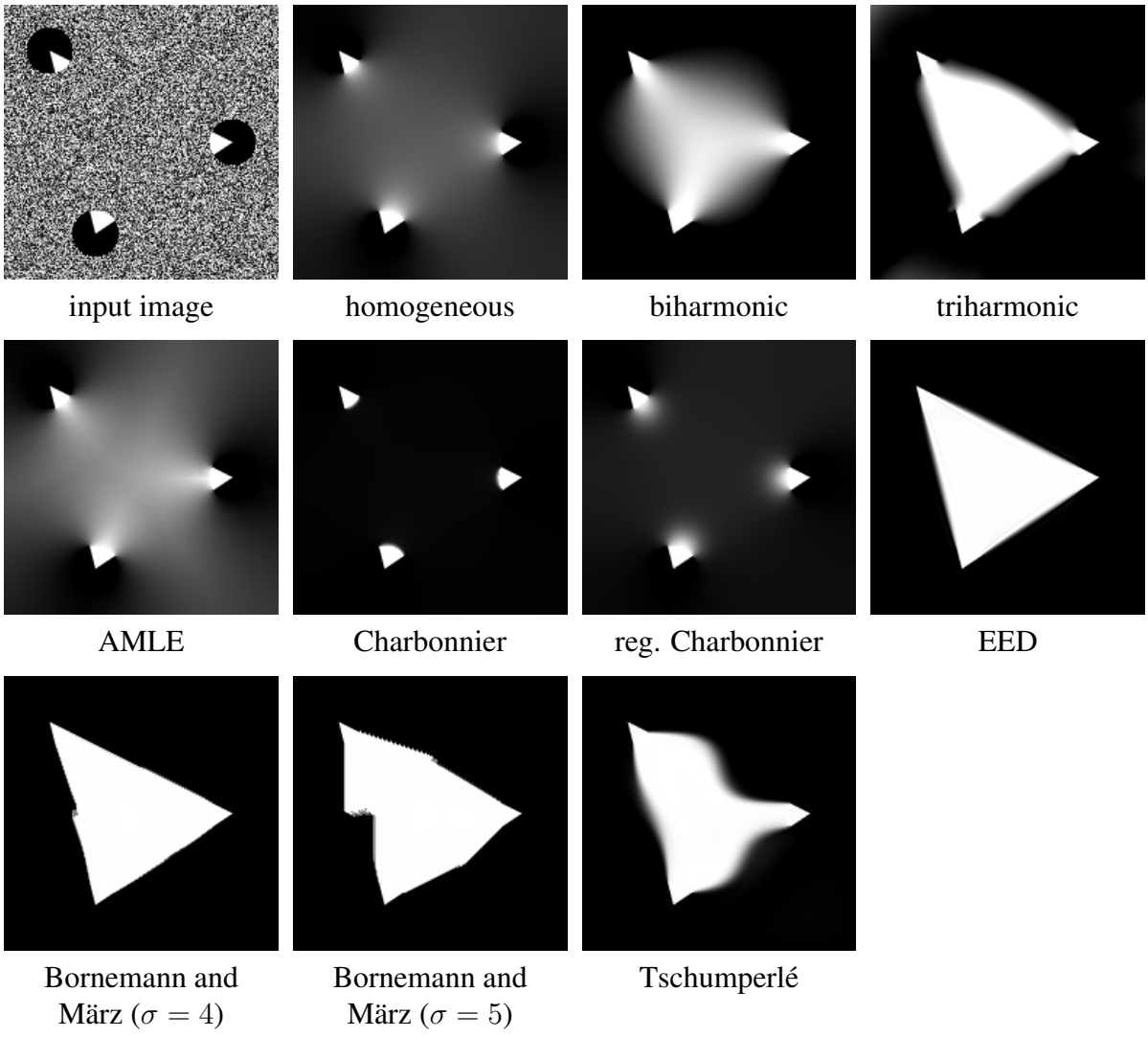
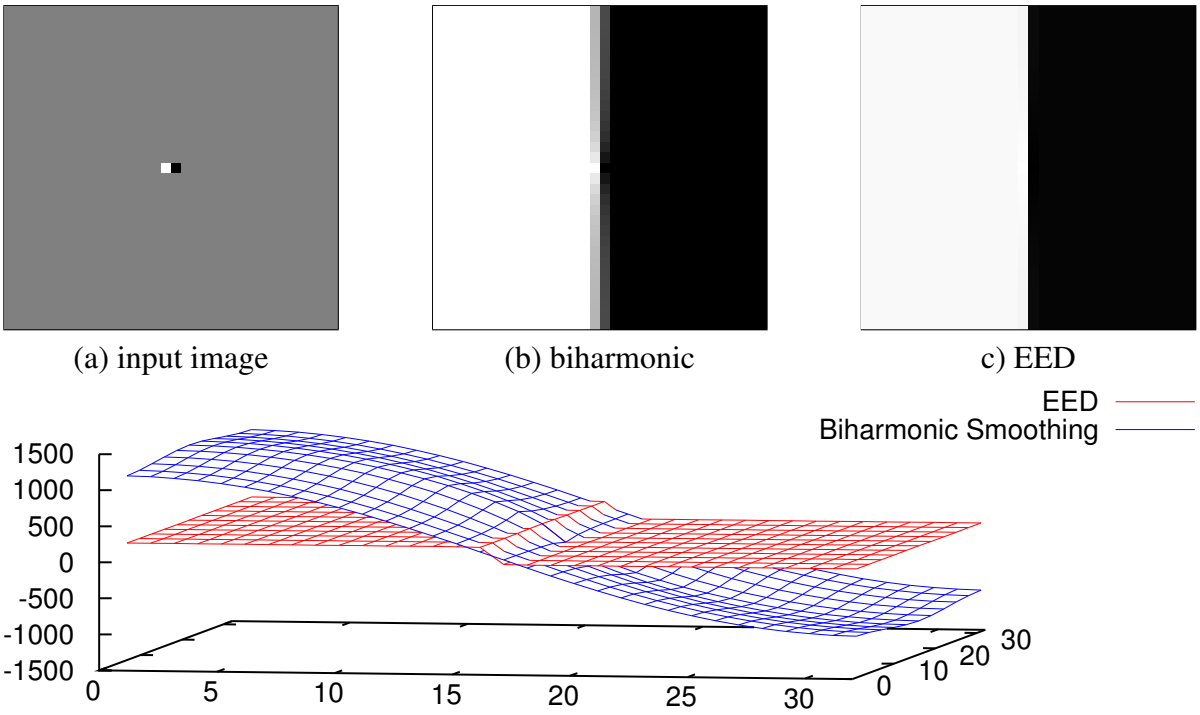


Figure 3.8.: **First image**: Input image with size 189×189. The values inside the three disks are given, while the remaining parts of the image was initialised with uniform noise. **Remaining images**: Reconstruction results with homogeneous diffusion interpolation, biharmonic interpolation, triharmonic interpolation, AMLE, Charbonnier interpolation ($\lambda = 1$), regularised Charbonnier interpolation ($\lambda = 0.1$, $\sigma = 8$), EED interpolation ($\lambda = 0.01$, $\sigma = 4$), a method by Bornemann and März [26] ($\epsilon = 5, \kappa = 100, \sigma = 4/5, \rho = 8$), and a method by Tschumperlé [276] (30 global and local iterations, $dt = 17.86, \alpha = 2.81, \sigma = 0.27$).



(a) input image (b) biharmonic c) EED

(d) Both results plotted as height fields

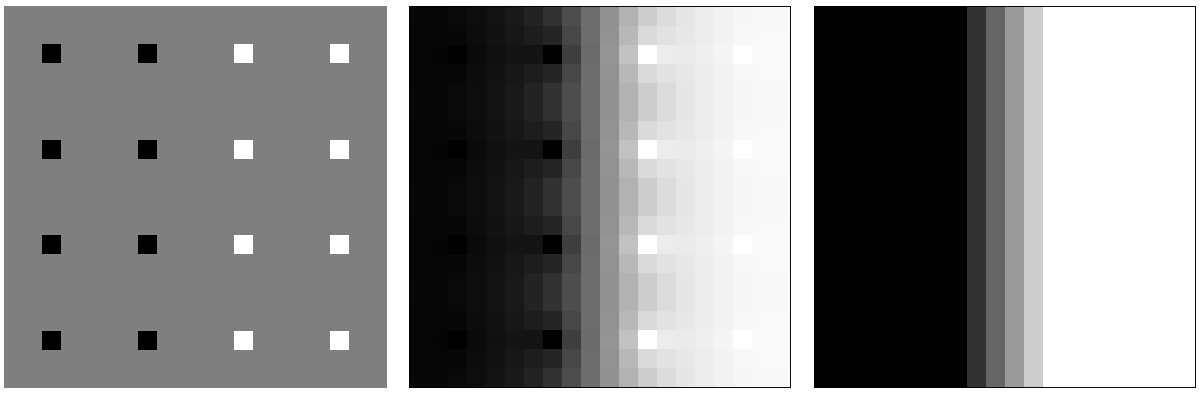
Figure 3.9.: **(a) Top left**: Input image with size 32×31. Only the two non-grey pixels in the middle of the image are given, while the remaining pixels are to be reconstructed. **(b) Top middle**: Interpolation result using biharmonic interpolation. **(c) Top right**: Interpolation result with EED. **(d) Bottom**: Results of both interpolation approaches plotted as height fields. Biharmonic interpolation (blue) produces large over- and undershoots, while EED (red) stays within the specified grey value range $[0, 255]$.

is 25 times the number of iterations used in the original paper. Furthermore, one can easily see that the resulting image is not rotationally invariant, as it seems to prefers horizontal and vertical structures. Due to these reasons we refrained from evaluating this method in our compression framework in the remainder of this chapter.

As can be clearly seen in the results shown, AMLE smoothes only in a direction perpendicular to isophotes. This is the main reason for its bad performance in both experiments.

The remaining operators are unable to create a good triangle since they are isotropic. Even though biharmonic and triharmonic interpolation create blurry triangle-like structures, this is caused by the higher degree of smoothness of the solution, not by any kind of anisotropy.

This high degree of smoothness can be problematic, though, as illustrated in the experiment shown in Figure 3.9. In this experiment, only a black and a white pixel in the middle of the image are given. When inpainting this “dipole” with EED and biharmonic interpolation, both operators split the image into a white and a black part. Since biharmonic interpolation tries to create a smooth result, the edge is quite blurry. Even worse, this results in large over- and undershoots at both sides of the edge since biharmonic interpolation violates the maximum-minimum principle.



input image homogeneous EED

Figure 3.10.: Inpainting experiments where 16 pixels are specified in an image of size 20×20 (left image). The results when inpainting with homogeneous diffusion and EED ($\lambda = 0.01, \sigma = 0.8$) are shown in the middle and right image, respectively. While interpolation with isotropic linear diffusion creates singularities at the specified pixels near edges, such artifacts are not visible with EED.

In Figure 3.8, one can also observe that unregularised Charbonnier interpolation fills the entire image region in black. This is due to the fact that regularised Charbonnier interpolation approximates TV interpolation (see Section 2.1.3). Since a large part of the boundary of the discs are black, the total variation is minimised by this solution. Consequently, TV disregards a part of the given data, which is undesired for compression.

Note that EED also utilises (an approximation of) TV interpolation across edges. In contrast, however, it uses homogeneous interpolation along edges. Thus, it does not create singularities as visible for isotropic second-order operators such as linear diffusion, standard or regularised Charbonnier interpolation (see corresponding images in Figure 3.7). Due to the vanishing diffusion across pronounced edges, the interpolation problem is segmented into subproblems in which the specified grey values are similar if they belong to the same segment. In each segment, essentially isotropic linear diffusion interpolation is performed. Since the grey values in each segment are similar, logarithmic point-like singularities, which often occur with homogeneous diffusion in the presence of fluctuating data, do not appear. In fact, EED only creates singularities across edges, i.e. it creates sharp edges, which is desirable in image compression.

This behaviour is further illustrated in Figure 3.10. While both methods create a smooth transition between the left and the right part of the image, the singularities when using homogeneous diffusion clearly stick out.

3.2.2. Comparison to Other Compression Methods

Let us now compare the performance of our image compression codec with the standard compression algorithms JPEG and JPEG 2000, as well as with the algorithm proposed by Galić *et al.* in [88]. For this comparison, we have used the tool "convert" (version ImageMagick

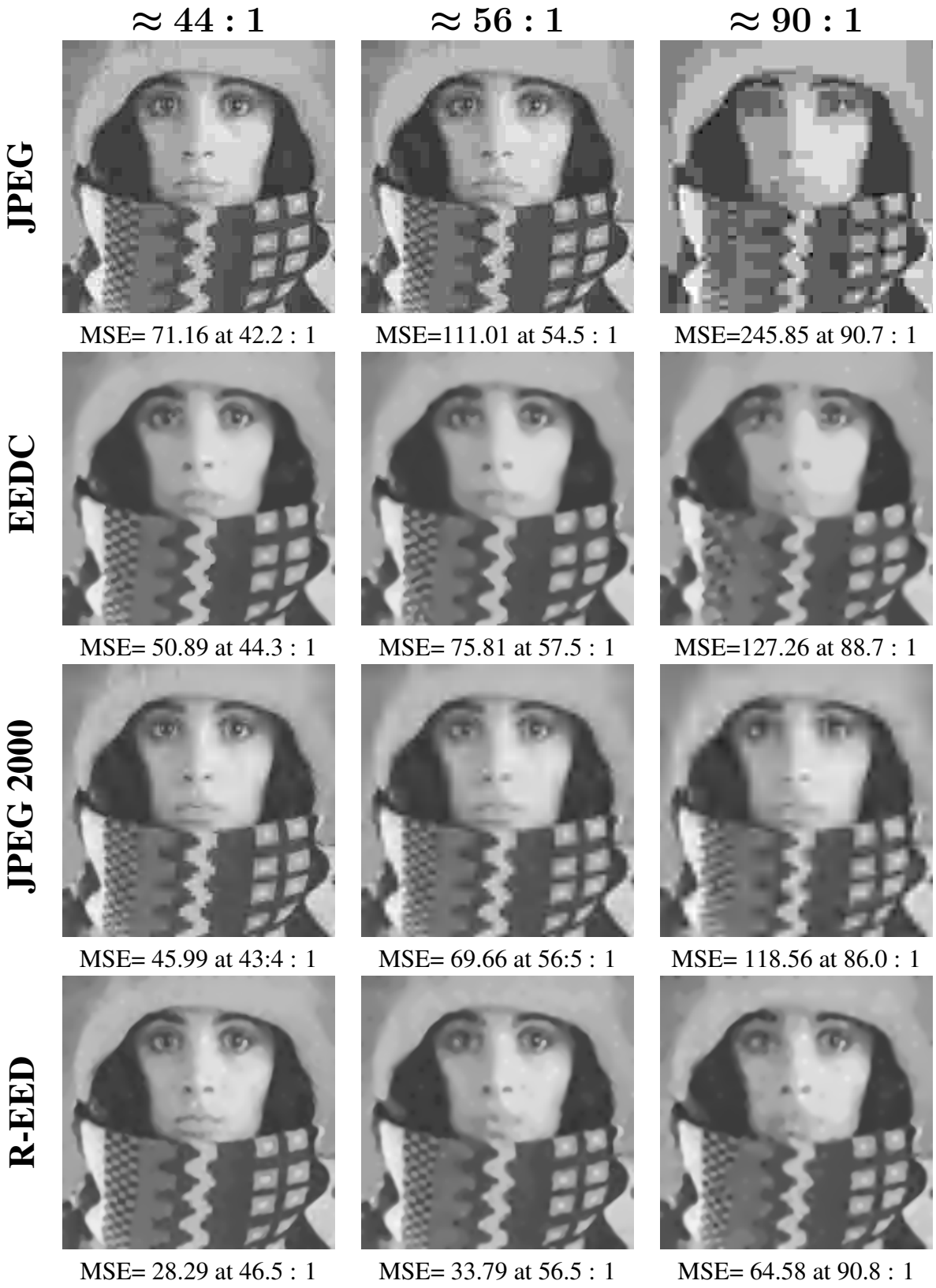


Figure 3.11.: Comparison of JPEG, EEDC, JPEG 2000, and our compression algorithm (R-EED) using the image “trui” and three different compression ratios.

Figure 3.12.: Comparison of JPEG, EEDC, JPEG 2000, and our compression algorithm (R-EED) using the image “walter” and three different compression ratios.

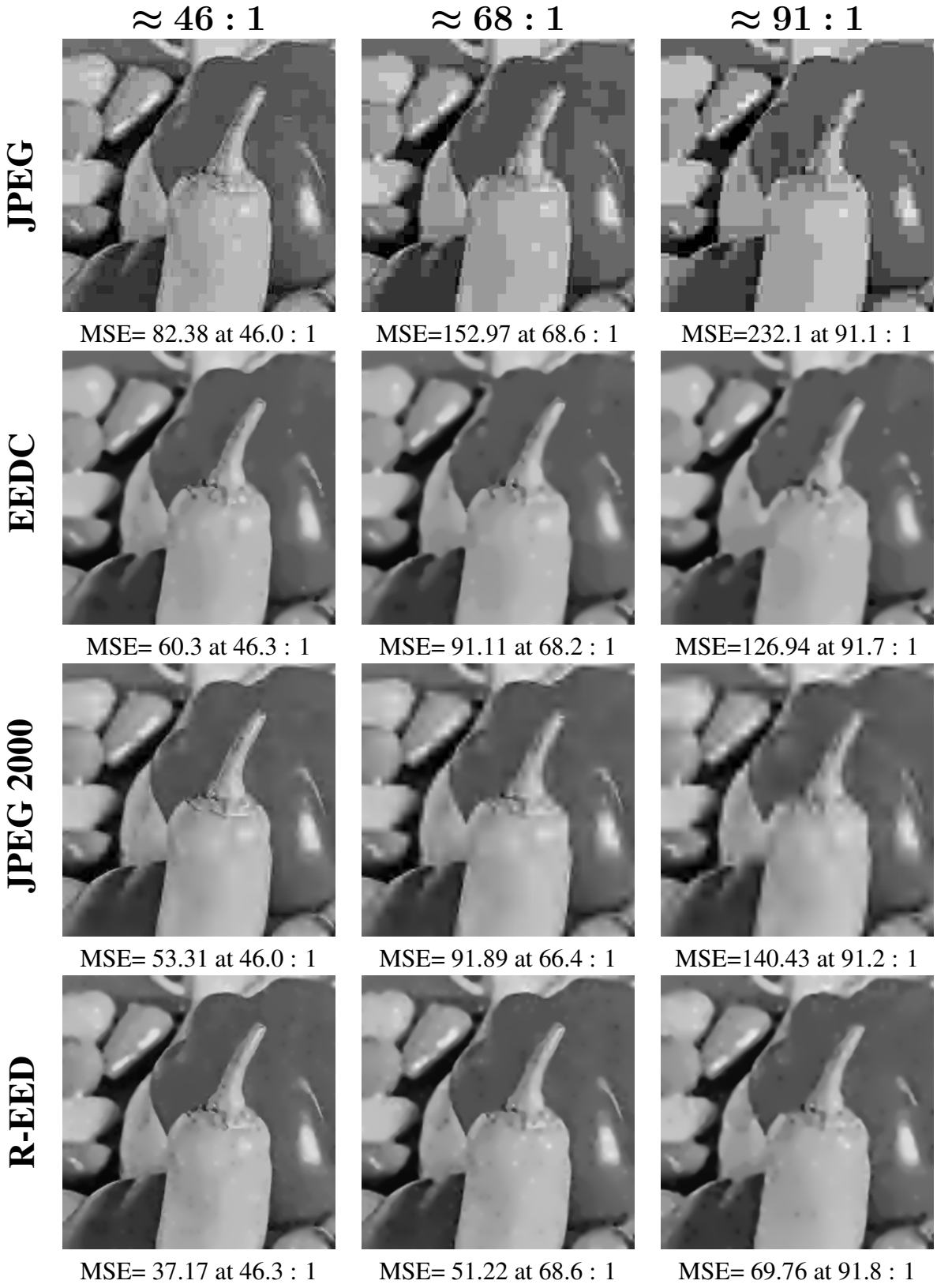


Figure 3.13.: Comparison of JPEG, EEDC, JPEG 2000, and our compression algorithm (R-EED) using the image “peppers” and three different compression ratios.

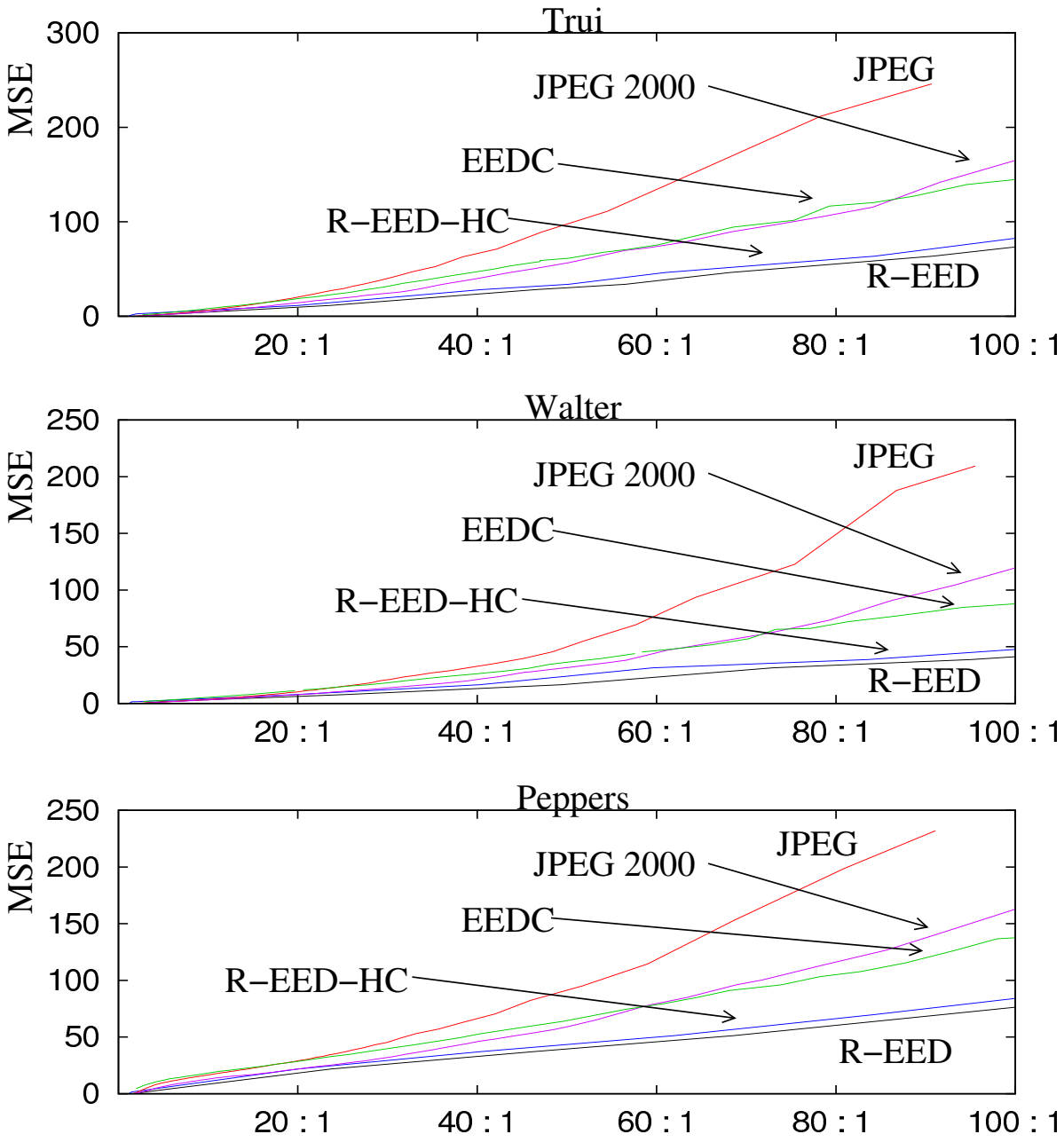


Figure 3.14.: Plots illustrating the performance of JPEG, JPEG 2000, R-EED, R-EED with Huffman coding (R-EED-HC), and EEDC for a large range of compression ratios by plotting the MSE against the compression ratio. **Top**: Result for the image “trui”. **Middle**: Result for the image “walter”. **Bottom**: Result for the image “peppers”.

6.2.4 02/10/07) and the source code kindly provided by Irena Galić. One should note that convert uses optimised entropy coding parameters when saving JPEG files, while this was not the case for the JPEG images shown in [88]. Thus, the images shown here differ from the images shown in [88]. Further note that we will denote the results by our algorithm by *R-EED* from now on, while the results by the approach by Galić *et al.* are denoted by *EEDC*.

For this evaluation, we compress the three grey-scale images shown in Figure 3.6 with all four methods using a large range of compression ratios. Example images with three different compression ratios for each method are shown in Figure 3.11 for the image "trui", in Figure 3.12 for the image "walter", and in Figure 3.13 for the image peppers. Figure 3.14 shows plots that illustrate the MSE obtained with each method using various compression ratios up to $100:1$.

It is obvious that, for all three images, JPEG yields the worst results. Especially at high compression ratios, block artifacts are clearly visible. Such artifacts are typical for JPEG, which tiles the image into patches of size 8×8 each of which is processed and quantised independently.

In contrast to the statement in [88] that their "EED-based codec ... even comes close to the quality of the highly optimised JPEG 2000...", one can see that the method by Galić *et al.* can even produce slightly better results than JPEG 2000 for sufficiently high compression ratios. However, R-EED produces results that clearly surpass those of EEDC for all three images and all compression ratios. This can be seen best when considering the image "walter" for which the MSE obtained with R-EED is always less than half the MSE obtained with EEDC. Even for the image "peppers", for which the performance difference between both methods is smallest, the MSE with EEDC is 62% larger than with R-EED for a compression ratio around $46:1$, and even 82% larger for a compression ratio around $91:1$. This clearly demonstrates the effect of the improvements introduced in R-EED.

R-EED also yields excellent results when it is compared to the sophisticated JPEG 2000 standard. For the image "peppers" and a compression ratio around $45:1$, the MSE when compressing with JPEG 2000 is 43% larger. For larger compression ratios, the difference is even more noticeable: The MSE with JPEG 2000 is more than twice as large as those obtained with R-EED for a compression ratio around $91:1$. Similarly R-EED yields better results than JPEG 2000 for the other images. When using the image "trui", the MSE of JPEG 2000 is 63% larger for a compression ratio around $44:1$, and 83% larger for a compression ratio around $90:1$. For the image "walter" the improvement is even bigger: JPEG 2000 has a 84% larger MSE for a compression ratio around $49:1$, and a 134% larger MSE at a compression ratio around $94:1$. Also note that the size of the compressed file was always slightly lower with R-EED than with JPEG 2000 in Figure 3.11, 3.12, and 3.13.

It is well known that PDE-based inpainting methods deteriorate in highly textured regions. However, R-EED still yields a better visual impression than JPEG 2000 in the textured regions of the scarf in Figure 3.11. JPEG 2000, on the other hand, is known to create blurry edges, or ringing artifacts. It seems the ability of EED to create sharp edges allows to use more points in textured regions, thus yielding a better reconstruction quality.

Neither our implementation of JPEG 2000 nor the method of Galić *et al.* use an advanced entropy coder. Thus, the graphs in Figure 3.14 also show results of our codec when a simple Huffman coding is used instead. Our results with Huffman coding are denoted by R-EED-

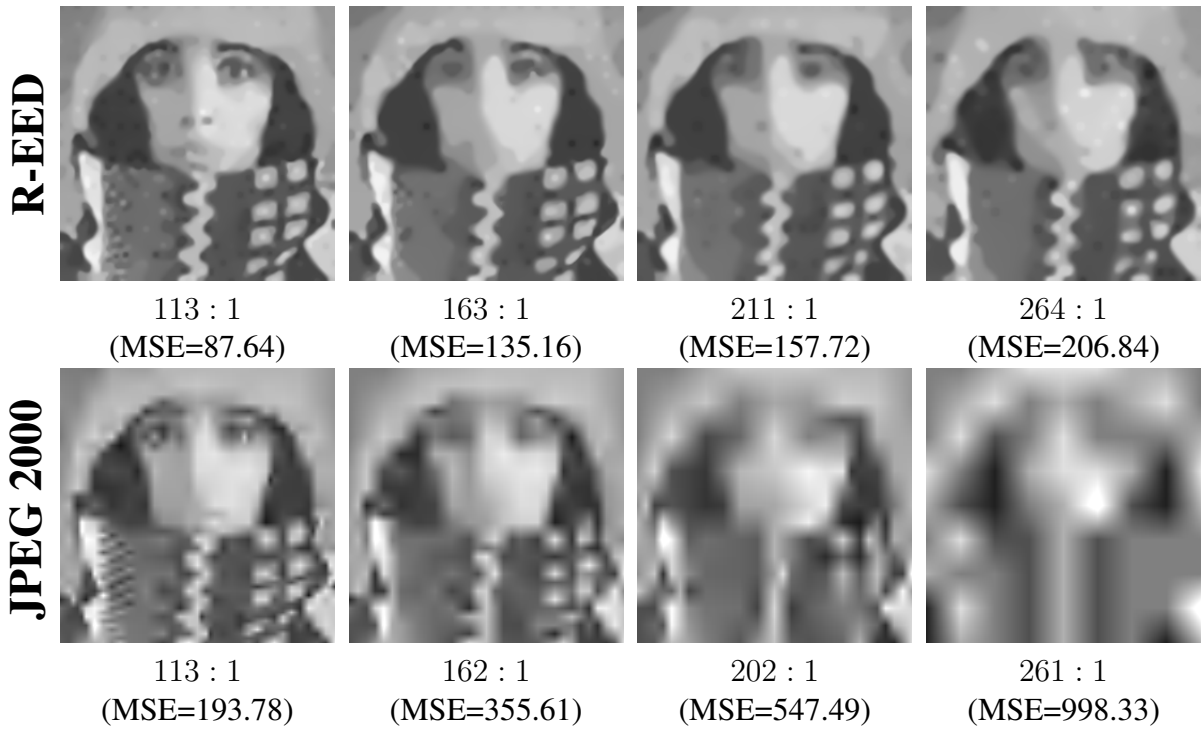


Figure 3.15.: Results with very high compression ratios. **Top**: Results generated with R-EED. **Bottom**: Results generated with JPEG 2000.

HC in these graphs. As can be clearly seen, this only slightly reduces the efficiency of our compression algorithm.

Especially when considering the graphs in Figure 3.14, it is noticeable that the performance of the EED-based compression algorithms (EEDC and R-EED) scales better with respect to the compression ratio than those of the transform-based approaches JPEG and JPEG 2000. Thus, we also compare results with very high compression ratios in Figure 3.15 for R-EED and JPEG 2000. Although the image quality with R-EED declined noticeably, one can see still guess pretty well what is depicted in the rightmost image. This is not the case with JPEG 2000, which is also reflected in the much higher mean square error when using JPEG 2000.

Finally, we compress a coloured version of the image peppers, which is shown in the upper left corner of Figure 3.16. Due to the trivial handling of colour (see Section 3.1.7), the difference of R-EED to JPEG and JPEG 2000 is closer than for the grey-valued variant of this image. This experiment shows that even with the most simple way to handle colour images, our compression algorithm already yields better results than the modern JPEG 2000 standard. With a more sophisticated colour handling, the performance difference will increase further.

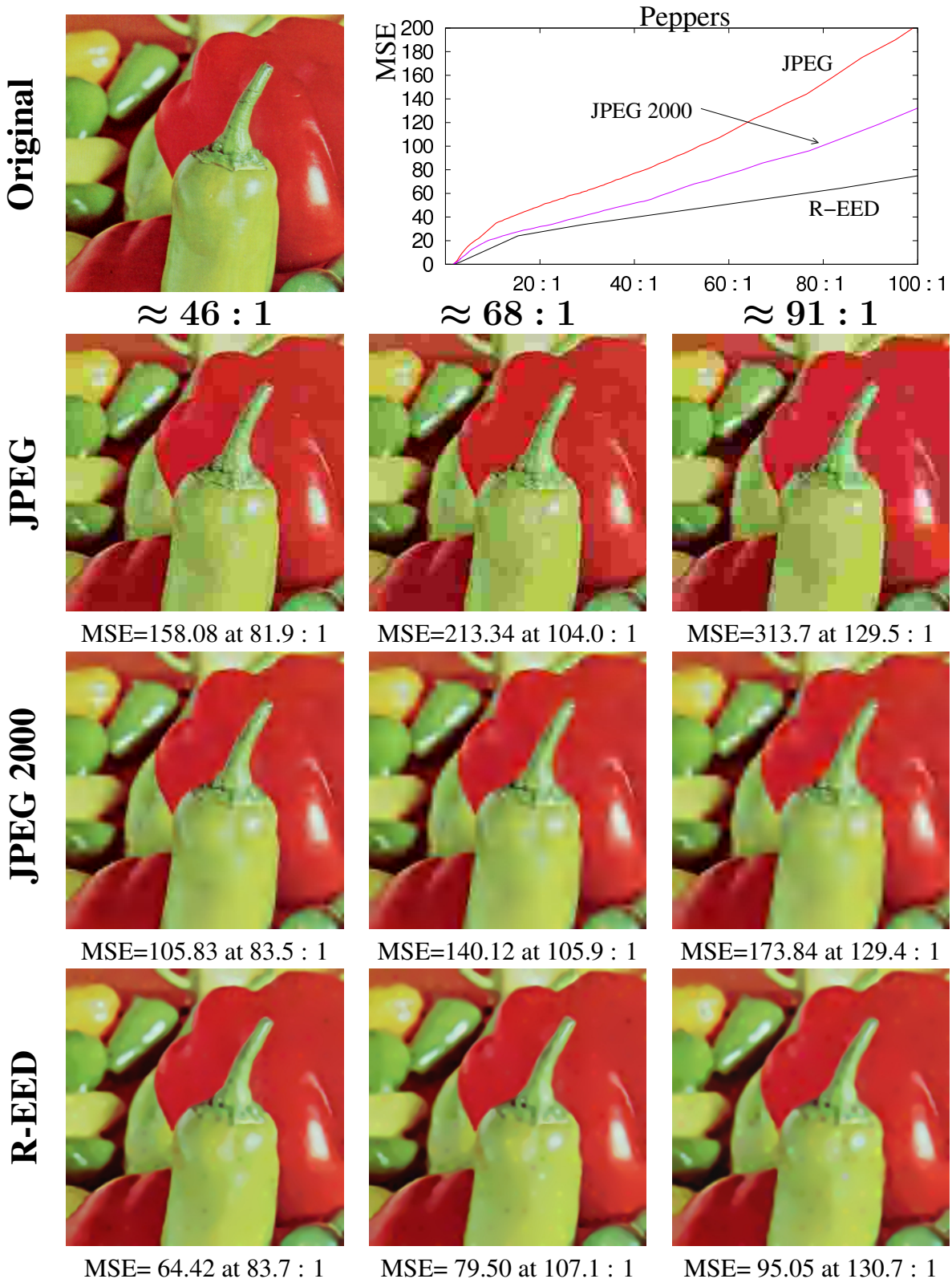


Figure 3.16.: Comparison of JPEG, JPEG 2000, and our compression algorithm (R-EED) using a colour version of the image "peppers" and three different compression ratios. The method by Galić *et al.* is not included, as it is not implemented for colour images.

3.3. Summary

"A conclusion is the place where you got tired thinking."

Martin H. Fischer (1879–1962)

In this chapter, we developed an image compression codec based on partial differential equations. We have demonstrated that edge-enhancing anisotropic diffusion (EED) is very well suited for our codec, since it behaves both anisotropic and semi-local, fulfils the maximum-minimum principle, and does not create singularities.

Furthermore, we have demonstrated that our codec outperforms both the approach it is based upon [88] as well as JPEG and JPEG 2000, especially for high compression ratios.

In the following chapters, we introduce the 3-D pose tracking algorithm we developed.

4 Silhouette-based Tracking

"It is necessary for me to establish a winner image.
Therefore, I have to beat somebody."

Richard M. Nixon (1913 – 1994)

The task to locate and follow an object in a video sequence is a core problem in computer vision, which is called *tracking*. It is necessary or helpful in a multitude of applications, including self localisation and object grasping in robotics, body language interpretation, foreground-background segmentation, human computer interaction, traffic or security surveillance, character animation, analysis of athletes, and content-based video retrieval.

Although tracking is quite easy for humans, it is still a challenging problem for computers due to the possible problems that may arise. Such problems include scenes with (partial) occlusions, cluttered backgrounds, noise, objects barely distinguishable from the background, or changing illumination conditions. Thus, it is not surprising that a satisfactory solution has not been found yet, even though about thirty years have passed since the initial work by Lowe [160]. Here, we will only give a brief summary of the topic. For a detailed overview of the field, we refer to the surveys by Gavrila [91], Aggarwal and Cai [5], Moeslund and Granum [177], Lepetit and Fua [153], Forsyth *et al.* [83], Moeslund *et al.* [178], Yilmaz *et al.* [301], and Poppe [208].

Depending on the task at hand, the requirements imposed on a tracking algorithm can vary greatly. For some application, such as traffic surveillance or face tracking, it can be sufficient to mark every object by a rectangle or circle [14], while foreground-background segmentation requires a decision for each pixel in the image. Other algorithms might only return a single point in the centre of the object, a silhouette, or some control points on the object contour. There are also many other possibilities how the result of a tracking algorithm may look like, some of which are illustrated in Figure 4.1.

If the object model is given as kinematic chains (see Section 2.4.2), tracking algorithms that return lines or elliptical patches that indicate the positions of different body parts are also common [301].

In contrast to those 2-D approaches, many applications try to solve the more difficult task of *3-D pose tracking*. The word *pose* refers to the transformation needed to map an object model from its own inherent coordinate system into agreement with the sensory data [102].

That is, for 3-D pose tracking, the goal is to find the position and orientation of the object to be tracked in 3-D space. For articulated objects, e.g. humans, additional parameters such as joint angles must also be estimated. Some common kinds of visualisation used for tracking results of kinematic chains are illustrated in Figure 4.2: Visualisation using a stick model (leftmost), using one 2-D primitive (left), e.g. a rectangle or an ellipse, for each part of the chain, or using the projection of the model utilised for tracking (right), possibly with multiple

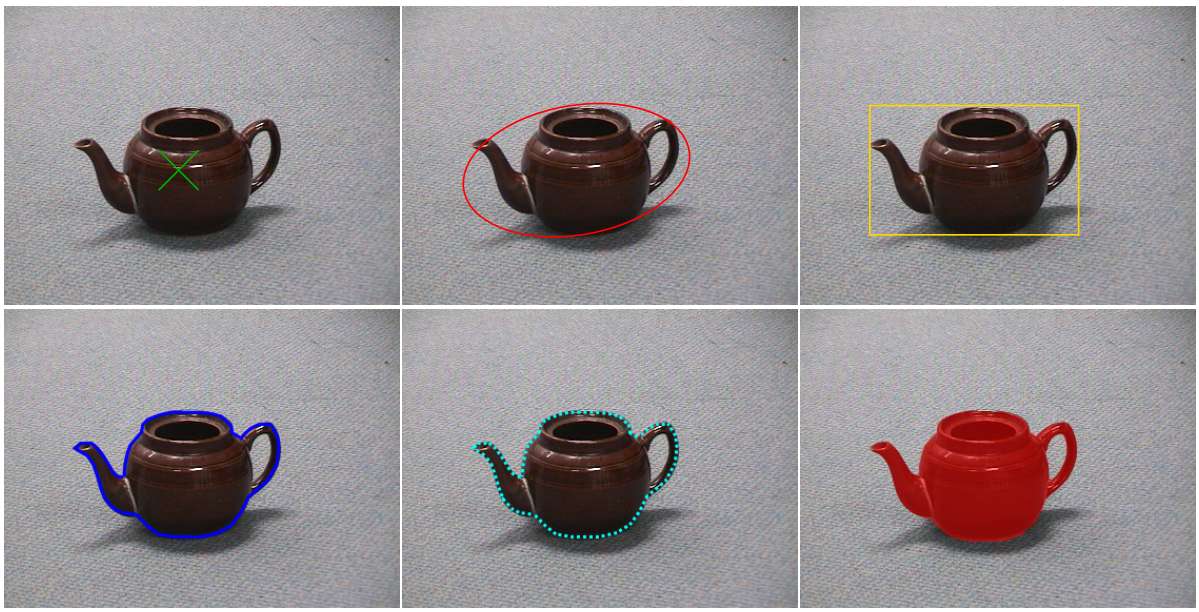


Figure 4.1.: This images illustrate different kinds of results a 2-D tracking algorithm can produce. **From upper left to lower right**: Tracking of a single point, an ellipse, a rectangle, a silhouette, silhouette points, and of 2-D regions.

internal parts (rightmost). Again, it is possible to consider other possibilities how to indicate where the tracked object parts are. For example, a set on non-connected edges is used in [261].

If a tracking via general 2-D regions was done, it is very easy to get a surrounding rectangle or circle. Thus, 2-D pose tracking is obviously more difficult than tracking via rectangles. Similarly, given the 3-D pose of an object, all that has to be done to get the 2-D image region the object occupies is to project it. That is why an algorithm that solves the 3-D pose tracking task automatically solves the other two tasks as well.

In order to perform 3-D pose tracking, an object model of each object to be tracked is usually required. In this thesis, we always used existing models, created them manually, or used models acquired by a laser scanner. However, it is also possible – up to some extent – to adapt a template model while tracking [41, 268, 152]. Alternatively, one can create object models via stereoscopic segmentation [134], shape-from-X [184, 304], structure from motion [163], or many other approaches.

Independent of the kind of model, 3-D pose tracking must involve some kind of matching between image features and entities on the 3-D object model. These features can be points [1], lines [161, 23], or more complex features such as vertices, T-junctions, cusps, three-tangent junctions, limb and edge injections, or curvature L-junctions [146]. In [67], edge-detection is used to achieve real-time tracking of articulated objects, while Pressigout and Marchand combined two tracking algorithms which use edges and texture information, respectively [210]. In [267], the pose is predicted using pixel displacements and improved afterwards by using image-based cues such as silhouettes.

Learning-based approaches are also common. In [4], Agarwal and Triggs describe learning-based methods that use regression for human pose tracking. Other learning-based approaches

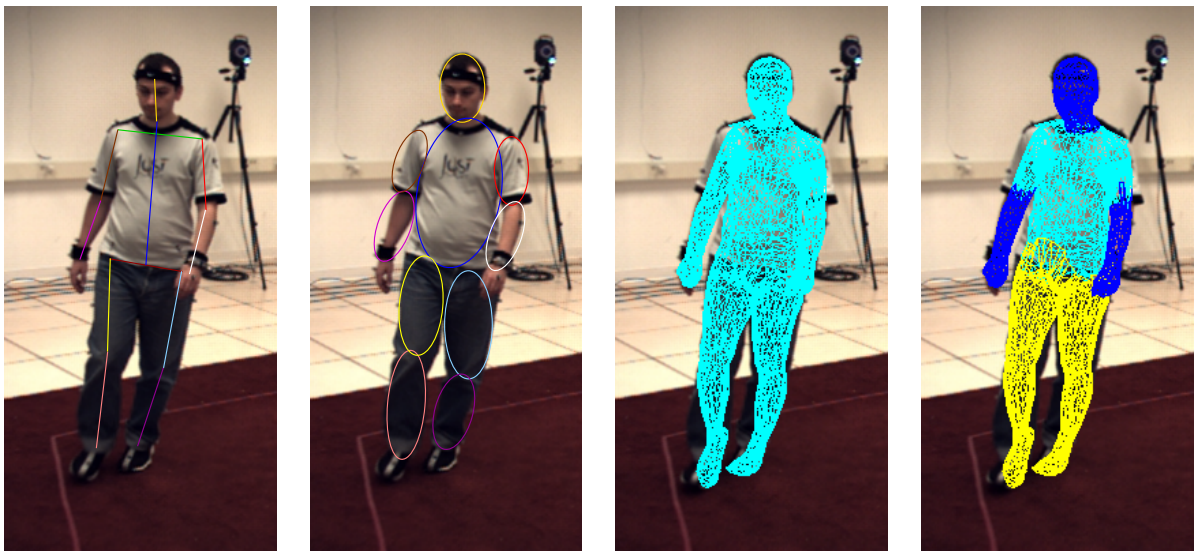


Figure 4.2.: Here, different ways to track objects consisting of kinematic chains are depicted. Shown are, from left to right, tracking using a stick model, tracking with ellipses, tracking using a full 3-D model, and tracking using models consisting of multiple internal regions (see Chapter 7).

have been introduced, e.g. by Taycher et al. [271], in which an undirected conditional random field is used, or by Sminchisescu et al. [254], in which generative models and recognition methods are combined. However, learning-based approaches require a lot of training data, which is not always available. Furthermore, the results are sometimes biased towards the training data.

Another common approach is to track the object by tracking key points using feature trackers such as KLT [162, 275] or by using SIFT features [159]. Although these approaches are usually fast, they often have similar problems. For example, a sufficient number of shared keypoints must be detected in successive images, which is not always possible. Moreover, tracking errors can accumulate and result in an undesired drift. Matching all images to a common frame can prevent this drift, where a *frame* is the set of all images for a certain time, i.e. one image per camera. However, this is not possible in strongly varying sequences.

Drift is avoided in detection-based tracking approaches, which try to detect a specific instance of a given object model in each frame. Since reliable detection is very hard, this problem is usually simplified by using additional assumptions. Some approaches assume the presence of a static background such that the object can be detected using background subtraction, as in [4, 89]. Another common assumption is that enough discriminative features are present on each object to be tracked [192]. It is also possible to learn different object parts from different points of view to detect objects and viewpoints in new images [266].

The approach from [219] first performs segmentation to find the 2-D position of the object, where the object model serves as a shape prior for the segmentation. Afterwards, a 3-D pose tracking approach uses the segmentation results to obtain the 3-D pose. This approach requires a good initialisation in the first frame and limited motion between frames. While

the requirement of an initial pose and limited motion resemble a classical tracking approach, the matching of the surface model to the silhouette in the image avoids drift and involves the silhouette of the object as a general descriptor (see [224]).

Dambreville *et al.* later proposed in [58] to jointly perform segmentation and pose tracking. In [229], this idea is further extended to use a catalogue of 3-D shapes instead of a single object model.

The tracking approach we developed builds upon the approach by Rosenhahn, Brox, and Weickert [219]. Consequently, it uses many of the concepts already present in this approach. Therefore, we describe this approach in the remainder of this chapter such that similarities and differences between these two approaches are clear.

Thus, we first describe the pose estimation approach from [221] and the level-set-based segmentation approach from [34] in Section 4.1 and 4.2, respectively. In Section 4.3, the combination of both approaches yields an energy function for pose tracking of rigid objects [219]. We will then illustrate in Section 4.4 how this energy function can be minimised, and what must be done to extend this approach to kinematic chains (see Sections 2.4.2 and 4.5). Section 4.6 finally shows the problems and limitations of this approach. In the next chapter, we will explain how to overcome these problems using the silhouette-based tracking approach we have developed. Note that we only deal with 3-D pose tracking in this thesis. Thus, the terms "3-D" and "pose" will usually be omitted in from now on.

4.1. Pose Estimation with Image-Vertex Point Correspondences

"The photographic image... is a message without a code."

Roland Barthes (1915 – 1980)

The basic idea behind the pose estimation approach introduced in this section, originally developed by Rosenhahn *et al.*, is to use *2-D–3-D point correspondences*, which are also called *image-vertex point correspondences*. That is, we assume we have a set $\{(\boldsymbol{c}_i, \boldsymbol{C}_i)\}$ of 2-D image points $\boldsymbol{c}_i := (x_i, y_i, k_i)$ lying at position (x_i, y_i) in the k_i-th image for which we know to which 3-D point $\boldsymbol{C}_i$ on an object model they correspond to. For the moment, we assume that the object is rigid. The extended model for kinematic chains is introduced in Section 4.5.

Before giving more details about this approach, let us define the exact task to be solved in this chapter. Since there will be similar, but slightly different tasks in later chapters, we show them in form of a table, as this allows a faster comparisons:

Given:
– One projection matrix P for each camera (see Section 2.6)
– A model of a rigid object (see Section 2.4.1)
– A set of image-vertex point correspondences $(\boldsymbol{c}_i, \boldsymbol{C}_i)$
Assumptions:
None
Task:
– Find the twist $\boldsymbol{\xi}$ describing the 3-D position and orientation of the object

The first step of this approach is to reconstruct the projection rays from the 2-D points $\boldsymbol{c}_i$ of the given correspondences, i.e. the lines passing through the points $\boldsymbol{c}_i$ and the camera origin of the corresponding camera. This results in one line $\boldsymbol{L}_i = (\boldsymbol{n}_i, \boldsymbol{m}_i)$ for each correspondence, given in Plücker form (see Section 2.5.1). Thus, we know that each 3-D point $\boldsymbol{C}_i$ lies on the corresponding Plücker line $\boldsymbol{L}_i$. After transforming the initial 3-D positions by the unknown twist $\boldsymbol{\xi}$, we have

$$\|(\exp(\hat{\boldsymbol{\xi}})\boldsymbol{C}_i)_{3\times 1} \times \boldsymbol{n}_i - \boldsymbol{m}_i\| = 0 \quad \forall\, i. \tag{4.1}$$

Here, the index 3×1 denotes the mapping from an affine to an Euclidean space. That is, the last element is simply removed since it will always be one in this setting.

In practice, the given correspondences are usually inaccurate. Thus, the term on the left hand side of Equation (4.1) is not exactly zero.

As a consequence, a least-squares approximation should be performed here. Instead of searching for the pose $\boldsymbol{\xi}$ such that Equation (4.1) is fulfilled for all points, we are looking for the pose that minimises

$$\sum_i \|(\exp(\hat{\boldsymbol{\xi}})\boldsymbol{C}_i)_{3\times 1} \times \boldsymbol{n}_i - \boldsymbol{m}_i\|^2. \tag{4.2}$$

Although the exponential function can be evaluated efficiently (see Equation (2.G)), the resulting equations are hard to solve. However, after linearising the exponential functions by using a first-order Taylor expansion

$$\exp(\hat{\boldsymbol{\xi}}) = \sum_{j=1}^{\infty} \frac{\hat{\boldsymbol{\xi}}^j}{j!} \approx \boldsymbol{I} + \hat{\boldsymbol{\xi}} \tag{4.3}$$

(where $\boldsymbol{I}$ is the identity matrix), we get quadratic equations whose minimisation results in the equations

$$((\boldsymbol{I} + \hat{\boldsymbol{\xi}})\boldsymbol{C}_i)_{3\times 1} \times \boldsymbol{n}_i - \boldsymbol{m}_i = 0. \tag{4.4}$$

When evaluating the cross product, one can see that this results in three linear equations for each correspondence. Note that the three equations from one correspondence are linearly dependent, though. More precisely, when reordering the equations into the form $\boldsymbol{A}\boldsymbol{\xi} = \boldsymbol{b}$ with $\boldsymbol{A} \in \mathbb{R}^{3\times 3}$ and $\boldsymbol{b} \in \mathbb{R}^3$, $\boldsymbol{A}$ has rank 2. Thus, three accurate correspondences are sufficient

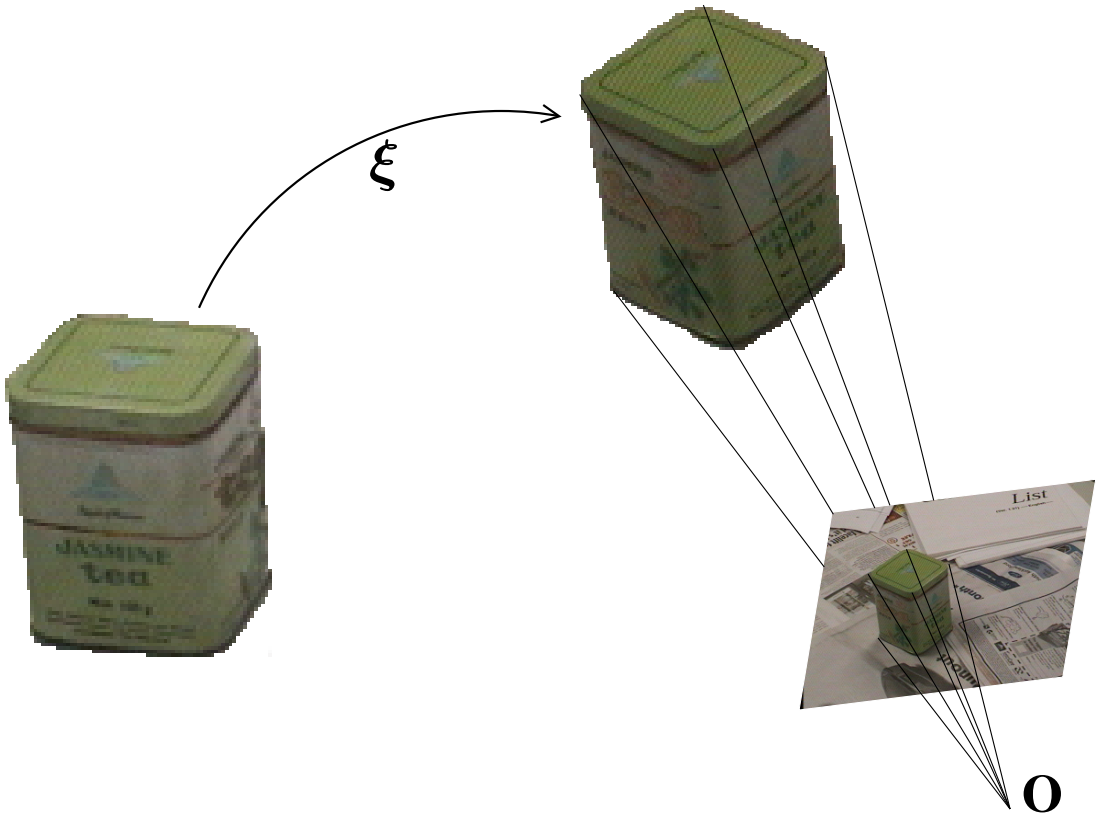


Figure 4.3.: This figure illustrates the goal of 3-D pose tracking with rigid objects: Find the pose and orientation change $\boldsymbol{\xi}$ necessary to move the object to be tracked from its initial position (left) to the position seen in the image.

to find the correct pose $\boldsymbol{\xi}$. Since correspondences can be inaccurate, we use as many correspondences as possible. This results in an overdetermined linear system of equations. Such systems can be solved with standard algorithms. Here, we use the Householder algorithm. Note that if we have a different *confidence* for each point correspondence indicating how accurate this correspondence is, this can be taken into account by scaling each equation in the system accordingly.

We still have to compensate for the error introduced by the linearisation in Equation (4.3). This compensation is done in the following way: First, an initial guess of the pose change $\boldsymbol{\xi}$ is estimated by solving Equation (4.4). Then, this twist is transformed into a rigid body motion by computing $\exp(\hat{\boldsymbol{\xi}})$ using the Rodriguez formula (see Equation (2.G)). The estimated motion is applied to the points $\boldsymbol{C}_i$, effectively setting $\boldsymbol{\xi}$ back to zero. Note that, the larger the pose change $\boldsymbol{\xi}$, the larger the linearisation error will be. However, as the linearisation error in the first step is usually smaller than the estimated pose change, a smaller linearisation error will occur if $\boldsymbol{\xi}$ is estimated again. Thus, we can reduce the linearisation error by iterating the process described in this paragraph. According to our experience, the linearisation error is negligible after three iterations.

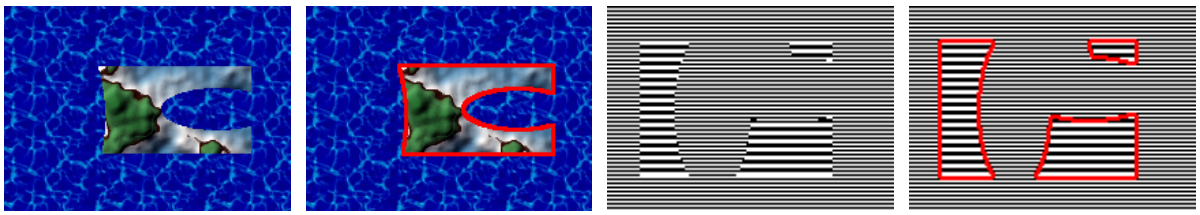


Figure 4.4.: In image segmentation, the task is to split the image into two (or more) regions with different appearance. This figure shows two examples for such a segmentation. In the left images, the two regions differ by colour. However, the average grey value and even its variance are the same in the right images. Thus, texture information is used to distinguish the two regions.

4.2. Segmentation

"The image is more than an idea.
It is a vortex or cluster of fused ideas and is endowed with energy."

Ezra Pound (1885 – 1972)

This section deals with the level-set-based two-phase segmentation method introduced by Brox *et al.* in [34]. Again, we give the task in form of a table:

Given:
– An image with domain $\Omega \subset \mathbb{R}$ and co-domain $\mathbb{R}^n$, $n \geq 1$
Assumptions:
– There are two regions Ω_1 and Ω_2 with the following properties: • The appearance within each region is similar • The appearance between the regions is dissimilar • The boundary between the regions is short
Task:
– Segment the image into the two regions, i.e. find the boundary between the regions

Since we will use segmentation for object-background separation, we only deal with segmentation into two regions. Extensions to multiple regions are possible [305, 35], but not of interest here.

As explained in the task description, the two image regions should have a different appearance. Since this formulation is rather vague, we have to clarify two things: (i) What exactly should be different, and (ii) how do we measure dissimilarity?

The answer to the first question depends on the image to be segmented. For simple grey-valued images, it can be sufficient to demand that the average grey value of the two regions differs as much as possible. For more difficult images, more complicated properties must

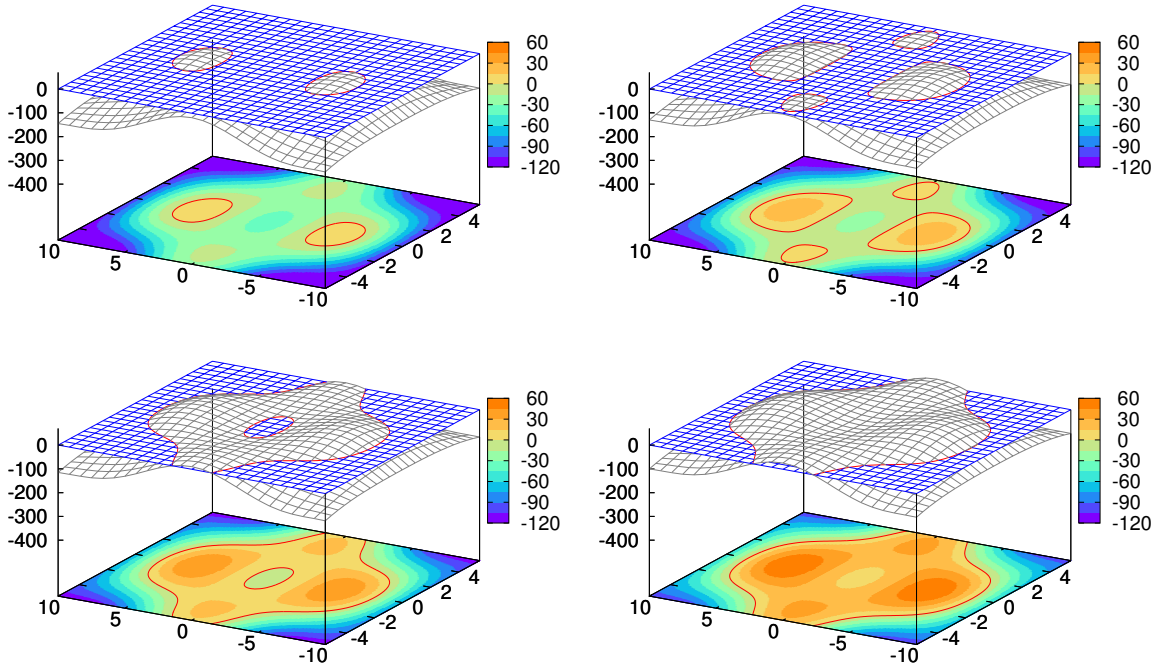


Figure 4.5.: These plots show a level-set function (grey), the zero-level plane (blue), and the intersection of both, i.e. the zero-level line (red), which divides the two regions. All four plots show the same level-set function except for an additive constant. As can be seen, new regions can appear or vanish, several regions can merge, and a region may split into several others. Furthermore, different parts of one region must not be connected, and there can be holes even though the level-set function is continuous.

be compared, see Figure 4.4. Examples for such properties include colour, *texture descriptors* [249, 195, 37], motion [56], or some other filter responses [85, 28]. The only requirement is that the properties used are dense, i.e. they can be computed at every image point. No matter which properties are used, the set of them will be called *image features* (or short *features*).

In order to compare image features between two regions Ω_1 and Ω_2, we use probability density functions (see Definition A.J). More precisely, we assume there are two unknown PDFs p_1 and p_2 describing the appearance of Ω_1 and Ω_2, respectively. Then, we treat all pixels within one region as independent and identically distributed random variables (see Definition A.M) drawn from the corresponding PDF, and estimate them by using one of the methods described in Section 2.7.2. However, since the amount of data is rather small, the approximations of these multi-dimensional PDFs will be rather bad, especially if a lot of features are used. Thus, we will assume independence (see Definition A.L) between the different features here. Then, the PDFs including all M features are given as product of the estimations of the 1-D PDFs p_{ij},

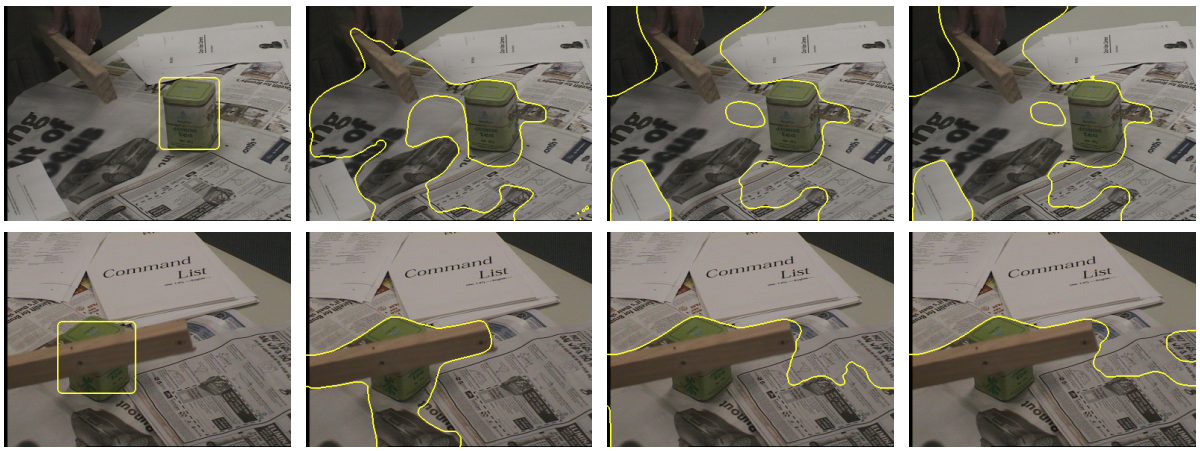


Figure 4.6.: Two views of the same scene in which the tea box is to be segmented. The leftmost image in each row show the initialisations of the level sets used, while the remaining three images show the result of a level-set-based segmentation using a coarse-to-fine (also called multiscale) strategy [34] with 10, 100, and 1000 iterations. As can be easily seen, segmentation can converge to bad results if no prior knowledge about the object to be segmented is used. This is especially true if the object is partially occluded (lower row),

where p_{ij} is an estimation of the PDF of the j-th feature inside Ω_i:

$$p_i = \prod_{j=1}^{M} p_{ij}, \quad i \in \{1, 2\}. \tag{4.5}$$

Although this is an approximation that is not always accurate, the PDFs p_{ij} can be estimated much more accurately than the complete PDFs p_i.

The approach described here uses a level-set function [61, 187] $\Phi : \Omega \to \mathbb{R}$ which splits the image domain Ω into two regions $\Omega_1 = \{\boldsymbol{x} \in \Omega | \Phi(\boldsymbol{x}) > 0\}$ and $\Omega_2 = \{\boldsymbol{x} \in \Omega | \Phi(\boldsymbol{x}) < 0\}$. The zero-level line denotes the boundary between these two regions. The advantage of such an approach is that all kinds of topological changes between the regions can be handled without problems, see Figure 4.5.

Using such a level-set function, the requirements explained in the task description can be modelled in a single energy functional [307]:

$$E(\Phi) = \underbrace{-\int_\Omega \left((H(\Phi)) \log p_1 + (1 - H(\Phi)) \log p_2 \right) \mathrm{d}\boldsymbol{x}}_{\text{feature separation}} + \underbrace{\nu \int_\Omega |\nabla H(\Phi)| \mathrm{d}\boldsymbol{x}}_{\text{Length constraint}}. \tag{4.6}$$

where $\nu \in \mathbb{R}^+$ is a weighting parameter, and H is a function that indicates to which region a point belongs. A typical choice is a regularised *Heaviside function*, i.e. a function $H : \mathbb{R} \to [0, 1]$ with $\lim_{x \to -\infty} H(x) = 0$, $\lim_{x \to \infty} H(x) = 1$, and $H(0) = 0.5$.

The first term demands that the features are separated into two parts with different appearance, and indirectly also states that the features within each region should be equal. The last term penalises long boundaries between the regions.

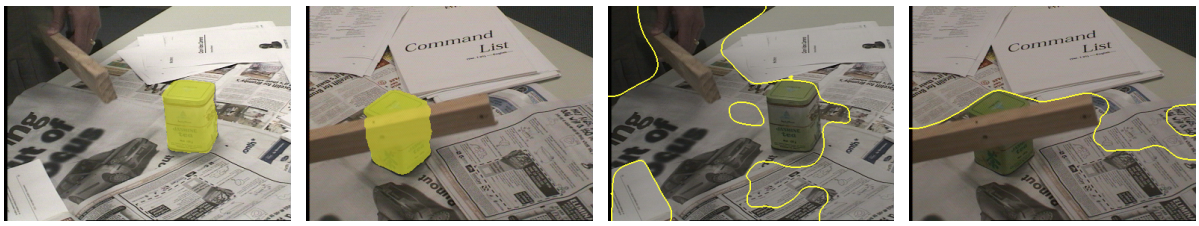


Figure 4.7.: Here, the same two views as in Figure 4.6 have been segmented. The left two images have been segmented using the shape prior from Equation (4.11) while the right images are the results without this prior.

If p_1 and p_2 are known, the energy functional (4.6) can be minimised using the gradient descent equation

$$\partial_t \Phi = H'(\Phi) \left(\log \frac{p_1}{p_2} + \nu \operatorname{div} \left(\frac{\nabla \Phi}{|\nabla \Phi|} \right) \right). \tag{4.7}$$

However, the PDFs are usually not known in advance. As a consequence, E not only depends on the level-set Φ, but also on the two PDFs. In such a situation, one usually estimates p_1 and p_2 using an initialisation of the level set function Φ, then performs minimisation of Equation (4.6) by alternatingly fixing the PDFs and performing the gradient descent given in Equation (4.7), and reestimating the PDFs. This is one specific instance of the *expectation-maximisation principle* [60]. However, this process only converges to a local minimum. Thus, the initialisation is important. Even if a good initialisation is available, and even when using a coarse-to-fine strategy [25, 9], which can diminish such problems, segmentation results are often inaccurate. This is due to the fact that image features are optimised, but the optimised features do not necessarily correspond to the actual objects in the image.

This is illustrated in Figure 4.6: Even if a good initialisation is provided for an unoccluded object, the minimisation in the segmentation algorithm converges to a result that has little to do with the desired solution (see upper row). If there is a partial occlusion as shown in the lower row, segmentation results can even be much worse.

4.3. Combined Energy Functional

"There's an alternative. There's always a third way, and it's not a combination of the other two ways. It's a different way."

David Carradine (1936 – 2009)

In each of the methods described in the last two sections, there is one point not discussed so far. For the pose estimation approach, this open point is how to find the necessary point correspondences. For segmentation, one should find a good initialisation to guide the algorithm as there is no prior knowledge about the appearance of the regions to be segmented.

Both open points are answered when combining these approaches into a silhouette-based tracking approach, as proposed in [33, 219]:

Given:
– One projection matrix P for each camera (see Section 2.6)
– One image sequence with domain $\Omega \subset \mathbb{R}$ for each camera
– A model of a rigid object (see Section 2.4.1)
– An approximate 3-D pose initialisation $\boldsymbol{\xi}_0$ in the first frame
Assumptions:
– The object is always (partly) visible
– The appearance of object and background differs
Task:
– Track the 3-D pose of the object through all images
– Segment the images into object and background regions

The idea was to extend the energy functional for segmentation (4.6) by an additional shape term that depends on the pose of the object. The new energy functional (for one camera) reads

$$E(\Phi, p_{\text{in}}, p_{\text{out}}, \boldsymbol{\xi}) = \underbrace{-\int_{\Omega} ((H(\Phi)) \log p_{\text{in}} + (1 - H(\Phi)) \log p_{\text{out}})\, \mathrm{d}\boldsymbol{x}}_{\text{Feature separation}} + \underbrace{\nu \int_{\Omega} |\boldsymbol{\nabla} H(\Phi)| \mathrm{d}\boldsymbol{x}}_{\text{Boundary length constraint}} + \underbrace{\lambda \int_{\Omega} (\Phi - \Phi_0(\boldsymbol{\xi}))^2 \mathrm{d}\boldsymbol{x}}_{\text{Shape prior}}, \tag{4.8}$$

where λ is a weighting parameter, p_{in} and p_{out} are the PDFs of the object and background region, and Φ_0 is a level-set function obtained in the following way: First, the object model with pose $\boldsymbol{\xi}$ is projected onto the image plane, yielding the set x_s of points occupied by the object. Next, one initialises Φ_0 by setting

$$\tilde{\Phi}_0(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in x_s \\ -1 & \text{else} \end{cases} \tag{4.9}$$

and finally, a signed distance transform [216] is performed. That is, Φ_0 is given as

$$\Phi_0(\boldsymbol{x}) = \begin{cases} \text{dist}(\boldsymbol{x}, Z) & \text{if } \tilde{\Phi}_0(\boldsymbol{x}) > 0 \\ -\text{dist}(\boldsymbol{x}, Z) & \text{else} \end{cases}, \tag{4.10}$$

where Z is the zero-level line of $\tilde{\Phi}_0$. For fast implementations of distance transformations, see [282, 75].

In a setting with C cameras with $C > 1$, a separate level-set function is used for each image, resulting in the energy functional:

$$\begin{aligned} & E(\Phi^1, \ldots, \Phi^C, p_1^1, \ldots, p_1^C, p_2^1, \ldots, p_2^C, \boldsymbol{\xi}) \\ = & \sum_{l=1}^{C} \left[- \int_\Omega \left((H(\Phi^l)) \log p_1^l + (1 - H(\Phi^l)) \log p_2^l \right) \mathrm{d}\boldsymbol{x} \right. \\ & \left. + \nu \int_\Omega |\boldsymbol{\nabla} H(\Phi^l)| \mathrm{d}x + \lambda \int_\Omega (\Phi^l - \Phi_0^l(\xi))^2 \mathrm{d}\boldsymbol{x} \right]. \end{aligned} \tag{4.11}$$

Since the energy functional for multiple-cameras is a straightforward extension of the monocular case, we will describe the setting for a single camera only from now on.

Since the first two terms of the new energy functional are equal to the energy functional used in segmentation, their meaning is obviously the same as in Section 4.2. The shape prior introduced in Equation (4.8) fulfils two tasks: It rewards object poses that match the segmentation obtained, and encourages the segmentation to yield results which are consistent with the projected object model. As can be seen in Figure 4.11, segmentation results drastically improve due to this shape prior. In the next section, we give detail on how this energy functional is minimised.

4.4. Minimisation

"I love sports. I love animals. I love kids. I want to save the world.
So how do I combine all those things? I don't know."

Joan Jett (*1958)

In this section, we explain how to minimise the energy functional given in Equation (4.8). This is a non-trivial task since it depends on a six-dimensional pose vector, an infinite-dimensional level-set and two infinite-dimensional PDFs.

As a global minimum is very hard to find in such a complicated setting, an explicit iterative scheme was proposed in [33]. Note that, in contrast to pose estimation, a pose initialisation is available in pose tracking. This initial pose can be used to initialise the level-set function used in the segmentation step.

Thus, we can initialise Φ with $\Phi_0(\boldsymbol{\xi}_{\text{last}})$, where $\boldsymbol{\xi}_{\text{last}}$ is either the initial pose (first frame), or a pose estimation from the poses in the previous frames. For example, one might set $\boldsymbol{\xi}_{\text{last}}$ as the pose result from the last frame, or a linear extrapolation might be used. Since segmentation results depend on the initialisation, a good initialisation helps to avoid local minima.

As next step, we assume $\boldsymbol{\xi}$ (and thus $\Phi_0(\boldsymbol{\xi})$) is fixed and minimise the segmentation parameters, i.e. the level-set Φ and the PDFs p_{in} and p_{out}. Since this is the same as segmentation with a 2-D shape prior, we can use the gradient descent described in [226, 33], given by

$$\partial_t \Phi = H'(\Phi) \left(\log \frac{p_{\text{in}}}{p_{\text{out}}} + \nu \text{div} \left(\frac{\boldsymbol{\nabla}\Phi}{|\boldsymbol{\nabla}\Phi|} \right) \right) + 2\lambda(\Phi_0(\boldsymbol{\xi}) - \Phi). \tag{4.12}$$

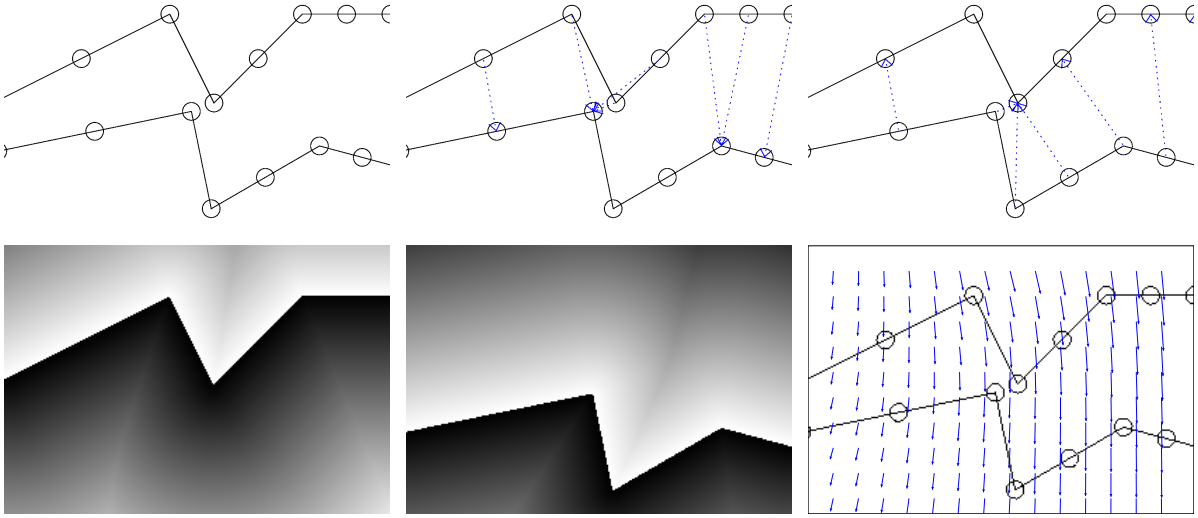


Figure 4.8.: Images illustrating silhouette matching with ICP and the optic flow. **Top left**: A part of the silhouette in two configurations. The silhouette points are marked by circles. **Top middle and top right**: Matching obtained using ICP when matching the first to the second curve or vice versa. **Bottom left and bottom middle**: Distance transform of the two silhouette parts. **Bottom right**: Optic flow between the two silhouettes. The optic flow was scaled down for visualisation.

As in the case of segmentation without shape prior, minimisation of Φ and estimating the PDFs p_{in} and p_{out} is done in turns to improve the results.

Minimisation of Φ with respect to the pose $\boldsymbol{\xi}$ is done by the following steps: First, the level-set function $\Phi_0(\boldsymbol{\xi})$ obtained by projecting the object is matched to the level-set function Φ obtained from segmentation. This can be done, for example, via an *iterated closest point* (*ICP*) algorithm [172, 22], or by using the optic flow [112].

Figure 4.8 illustrates ICP and optic flow for silhouette matching. When using ICP on two curves, the silhouette points on the first curve are assigned to the closest point on the second curve, as demonstrated in the first row of Figure 4.8. Additionally, one can match the second curve to the first to find twice the amount of correspondences. Using these correspondences and a least-squares approach, one approximates the rigid motion that moves the curves over each other. This motion is applied to one of the curves. The whole procedure is iterated until convergence.

When using the optic flow, one first computes the distance transform (see Equation 4.10 and Figure 4.8) of the curves, and computes the optic flow between the two distance transformed images. This yields better results than computing the optic flow directly on images only containing the curves since there are more different features this way. By evaluating the appropriate positions in the flow field, namely those where a silhouette point is, a matching between silhouette points is found. As in the previous case, this matching is used to estimate the rigid motion necessary to match the two curves. After applying the motion to one curve, the whole process is iterated until convergence.

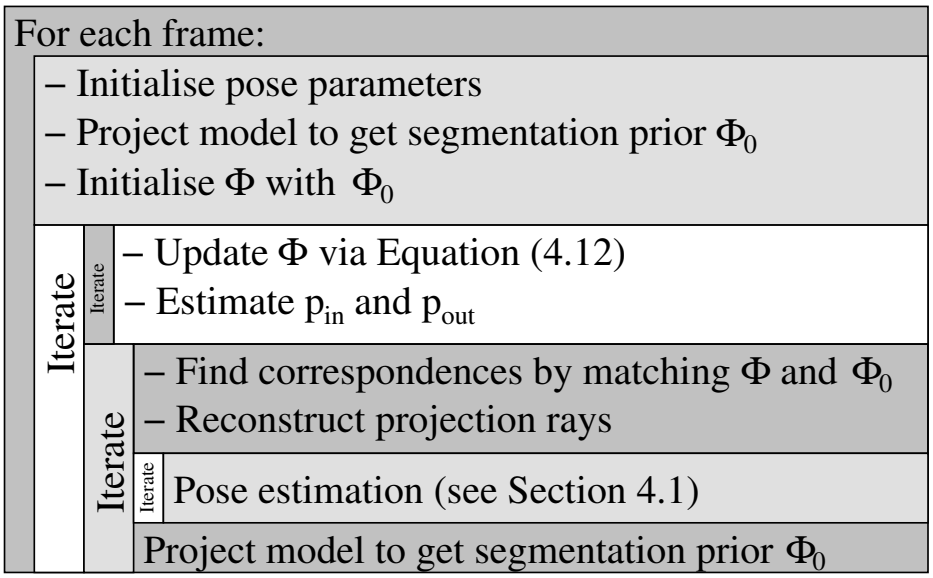


Figure 4.9.: Overview over the steps performed to minimise the energy functional (4.8) (or (4.13)) occurring in the silhouette-based tracking approach by Brox, Rosenhahn, and Weickert [33, 219] explained in Section 4.3.

Since both ICP and the optic flow can be used to estimate correspondences, using both methods together is also possible [217].

The matching step explained above assigns each point on the level-set Φ to a point on $\Phi_0(\boldsymbol{\xi})$. However, since $\Phi_0(\boldsymbol{\xi})$ is just the projection of the object model, we also know which point on $\Phi_0(\boldsymbol{\xi})$ corresponds to which point on the 3-D model. That is, we get correspondences between the 2-D segmentation result Φ and the 3-D model this way. Using these image-vertex point correspondences, we can estimate the pose as described in Section 4.1.

Since we perform a local minimisation, a good initialisation is beneficial to find a good pose. Thus, once the pose was found, the pose in the next image frame is estimated by linear extrapolation from the previous two frames. More advanced approaches such as using the optical flow to predict the new pose [31], or using a physical simulation [285], could also be considered. This extrapolation also helps to reduce problems introduced by linearising the exponential function as done in Equation (4.3).

An overview over the complete minimisation algorithm can be found in Figure 4.9.

4.5. Extension to Kinematic Chains

"It is a point where our old models must be discarded and a new reality rules."

Vernor Vinge (*1944)

The tracking approach illustrated in the last section is designed for rigid objects. In this Section, we present the extension to kinematic chains (see Section 2.4.2) introduced in [220].

Given:
– One projection matrix P for each camera (see Section 2.6)
– One image sequence with domain $\Omega \subset \mathbb{R}$ for each camera
– A model of an object given as kinematic chain (see Section 2.4.2)
– An approximate 3-D pose initialisation $\boldsymbol{\xi}_0$ in the first frame
Assumptions:
– The object is always (partly) visible
– The appearance of object and background differs
Task:
– Track the 3-D pose of the object through all images
– Segment the images into object and background regions

Using kinematic chains instead of rigid objects is straightforward when using the representation of kinematic chains introduced in Section 2.4.2:

$$E(\Phi, p_{\text{in}}, p_{\text{out}}, \boldsymbol{\chi}) = \underbrace{-\int_{\Omega} \left((H(\Phi)) \log p_{\text{in}} + (1 - H(\Phi)) \log p_{\text{out}} \right) \mathrm{d}\boldsymbol{x}}_{\text{Feature separation}} + \underbrace{\nu \int_{\Omega} |\boldsymbol{\nabla} H(\Phi)| \mathrm{d}x}_{\text{Boundary length constraint}} + \underbrace{\lambda \int_{\Omega} (\Phi - \Phi_0(\boldsymbol{\chi}))^2 \mathrm{d}\boldsymbol{x}}_{\text{Shape prior}}. \tag{4.13}$$

Compared with the energy function from Section 4.3 (Equation (4.8)), the 6-D pose vector ξ is replaced by the (6+n)-D pose vector $\boldsymbol{\chi}$ necessary to describe the complete pose of the kinematic chain.

The minimisation with respect to the level-set function Φ is performed in exactly the same way as before. The same is true for the estimation of the PDFs p_{in} and p_{out}, and for the matching between Φ and $\Phi_0(\boldsymbol{\chi})$. For the point transformation needed, e.g. to compute the projection $\Phi_0(\boldsymbol{\chi})$, Equation (2.42) must be used instead of Equation (2.41). In particular, this is of importance when linearising the term that minimises the distance of the transformed point to the reconstructed Plücker line (compare Equation (4.4)):

$$\begin{aligned} &\exp\left(\hat{\boldsymbol{\xi}}\right) \exp\left(\theta_{i_1} \hat{\boldsymbol{\xi}}_{i_1}\right) \ldots \exp\left(\theta_{i_j} \hat{\boldsymbol{\xi}}_{i_j}\right) \boldsymbol{X} \times \boldsymbol{n}_i - \boldsymbol{m}_i \\ \approx &\left(\boldsymbol{I} + \hat{\boldsymbol{\xi}} + \theta_{i_1} \hat{\boldsymbol{\xi}}_{i_1} + \ldots \theta_{i_j} \hat{\boldsymbol{\xi}}_{i_j}\right) \boldsymbol{X} \times \boldsymbol{n}_i - \boldsymbol{m}_i. \end{aligned} \tag{4.14}$$

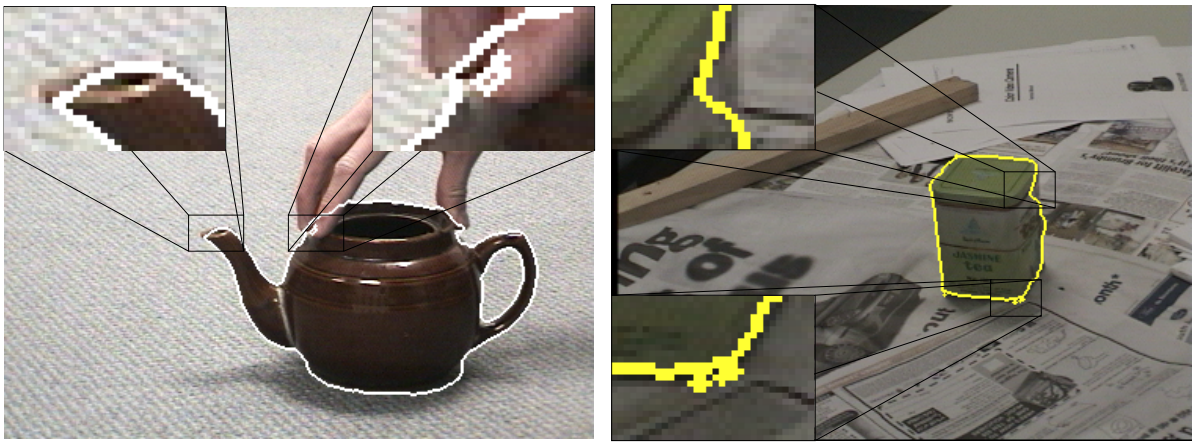


Figure 4.10.: These figures illustrate problems that can occur in the pose tracking approach by Brox, Rosenhahn, and Weickert [33, 219]: **Left**: The segmentation result can be too smooth (due to the boundary length constraint, see left blow-up), and there might be undesirable hole (see right magnification). **Right**: The segmentation can be concave even if convex objects are being tracked (see magnification in the upper left corner), or one might get multiple regions (see blow-up in the lower left corner).

4.6. Drawbacks of this Algorithm

"Mistakes, obviously, show us what needs improving.
Without mistakes, how would we know what we had to work on?"

Peter McWilliams
in *Life 101*

Although the tracking approach described in this chapter works quite well, it has a number of shortcomings, which are illustrated in this section.

The quality of the pose estimation approach presented in Section 4.1 directly depends on the quality of the point correspondences estimated. Thus, an erroneous segmentation leads to bad tracking results. A bad segmentation result can have several reasons:

First of all, the boundary length constraint can lead to oversmoothing in some regions (see left image in Figure 4.10). Reducing the influence of this constraint is possible by reducing the parameter λ, which in turn reduces the risk of oversmoothing. However, a small λ can lead to oversegmentation. That is, undesired holes might appear, or one might get multiple regions as illustrated in the magnifications shown in Figure 4.10. Note that it is not possible to simply fill the holes, as there can be holes in the correct segmentation result, e.g. when tracking a torus, or the tea pot depicted in the left image of that figure.

Furthermore, even when using the shape prior, the segmentation step might return segmentations which are not consistent with the object model. As an example, consider the magnification in the upper left part of the right image in Figure 4.10. One can see that the segmentation

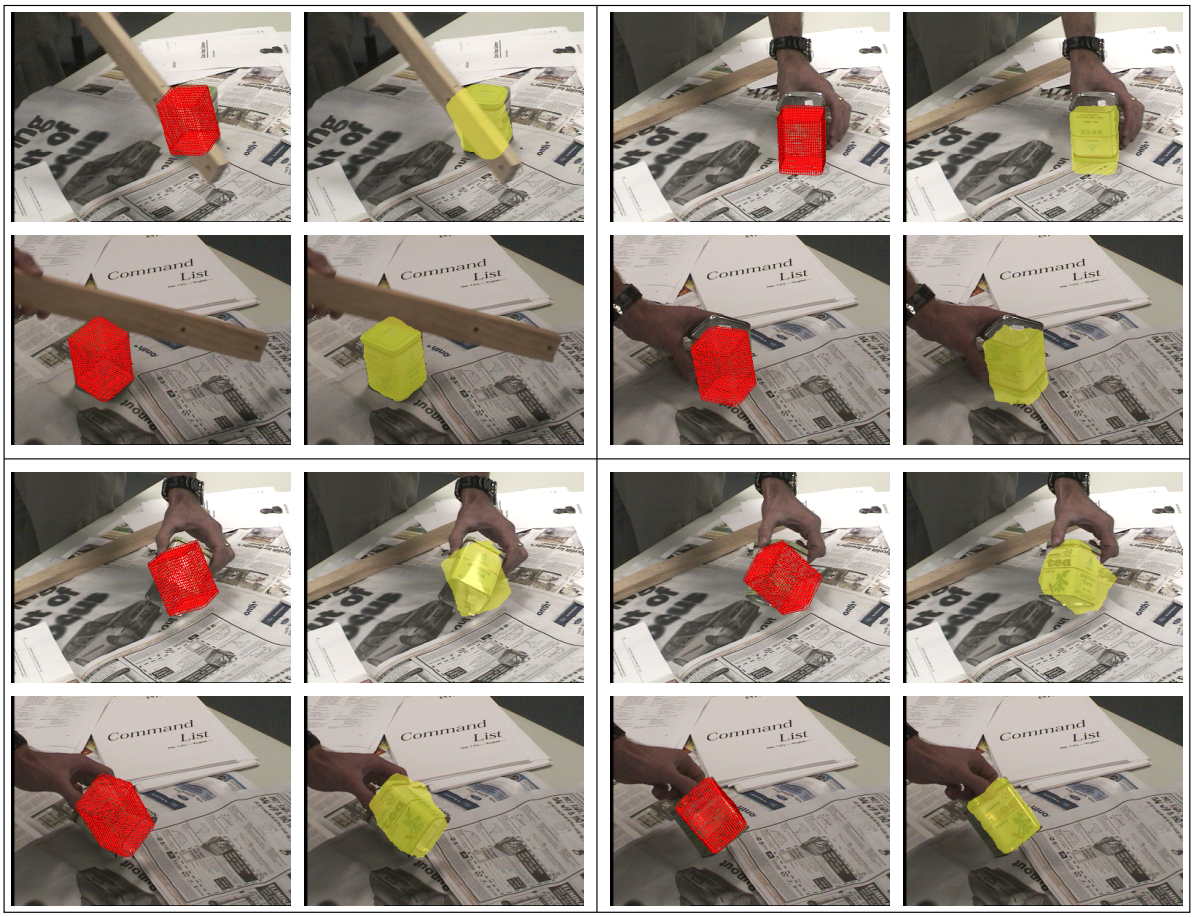


Figure 4.11.: Tracking results (red) and silhouettes (yellow) estimated using the approach described in this chapter [33]. Shown are, from top left to bottom right, the frames 97, 230, 269, and 277.

boundary is concave. Since the 3-D model is convex, this segmentation cannot match to the object.

Figure 4.11 illustrates the problems in a real-world stereo sequence. Although tracking works for some parts of the sequence, there are easily noticeable problems in others: Around frame 97 (upper left corner of Figure 4.11), the segmentation is inaccurate due to an occlusion by a block of wood, resulting in an imprecise tracking result. Around frame 230 (upper right corner), problems occur since the lower surface of the tea box has a completely different appearance than the remainder of the tea box. Around frame 269 (lower left corner), there are specular reflections that look similar to the background, and around frame 277 (lower right corner), the tea box simultaneously rotates around different axes, both of which leads to tracking failures.

In addition to possibly giving wrong results, steps such as segmentation and matching are also quite expensive; the fact that they are merely used as intermediate steps only increases our concern about them. Nevertheless, we have to estimate an infinite-dimensional level set

function several times per frame with this approach even though only a few pose parameters are of interest. In the following chapter, we will see how to prevent this detour by directly coupling the image statistics with the pose parameters without using image segmentation or matching as intermediate steps. As we will see, the problems described above will not be present in our improved approach any more.

5 Segmentation-free Pose Tracking

"A journey of a thousand miles must begin with a single step."

Lao Tzu, Chinese philosopher (604 BC – 531 BC)

The pose tracking approach explained in the last chapter works well in some situations. However, as explained in detail in Section 4.6, it has several problems that can result in a long runtime, inaccurate tracking results, or even a failure to track. Most of these problems are due to the fact that segmentation and matching steps are used as intermediate steps. The tracking approach presented in this chapter will solve these problems by performing a direct estimation of the pose parameters, without using segmentation or matching as intermediate steps. We first described the approach introduced in this chapter in [235], and some extensions in [222] and [223].

We start by introducing the energy function used in our approach, see Section 5.1. The following two sections explain and illustrate how to minimise this new function. Section 5.4 deals with reusing information from the PDFs of the previous frame and other optional extensions. Section 5.5 performs an experimental evaluation, while the chapter is concluded in Section 5.6.

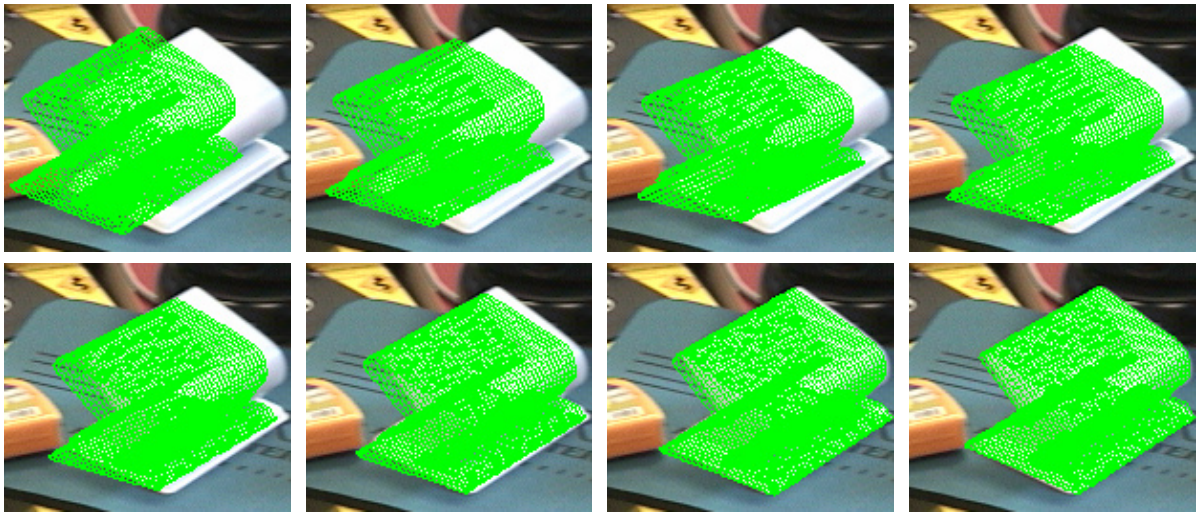


Figure 5.1.: Illustration of the evolution of our 3-D pose tracking algorithm. **From top left to bottom right:** Initial pose estimation obtained from the last frame, and pose estimated after 10, 20, 30, 40, 50, 60, and 70 iterations (magnified). The complete image is shown in Figure 5.3.

5.1. Energy Function for Region-based Pose Tracking

"Every man builds his world in his own image. He has the power to choose, but no power to escape the necessity of choice."

Ayn Rand (1905 – 1982)

The task to be solved in this chapter is the same as in Chapter 4:

Given:
– One projection matrix for each camera (see Section 2.6)
– One image sequence with domain $\Omega \subset \mathbb{R}$ for each camera
– Model of an object given as rigid object or kinematic chain with n joints (see Section 2.4)
– Approximate 3-D pose initialisation $\boldsymbol{\xi}_0 \in \mathbb{R}^6$ or $\boldsymbol{\chi}_0 \in \mathbb{R}^{6+n}$ in the first frame
Assumptions:
– The object is always (partly) visible
– The appearance of object and background differs
Task:
– Track the 3-D pose of the object through all images
– Segment the images into object and background regions

As before, we describe the tracking algorithm only for the single-camera case since this simplifies the formulas. Again, the extension to multiple cameras is straightforward, though. Additionally, we assume that the object model is given as a kinematic chain because rigid objects are in fact a special case of kinematic chains with zero joints.

When looking at the shortcomings of the energy functional given in Equation (4.8), one notices that the problems explained in Section 4.6 are due to the segmentation step: Oversmoothing can happen due to the boundary length constraint needed for the segmentation step, holes and multiple regions appearing in the segmentation can result in wrong correspondences, and both segmentation and matching are computationally expensive.

Thus, we proposed in [235] to remove the explicit contour computation step from the pose tracking algorithm. Instead, we suggested to directly use the projected object to distinguish between object and background region. This results in the energy function:

$$E(\boldsymbol{\chi}, p_{\text{in}}, p_{\text{out}}) = -\int_{\Omega} \left(c_{\boldsymbol{\chi}}(\boldsymbol{x}) \log p_{\text{in}} + (1 - c_{\boldsymbol{\chi}}(\boldsymbol{x})) \log p_{\text{out}}\right) \mathrm{d}\boldsymbol{x}, \tag{5.1}$$

where $\boldsymbol{\chi}$ is the pose of the object, p_{in} and p_{out} are PDFs for the inside and outside of the object region, Ω is the image domain, and $c_{\boldsymbol{\chi}} : \Omega \mapsto \{0, 1\}$ is an *indicator function* of the points occupied in the image plane when projecting the object model with the pose $\boldsymbol{\chi}$:

$$c_{\boldsymbol{\chi}}(\boldsymbol{x}) = \begin{cases} 1 & \text{if the surface of the 3-D model with pose } \boldsymbol{\chi} \\ & \text{projects to the point } \boldsymbol{x} \text{ in the 2-D image plane .} \\ 0 & \text{else} \end{cases} \tag{5.2}$$

In this setting, the object and background regions Ω_{in} and Ω_{out} are given by

$$\Omega_{\text{in}}(\boldsymbol{\chi}) = \{x \in \Omega | c_{\boldsymbol{\chi}}(\boldsymbol{x}) = 1\}, \qquad \Omega_{\text{out}}(\boldsymbol{\chi}) = \{x \in \Omega | c_{\boldsymbol{\chi}}(\boldsymbol{x}) = 0\}. \tag{5.3}$$

When comparing the energy function (5.1) with the energy functional in Equation (4.8), one notes some important differences: First of all, the level-set function is no longer necessary. Thus, instead of estimating an infinite-dimensional energy functio*nal*, we only need to estimate an energy function with $6 + n$ unknowns, where n is the number of joints. Nevertheless, we can still employ all of the dense feature sets used in segmentation. In this thesis, we use either the grey value or the colour in the *CIELAB colour space* [125] as features. For some difficult sequences, we additionally utilise the four channels from the texture space explained in [37].

Strictly speaking, Equation (5.1) is still an energy functional since it depends on the two PDFs p_{in} and p_{out}. As we assume an underlying model for these functions, minimising them is rather easy (see Section 2.7.2), and does not require a gradient descent equation such as the one given in Equation (4.7), though. Thus, we refer to this equation as "energy function". Furthermore, we will reduce this energy functional to a real energy function in Section 5.4.1.

Another difference is that the boundary length constraint is no longer necessary. Thus, the problems due to oversmoothing that could occur in the approach explained in the last chapter are not present in our new approach.

Even more important is that the segmentation obtained by the projection function is, by construction, consistent with the object model. Thus, a segmentation containing too many regions cannot occur. Likewise, there cannot be undesired holes.

If there are holes in the 3-D model, they are still visible in the projected object for some poses. Similarly, if the object consists of several parts which are not connected, there can be multiple regions. Thus, the approach presented in this section still has the advantage that all kinds of topological changes can be modelled.

5.2. Minimisation of the Energy Function

"Mathematics is the most beautiful and most powerful creation of the human spirit."

Stefan Banach (1892 – 1945)

In order to minimise Equation (5.1) we would like to perform a gradient descent. However, this requires the indicator function $c_{\boldsymbol{\chi}}$ to be differentiable. Since $c_{\boldsymbol{\chi}}$ is a binary function this is usually not the case, though.

This problem can be solved by smoothing $c_{\boldsymbol{\chi}}$, e.g. by convolving it with a small Gaussian kernel. In practise, this smoothing is not necessary because $c_{\boldsymbol{\chi}}$ is only evaluated at discrete pixel positions, and discretisation is also a form of regularisation. Nevertheless, we use Sobel operators [96], which compute a smoothed central difference, to approximate $\nabla c_{\boldsymbol{\chi}}$ in our implementation.

Assuming $c_{\boldsymbol{\chi}}$ to be differentiable, the gradient descent of the energy function (5.1) with respect to the pose is given by

$$-\boldsymbol{\nabla}_{\boldsymbol{\chi}} E(\boldsymbol{\chi}, p_{\text{in}}, p_{\text{out}}) = \int_{\Omega} (\boldsymbol{\nabla}_{\boldsymbol{\chi}} c_{\boldsymbol{\chi}}(\boldsymbol{x}) \log p_{\text{in}} - \boldsymbol{\nabla}_{\boldsymbol{\chi}} c_{\boldsymbol{\chi}}(\boldsymbol{x}) \log p_{\text{out}}) + \big(c_{\boldsymbol{\chi}}(\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\chi}} (\log p_{\text{in}}) + (1 - c_{\boldsymbol{\chi}}(\boldsymbol{x})) \boldsymbol{\nabla}_{\boldsymbol{\chi}} (\log p_{\text{out}})\big) \mathrm{d}\boldsymbol{x}. \tag{5.4}$$

Since the estimated PDFs p_{in} and p_{out} change very slowly with varying $\boldsymbol{\chi}$, the two terms $\boldsymbol{\nabla}_{\boldsymbol{\chi}}(\log p_{\text{in}})$ and $\boldsymbol{\nabla}_{\boldsymbol{\chi}}(\log p_{\text{out}})$ are quite small. Thus, and since it is non-trivial to find the change induced into the PDFs by changing the pose, we set them to zero, which results in the approximation

$$-\boldsymbol{\nabla}_{\boldsymbol{\chi}} E(\boldsymbol{\chi}, p_{\text{in}}, p_{\text{out}}) \approx \int_{\Omega} (\boldsymbol{\nabla}_{\boldsymbol{\chi}} c_{\boldsymbol{\chi}}(\boldsymbol{x}) (\log p_{\text{in}} - \log p_{\text{out}})) \, \mathrm{d}\boldsymbol{x}. \tag{5.5}$$

We will further justify this approximation in Section 5.4.1.

If $\boldsymbol{x}$ is not a silhouette point of the projected object, $c_{\boldsymbol{\chi}}(\boldsymbol{x})$ is constant in a small neighbourhood of $\boldsymbol{x}$. Thus, $\boldsymbol{\nabla}_{\boldsymbol{\chi}} c_{\boldsymbol{\chi}}(\boldsymbol{x})$ is zero at such points. Therefore, we can rewrite Equation (5.5) as

$$-\boldsymbol{\nabla}_{\boldsymbol{\chi}} E(\boldsymbol{\chi}, p_{\text{in}}, p_{\text{out}}) \approx \int_{s} (\boldsymbol{\nabla}_{\boldsymbol{\chi}} c_{\boldsymbol{\chi}}(c(s)) (\log p_{\text{in}} - \log p_{\text{out}})) \, \mathrm{d}s, \tag{5.6}$$

where s is the arc-length parametrisation of the 2-D silhouette $c := c(s)$.

When minimising, we are in a discrete setting in which only a finite number of object points lies on the 2-D silhouette c. We will denote the set of these points by O_s. Using this set, Equation (5.6) simplifies to

$$-\boldsymbol{\nabla}_{\boldsymbol{\chi}} E(\boldsymbol{\chi}, p_{\text{in}}, p_{\text{out}}) \approx \sum_{\boldsymbol{x}_s \in O_S} \boldsymbol{\nabla}_{\boldsymbol{\chi}} c_{\boldsymbol{\chi}}(\boldsymbol{x}_s) (\log p_{\text{in}} - \log p_{\text{out}}). \tag{5.7}$$

Intuitively, this equation states that, in order to minimise Equation (5.1), each 2-D silhouette point $\boldsymbol{x}_s$ should move into the direction indicated by

$$\boldsymbol{\nabla}_{\boldsymbol{\chi}} c_{\boldsymbol{\chi}}(\boldsymbol{x}_s) \text{sign}(\log p_{\text{in}} - \log p_{\text{out}}). \tag{5.8}$$

However, the positions of the 2-D silhouette points depend on the pose of the object to be tracked. That is, the points $\boldsymbol{x}_s$ can only be moved indirectly by changing the pose of the object.

Our idea here is to transfer the movement of each 2-D point to the corresponding 3-D point on the object model by using the point-based pose estimation algorithm from Section 4.1. More precisely, we first create a set of image-vertex point correspondences containing all 2-D silhouette points and the corresponding 3-D model points. Then, the 2-D part of each point correspondence is moved along the direction indicated by Equation (5.8). We will denote the 2-D vector that is added to a silhouette point as *force vector*. As a last step, we use the adapted point correspondences to estimate a pose change using the pose estimation algorithm explained in Section 4.1. As a consequence, the 2-D silhouette of the new pose will separate

For each frame:

Extrapolate new pose from previous poses and compute image features

Iterate

- Project 3D object model onto image plane (Section 2.3 and 2.6)
- Generate PDFs for interior/exterior of projected model (Section 2.7.2)
- Adapt 2D–3D point correspondences ($\mathbf{x}_i$,$\mathbf{X}_i$) to ($\mathbf{x}'_i$,$\mathbf{X}_i$) (Section 5.2)
- Construct projection rays from ($\mathbf{x}'_i$,$\mathbf{X}_i$) (Section 2.5.1 and 2.6)
- Generate and solve system of equations to get pose update (Section 4.1)

Figure 5.2.: This figure gives an overview over the steps done to minimise the energy function (5.1) occurring in our basic pose tracking algorithm.

the features in the image more clearly. The whole minimisation procedure is repeated until the pose change is smaller than a threshold $\boldsymbol{T} := (t, r)$. More precise, minimisation is stopped if

$$\boldsymbol{\chi}_i < t \quad \forall\, i \in \{0, 1, 2\}, \qquad \boldsymbol{\chi}_i < r \quad \forall\, i > 3. \tag{5.9}$$

That is, the joint angles and the part of the pose corresponding to the rotation use one threshold r, while another threshold t is used for the translational part of the pose. The whole minimisation process is illustrated in Figure 5.2.

Similar to the minimisation step in the level-set-based segmentation (see Section 4.2), the estimation of the two PDFs p_{in} and p_{out} and the minimisation of the pose parameters is done alternatingly.

As before, we use linear extrapolation to estimate an initial pose guess for the next frame. Even though the initial pose guess is often suboptimal, our minimisation step is usually able to find the correct pose. An experiment with a bad initialisation is discussed in Section 5.5.

The last open question in this minimisation step is how the lengths of the force vectors should be chosen. Equation (5.7) indicates that the length should be $|\log p_{\text{in}} - \log p_{\text{out}}|$, i.e. it should depend on the ratio of p_{in} and p_{out}. After some experiments, we decided to set all force vectors to a fixed length l depending on the sequence, though.

To explain the reasons for this decision, we must distinguish two cases: The appearance change between object and background is either continuous or abrupt. In the first case, the blurring at the object boundaries leads to pixel values not fitting to any of the two PDFs. Using only the sign of the term $|\log p_{\text{in}} - \log p_{\text{out}}|$ leads to a robustification since all silhouette points have the same influence on the movement direction. In the second case, the appearance difference is not a good indicator to decide how much a point should move since the appearance difference at a certain silhouette points stays the same as long as this point does not cross a boundary. Consequently, the appearance difference should not be used to decide how far the silhouette points should move.

Furthermore, due to unmodelled camera aberrations or small errors in the model creation step, it can happen that the model does not exactly fit the foreground region. As a consequence, a part of the silhouette points will be in wrong regions even if the pose is perfect. When not ignoring the term $|\log p_{\text{in}} - \log p_{\text{out}}|$, these points would induce large force vectors, especially in case of abrupt boundaries.

We also tested a multitude of other approaches how to chose the length of the force vectors. Especially, we tried several different adaptive approaches which start at a silhouette point, follow the direction given by Equation (5.8) until the sign of the term $|\log p_{\mathrm{in}} - \log p_{\mathrm{out}}|$ changes, and set the length of the force vectors to this distance. However, all considered variants of this approach, which differ in the way the length of the force vectors and confidence of each point is set, yield inferior results.

The threshold T and the length l that should be used depend on different parameters, e.g. the distance of the object to the camera, the number of silhouette points, or the image resolution used. As in most gradient descents, using a too small l results in a slow convergence. Choosing l too large or T too small leads to oscillations, while a threshold T that is too high results in inaccurate results. As demonstrated in the experiments, there is a quality/speed tradeoff between large and small values of l and T.

Recently, a different gradient descent approach has been proposed for this energy function. In [58], the gradient is computed in 3-D, resulting in several differences: First of all, this approach directly changes the pose instead of using force vectors and reconstruction rays as intermediate steps. Then, there is an additional curvature-dependent multiplicative factor. However, this term is neglected in [58]. Finally, the term $|\log p_{\mathrm{in}} - \log p_{\mathrm{out}}|$ also appearing in their approach is not neglected. Although this makes the optimisation theoretically more sound, we believe that their approach would also benefit from discarding this factor, due to the same reasons explained above.

Later, their approach was extended such that it can handle shape variations by performing a principal component analysis (PCA) and simultaneously estimating pose and shape variation parameters [229]. This approach can also be used in the framework presented in this thesis.

5.3. Illustration of the Minimisation Process

"An algorithm must be seen to be believed."

Donald E. Knuth (*1938)

The minimisation process described in the last section may sound complicated at first, but there is a clear intuition behind it. In this section, we will illustrate the minimisation process.

Consider the example shown in Figure 5.3, in which a puncher is to be tracked. The left image shows the initial image while the projection of the initial pose guess is displayed in blue in the middle image. The 2-D silhouette of this projection is shown in red in the right image.

Our algorithm first creates the PDFs for object and background region. For this example, we used the Parzen model (see Section 2.7.2). The PDFs are shown in red in Figure 5.4.

Next, the minimisation algorithm evaluates the two PDFs at 2-D silhouette points to decide to which PDF these points match better. As an example, let us consider the point $\boldsymbol{x}$ marked by the golden circle in the middle image of Figure 5.3. This point has a colour of $(186, 202, 218)$ in the RGB colour space. Since we use the *CIELAB colour space* [125], this colour must first be converted to that colour space, resulting in the (approximate) colour $(80.6, -2.8, -12.7)$.

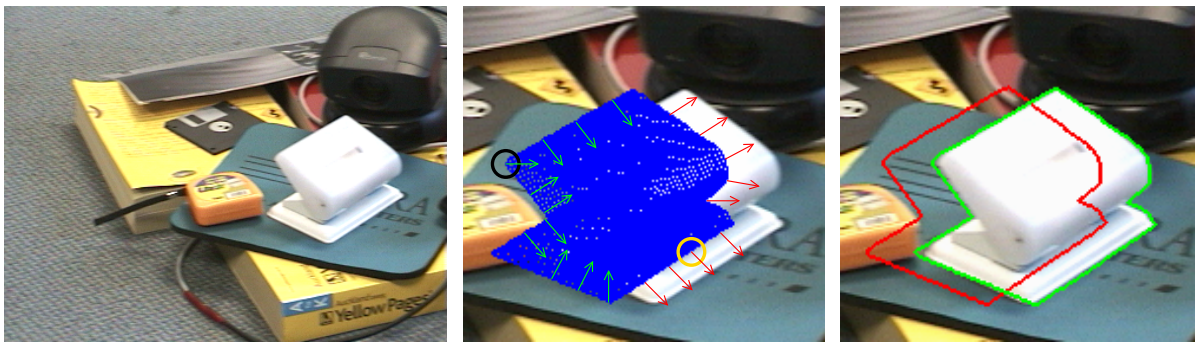


Figure 5.3.: These images illustrate the minimisation process with the region-based pose tracking algorithm presented. **Left**: Initial image. **Middle**: Object in initial pose (blue), and examples in which direction silhouette points should move. The green arrows indicate a movement towards the interior of the object, the red arrows a movement away from the object. The two circles indicate points considered as examples in Section 5.3 and 5.4.1. **Right**: Silhouette in the initial pose (red) and the silhouette after the minimisation step is finished (green). The last two images have been cropped and magnified.

Next, our algorithm checks whether it is "more likely" that this feature appears when drawing randomly from p_{in} than when doing the same from p_{out}. This is the case if $p_{\text{in}}(\boldsymbol{x}) > p_{\text{out}}(\boldsymbol{x})$. Remember that, since we assume independence between the different features, this holds if

$$\prod_{i=1}^{n} p_{\text{in},i}(\boldsymbol{x}) > \prod_{i=1}^{n} p_{\text{out},i}(\boldsymbol{x}), \tag{5.10}$$

where n is the number of features, i.e. three in our example, and $p_{\text{in},i}$ and $p_{\text{out},i}$ are the PDFs for the i-th feature, estimated for the inside and outside region, respectively.

Note that, strictly speaking, the probability that any given feature was drawn from any PDF p is always zero (see Lemma A.K). That is, instead of evaluating p at $\boldsymbol{x}$, one has to integrate a (small) neighbourhood $B(\boldsymbol{x})_\epsilon := \{\boldsymbol{y} : \|\boldsymbol{x}-\boldsymbol{y}\| < \epsilon\}$, $\epsilon > 0$ around the feature to get a positive probability. However, since the PDFs p_{in} and p_{out} are continuous, we know that, if $p_{\text{in}}(\boldsymbol{x}) > p_{\text{out}}(\boldsymbol{x})$ for some $\boldsymbol{x}$, there is a neighbourhood $B(\boldsymbol{x})_\epsilon$ of $\boldsymbol{x}$ such that $p_{\text{in}}(\boldsymbol{y}) > p_{\text{out}}(\boldsymbol{y})$ holds for all $\boldsymbol{y} \in B(x)_\epsilon$. The same arguments holds for $p_{\text{in}}(\boldsymbol{y}) < p_{\text{out}}(\boldsymbol{y})$. Thus, for sufficiently small ϵ, we have

$$p_{\text{in}}(\boldsymbol{x}) \;<\; p_{\text{out}}(\boldsymbol{x}) \quad \Rightarrow \quad \int_{\boldsymbol{y}\in B(\boldsymbol{x})_\epsilon} p_{\text{in}}(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \;>\; \int_{\boldsymbol{y}\in B(\boldsymbol{x})_\epsilon} p_{\text{out}}(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \tag{5.11}$$

and

$$p_{\text{in}}(\boldsymbol{x}) \;>\; p_{\text{out}}(\boldsymbol{x}) \quad \Rightarrow \quad \int_{\boldsymbol{y}\in B(\boldsymbol{x})_\epsilon} p_{\text{in}}(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \;<\; \int_{\boldsymbol{y}\in B(\boldsymbol{x})_\epsilon} p_{\text{out}}(\boldsymbol{y})\mathrm{d}\boldsymbol{y}. \tag{5.12}$$

Therefore, it is sufficient to compare $p_{\text{in}}(\boldsymbol{x})$ and $p_{\text{out}}(\boldsymbol{x})$ to find out to which PDF a feature matches better.

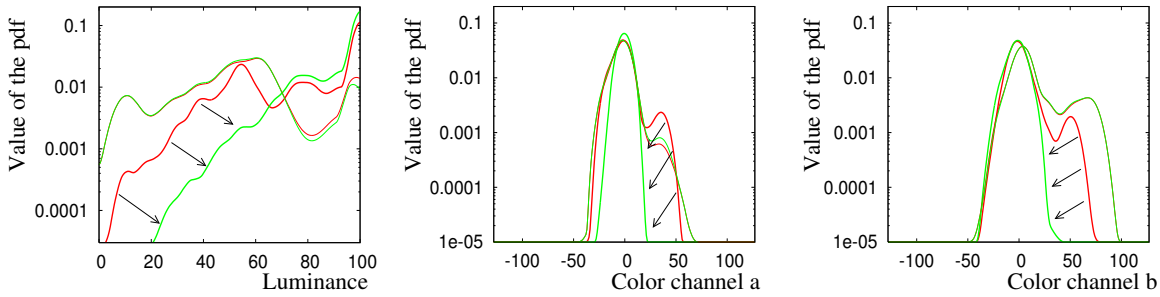


Figure 5.4.: PDFs estimated for the three channels in the puncher image shown in Figure 5.3, i.e. the luminance channel and the colour channels a and b. Shown are the estimated PDFs in the initialisation (red lines) and after minimisation (green lines) for the foreground (thick lines) and the background (thin lines) region. As can be seen, the PDFs are more clearly separated after the minimisation.

For the example point described above, the "a" colour channel indicates that the point belongs to the background region, since $p_{\text{in},2}(-2.8) \approx 0.046 < 0.049 \approx p_{\text{out},2}(-2.8)$ (see Figure 5.4). However, the luminance and "b" colour channel indicate the opposite, since $p_{\text{in},1}(80.6) \approx 0.012 > 0.0017 \approx p_{\text{out},1}(80.6)$ and $p_{\text{in},3}(-12.7) \approx 0.016 > 0.0073 \approx p_{\text{out},3}(-12.7)$. In total, we get

$$p_{\text{in}}(\boldsymbol{x}) = p_{\text{in},1}(\boldsymbol{x})p_{\text{in},2}(\boldsymbol{x})p_{\text{in},3}(\boldsymbol{x}) \quad \approx 0.012 \cdot 0.046 \cdot 0.016 \quad \approx 8.602 \cdot 10^{-6} \tag{5.13}$$

$$p_{\text{out}}(\boldsymbol{x}) = p_{\text{out},1}(\boldsymbol{x})p_{\text{out},2}(\boldsymbol{x})p_{\text{out},3}(\boldsymbol{x}) \quad \approx 0.0017 \cdot 0.049 \cdot 0.0073 \quad \approx 0.58762 \cdot 10^{-6} \tag{5.14}$$

and thus, since $p_{\text{in}}(\boldsymbol{x}) > p_{\text{out}}(\boldsymbol{x})$, the point $\boldsymbol{x}$ seems to belong to the object region.

Note that, since the logarithm is monotonically increasing, we have the following equivalence:

$$\begin{aligned} & p_{\text{in}}(\boldsymbol{x}) && > p_{\text{out}}(\boldsymbol{x}) \\ \Leftrightarrow\ & \log p_{\text{in}}(\boldsymbol{x}) && > \log p_{\text{out}}(\boldsymbol{x}) \\ \Leftrightarrow\ & \log p_{\text{in}}(\boldsymbol{x}) - \log p_{\text{out}}(\boldsymbol{x}) && > 0 \\ \Leftrightarrow\ & \text{sign}(\log p_{\text{in}}(\boldsymbol{x}) - \log p_{\text{out}}(\boldsymbol{x})) && = 1 \end{aligned} \tag{5.15}$$

Thus, Equation (5.8) states that those points which seem to originate from the PDF p_{in} are moved towards the interior of the projected object model. This is indicated by green arrows in the middle image of Figure 5.3. Similarly, those points $\boldsymbol{x}$ where $p_{\text{out}}(\boldsymbol{x}) > p_{\text{in}}(\boldsymbol{x})$ holds are moved in the opposite direction, i.e. away from the object silhouette, as denoted by the red arrows in the same image.

As explained before, the movement of these 2-D points is transferred to the corresponding 3-D point, and this process is iterated. Initially, the object's silhouette contains parts of the orange measuring tape, whose colour is approximately (55,40,55) in CIELAB colour space, and of the cyan mouse pad, which has a colour of (51, -15, -5). Both regions, but especially the first, are well visible in the estimated PDFs for the object region (thick red lines in Figure 5.4).

After tracking is finished, the silhouette mainly includes white image parts. Consequently, the undesired peaks – in the luminance channel around 53, and in the colours channel around 35 and 50, respectively – are gone (see thick green lines in Figure 5.4). Obviously, the PDFs for foreground (thick lines) and background (thin lines) are separated more clearly after pose tracking (green lines) than they have been before (red lines). Note that the PDFs estimated for the background region only changed slightly since only a small percentage of the outside region changed.

5.4. Extensions

"We worked very hard to make extensions very simple."

Mitchell Baker (*1957)

In the following sections, we show several optional extensions that can be used to stabilise tracking in different situations.

5.4.1. Reusing PDFs from the Preceding Frame

The minimisation strategy introduced in the last two sections is local, and can thus result in a local minimum. This problem is especially imminent if the pose is far away from the correct position as this leads to inaccurately estimated PDFs. Using extrapolation (as explained in Sections 4.4 and 5.2) can help to diminish this problem only if the object moves in a predictable way.

To understand this problem, consider the point $\boldsymbol{x}$ marked with a black circle in the middle image of Figure 5.3. Its colour in the CIELAB colour space is $(142.7, 111.4, 123.4)$, and thus, we have (compare red graphs in Figure 5.4):

$$p_{\mathrm{in}}(\boldsymbol{x}) = p_{\mathrm{in},1}(\boldsymbol{x})p_{\mathrm{in},2}(\boldsymbol{x})p_{\mathrm{in},3}(\boldsymbol{x}) \quad \approx 0.021 \cdot 0.013 \cdot 0.043 \quad \approx 1.1942 \cdot 10^{-5}, \tag{5.16}$$

$$p_{\mathrm{out}}(\boldsymbol{x}) = p_{\mathrm{out},1}(\boldsymbol{x})p_{\mathrm{out},2}(\boldsymbol{x})p_{\mathrm{out},3}(\boldsymbol{x}) \quad \approx 0.026 \cdot 0.011 \cdot 0.022 \quad \approx 0.6531 \cdot 10^{-5}. \tag{5.17}$$

That is, since $p_{\mathrm{in}}(\boldsymbol{x}) > p_{\mathrm{out}}(\boldsymbol{x})$, this point fits better to the foreground region, and is thus moved into the wrong direction. This is the case because the percentage of the cyan mouse pad of the total area is larger for the foreground than for the background.

The problem described is due to the fact that the PDFs are inaccurate at the beginning of a new frame. Thus, we propose to refrain from estimating new PDFs before each iteration. Instead, the PDFs estimated at the end of the last frame are used for all computations in the current frame. That is, starting from the second frame, only one PDF for each region and each feature channel is estimated. Since much fewer PDF estimations are necessary this way, this also speeds up the minimisation algorithm.

The assumption behind this approach is that the appearance of fore- and background, i.e. their description by a PDF, changes only slightly each frame. This assumption is well satisfied if the frame rate is high enough to capture the movement adequately.

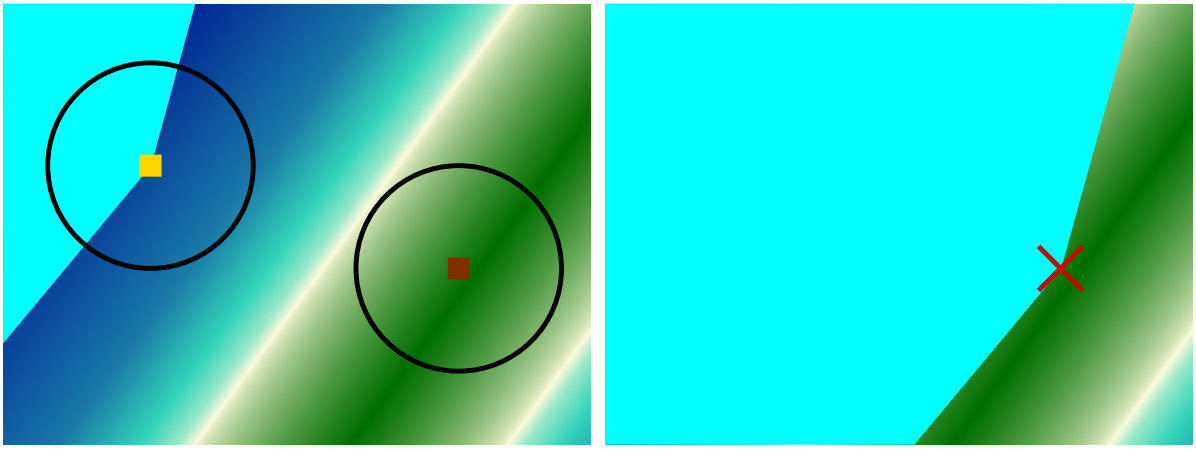


Figure 5.5.: Schematic used to illustrate the possible problems when mixing local PDF estimations and information reusing. In the images, the monochromatic cyan region is the object to be tracked in two consecutive images. The right image shows the image currently being processed, the left image the image considered in the last frame. The black circles indicate the two possible regions in which the local PDF estimations can be performed to decide if the point indicated by the red cross belongs to the foreground region.

There is a third advantage when keeping the PDFs from the previous frame: Since p_{in} and p_{out} are known, only the pose parameters are unknown in the energy function in Equation (5.1). This justifies calling this equation an energy function. Remember that we have approximated the terms $\nabla_{\boldsymbol{\chi}}(\log p_{\text{in}})$ and $\nabla_{\boldsymbol{\chi}}(\log p_{\text{out}})$ with zero in the minimisation of the energy function in Section 5.2. However, when reusing the PDFs from the previous frame, this approximation is not necessary as the derivative of the constants p_{in} and p_{out} are zero anyway.

There is one slight drawback of this approach, though. When estimating the PDFs by a local model, e.g. the local Gaussian model introduced in Section 2.7.2, it is not clear at which point the PDFs should be evaluated. Let $(\boldsymbol{c}, \boldsymbol{C})$ be the image-vertex point correspondence for which we want to check whether it better fits to the object or background region. We can then evaluate the old PDFs either at $\boldsymbol{c}$, or at the position to which $\boldsymbol{C}$ was projected at the end of the last frame.

Figure 5.5 illustrates the advantages and disadvantages of both methods: The point $\boldsymbol{c}$ is marked by a red cross in the right image, i.e. in the current frame, while the points at which the local PDFs are evaluated with the first or second method are shown as brown or golden points, respectively.

The first method ignores the motion of the object, resulting in a bad approximation of the foreground appearance. Even worse, if the object moved very fast, the point $\boldsymbol{c}$ might be so far away from the foreground region of the last frame that only the background region is present in the local regions. That is, the PDF of the foreground region cannot be estimated at all.

With the second method, the estimation of the background region might be completely wrong. In the schematic illustration in Figure 5.5, for example, the background would be estimated as blue while it is in fact green. However, this problem can also occur present in case of a moving background. Also note that this method is not applicable in pose tracking

algorithms that use explicit segmentation without using real 3-D information in the segmentation step. Furthermore, additional time and memory is necessary for this method, albeit the overhead is rather small.

Additionally, it is also possible to estimate the PDF of the background region using the first method, and PDF of the foreground region using the second method. This should combine the advantages of both methods. However, we have not tested this combined approach since the movement between two frames is usually quite small. Thus, both methods yield very similar results in practise. The experiments we show in this thesis use the first method since it is easier to implement and slightly faster.

5.4.2. Prior Knowledge on Probable Joint Configurations

"Learning is not compulsory... neither is survival."

W. Edwards Deming (1900 – 1993)

Especially in monocular sequences, it can happen that a component at the end of a kinematic chain is not visible due to (self-)occlusions. In such a case, there is no knowledge about the corresponding joint angles. Consequently, such angles cannot be estimated at all, even by humans.

However, there is one possible remedy in such a situation, namely to use *prior knowledge* on likely joint configurations. Such priors will be called *angle priors* in this thesis. For example, explicit joint angle limits may be enforced [255, 109]. In this thesis, we use static pose priors [253, 32] for complex sequences, while easy sequences are tracked without any priors. We briefly show how to incorporate pose priors into our tracking approach in this section. For more details, we refer to [32].

The main idea how to incorporate prior knowledge on probable joint configurations is to add additional constraints to the system of equations that is solved to find the pose (compare Equation (4.4)). These additional equations draw the solution towards a state in which the constraints are fulfilled. If the constraints are incorrect, it is still possible to gain a solution that contradicts the constraints, as we are looking for a least-squares solution of an overdetermined system of equations. Thus, the equations derived from the image data can still overrule those from the constraints.

Let us denote the known N joint angle configurations as $\boldsymbol{\Theta}_i, i \in \{1, \ldots, N\}$. In order to obtain likely pose configurations, we first estimate the probability density function $p(\boldsymbol{\Theta})$ of all joint configurations via a Parzen estimation (see Section 2.7.2). As kernel, a Gaussian function is employed. This leads to the estimation

$$p(\boldsymbol{\Theta}) = \frac{1}{\sigma N \sqrt{2\pi}} \sum_{i=1}^{N} \exp\left(-\frac{|\boldsymbol{\Theta}_i - \boldsymbol{\Theta}|^2}{2\sigma^2}\right), \tag{5.18}$$

where the standard deviation σ is chosen as the maximal distance between any nearest neighbours in the training set.

For likely joint configurations, $p(\boldsymbol{\Theta})$ is large. Thus, $-\log(p(\boldsymbol{\Theta}))$ is small for such configurations. Therefore, we add the term

$$E_{\text{Prior}} = -\log(p(\boldsymbol{\Theta})) \tag{5.19}$$

to our energy function (5.1) to incorporate this prior knowledge.

When minimising the new energy function, the additional term $\boldsymbol{\nabla}_{\boldsymbol{\chi}} E_{\text{Prior}}$ appears in the modified gradient descent. As E_{Prior} only depends on the joint angles, this term is zero for the rigid transformation of the complete object. For the joint angles, one obtains the gradient

$$\boldsymbol{\nabla}_{\boldsymbol{\Theta}} E_{\text{Prior}} = \frac{\sum_{i=1}^{N} w_i(\boldsymbol{\Theta}_i - \boldsymbol{\Theta})}{\sigma^2 \sum_{i=1}^{N} w_i}, \tag{5.20}$$

where the weights w_i are given by

$$w_i := \exp\left(-\frac{|\boldsymbol{\Theta}_i - \boldsymbol{\Theta}|^2}{2\sigma^2}\right). \tag{5.21}$$

Intuitively, this draws the angles of the current pose to nearby prior joint configurations, i.e. to the next local maximum of the PDF. This is accounted for by adding an additional equation

$$\theta_j^{t+1} = \theta_j^t + \tau \partial_j \theta_j^t \tag{5.22}$$

for each joint angle j to the linear system obtained in the minimisation step. Here, the superscript t denotes time, i.e. the frame number, and τ is a time step parameter always set to $\frac{1}{8}\sigma^2$. In order to obtain an appropriate and constant weighting between prior and image data, these equations are multiplied by the total number of image-vertex correspondences divided by eight.

There are more advanced methods that might be used instead. For example, an anisotropic kernel might be utilised for estimating the PDF, or motion dynamics might be employed, see [30, 248, 281]. As motion priors always bias tracking results towards the training data, we tried to avoid such priors as much as possible, and did not invest much time into improving them.

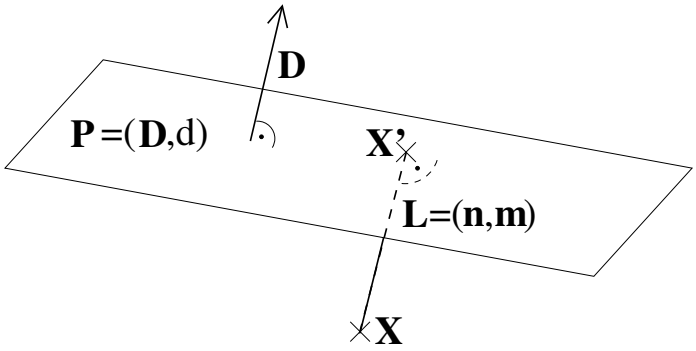
5.4.3. Positional Constraints

The constraints introduced in this section model further prior knowledge about the scene that cannot be handled with the approach from the last section. Here, we discuss for a couple of situations how to integrate this knowledge into our tracking approach. The constraints explained here were first published in [222] and [223].

3-D–3-D Point Correspondences

As first example, consider the cyclist shown in Figure 5.6(a), whose model consists of a human rigidly attached to the pedals of a bike. The left foot is in the same rigid object as both pedals. However, if the right foot were also connected to the pedals, we would obtain a cyclic

(a)



(b)

Figure 5.6.: **Right**: A cyclists for which various constraints can be used. **Right**: This figure illustrates the projection of the point $\boldsymbol{X}$ onto the plane $\boldsymbol{P} = (\boldsymbol{D}, d)$ by using the Plücker line $\boldsymbol{L} = (\boldsymbol{n}, \boldsymbol{m}) = (\boldsymbol{D}, \boldsymbol{X} \times \boldsymbol{D})$. The projected point is called $\boldsymbol{X}'$.

dependency, which is impossible in kinematic chains. For this sequence, we want to exploit several facts: The hands are always at the handle of the bicycle, the pedals have to rotate around a fixed axis, and the right foot should stay on the right pedal. The first two constraints can be incorporated by demanding that some vertices of the model are at a fixed position. For the latter constraint, we demand that some vertices are at the same positions as other vertices lying in different rigid parts of the kinematic chain.

For each of these constraints, we have equations of the form

$$\mathbf{0} = \boldsymbol{P_i} - \boldsymbol{P_i'}, \tag{5.23}$$

where $\boldsymbol{P_i}$ are vertices of the model, and where each $\boldsymbol{P_i'}$ denotes either a fixed point or a vertex of the model. Next, we express the vertices as

$$\boldsymbol{P_i} = \exp\left(\hat{\boldsymbol{\xi}}\right) \exp\left(\theta_{i_1}\hat{\boldsymbol{\xi}}_{i_1}\right) \exp\left(\theta_{i_2}\hat{\boldsymbol{\xi}}_{i_2}\right) \ldots \exp\left(\theta_{i_j}\hat{\boldsymbol{\xi}}_{i_j}\right) \boldsymbol{X}_i, \tag{5.24}$$

where the indices i_j denote the rigid body parts from the root of the kinematic chain to the rigid part on which $\boldsymbol{P_i}$ lies, as explained in Equation (2.42).

Finally, these equations are linearised in the same way as before, resulting in either

$$\left(\boldsymbol{I} + \hat{\boldsymbol{\xi}} + \theta_{i_1}\hat{\boldsymbol{\xi}}_{i_1} + \ldots \theta_{i_j}\hat{\boldsymbol{\xi}}_{i_j}\right) \boldsymbol{X}_i - \boldsymbol{P_i'} = \mathbf{0}, \tag{5.25}$$

if $\boldsymbol{P_i'}$ is a fixed point, or in

$$\left(\boldsymbol{I} + \hat{\boldsymbol{\xi}} + \theta_{i_1}\hat{\boldsymbol{\xi}}_{i_1} + \ldots \theta_{i_j}\hat{\boldsymbol{\xi}}_{i_j}\right) \boldsymbol{X}_i - \left(\boldsymbol{I} + \hat{\boldsymbol{\xi}} + \theta_{i'_1}\hat{\boldsymbol{\xi}}_{i'_1} + \ldots \theta_{i'_j}\hat{\boldsymbol{\xi}}_{i'_j}\right) \boldsymbol{X'}_{i'} = \mathbf{0}, \tag{5.26}$$

if both points are vertices of the model.

In both cases, we get additional linear equations which are added to the system of linear equations obtained before. After solving the system of equations, the resulting pose will approximately fulfil the constraints. The stronger these new equations are weighted, the stronger deviations from the desired constraints are penalised. Tracking results using these constraints will be shown in Figure 6.11 in Section 6.4.

Floor constraints

As a second example, consider a human moving on a flat floor. Although no part of the human can pass through the floor in reality, our current tracking approach can produce results where this is the case.

To include the prior knowledge that the floor is impenetrable, we first search all vertices lying below the floor in the current pose. To find these vertices, we model the floor as a 3-D plane $\boldsymbol{P} = (\boldsymbol{D}, d)$ in Hessian normal form and compute the sign of $\boldsymbol{x} \cdot \boldsymbol{D} + d$ for each vertex $\boldsymbol{x}$ to find out which points lie below the floor (see Section 2.5.2).

For each point lying below the floor, we compute the point X' on the plane $\boldsymbol{P}$ that is closest to $\boldsymbol{X}$. This point is found as intersection between the plane $\boldsymbol{P}$ and the Plücker line $\boldsymbol{L} = (\boldsymbol{n}, \boldsymbol{m}) = (\boldsymbol{D}, \boldsymbol{X} \times \boldsymbol{D})$ passing through $\boldsymbol{X}$ and parallel to the normal of the plane, see Figure 5.6(b). The intersection point between a plane and a line can in general be computed as

$$\boldsymbol{X}' = \frac{\boldsymbol{D} \times \boldsymbol{m} - \boldsymbol{n}d}{\boldsymbol{D}\boldsymbol{n}} \tag{5.27}$$

Here, $\boldsymbol{D}$ is equal to $\boldsymbol{n}$ and thus, $\boldsymbol{D}\boldsymbol{n} = 1$ holds due to normalisation. Using *Lagrange's formula*,

$$\boldsymbol{a} \times (\boldsymbol{b} \times \boldsymbol{c}) = \boldsymbol{b}(\boldsymbol{a}\boldsymbol{c}) - \boldsymbol{c}(\boldsymbol{a}\boldsymbol{b}), \tag{5.28}$$

Equation (5.27) can be simplified in the following way:

$$\begin{aligned} \boldsymbol{X}' =& \frac{\boldsymbol{D} \times \boldsymbol{m} - \boldsymbol{n}d}{\boldsymbol{D}\boldsymbol{n}} \\ =& \boldsymbol{D} \times (\boldsymbol{X} \times \boldsymbol{D}) - \boldsymbol{D}d \\ =& \boldsymbol{X}(\boldsymbol{D}\boldsymbol{D}) - \boldsymbol{D}(\boldsymbol{D}\boldsymbol{X}) - \boldsymbol{D}d \\ =& \boldsymbol{X} - \boldsymbol{D}(\boldsymbol{X}\boldsymbol{D} + d) \end{aligned} \tag{5.29}$$

That is, the intersection point can be found by moving from the point $\boldsymbol{X}$ into the normal direction of the plane. The length and sign of this movement is determined by the signed distance of the point $\boldsymbol{X}$ to the plane $\boldsymbol{P}$, i.e. by $\boldsymbol{X}\boldsymbol{D} + d$, see Section 2.5.2.

For each point $\boldsymbol{X}_i$ below the ground plane, we obtain an equation of the form

$$\boldsymbol{0} = \boldsymbol{X}_i - \boldsymbol{X}_i', \tag{5.30}$$

where $\boldsymbol{X}_i'$ is the projection of the point $\boldsymbol{X}_i$ onto the plane. As before, $\boldsymbol{X}_i$ is expressed with respect to the pose and the joint angles:

$$\exp\left(\hat{\boldsymbol{\xi}}\right) \exp\left(\theta_{i_1}\hat{\boldsymbol{\xi}}_{i_1}\right) \ldots \exp\left(\theta_{i_j}\hat{\boldsymbol{\xi}}_{i_j}\right) \boldsymbol{X}_i - \boldsymbol{X}_i' = \boldsymbol{0}. \tag{5.31}$$

After linearising this equation, one obtains

$$\left(\boldsymbol{I} + \hat{\boldsymbol{\xi}} + \theta_{i_1}\hat{\boldsymbol{\xi}}_{i_1} + \ldots \theta_{i_j}\hat{\boldsymbol{\xi}}_{i_j}\right) \boldsymbol{X}_i - \boldsymbol{X}_i' = \boldsymbol{0}. \tag{5.32}$$

As before, these linear equations are added to the system of equations obtained in the minimisation step. We use this constraint in several experiments with the HumanEva-II benchmark, e.g. in Figure 5.19 in Section 5.5.

5.4.4. Jitter Prevention

Even when using prior knowledge to handle partial occlusions and model inaccuracies, tracking results can still be jittery. Thus, many approaches in the literature have proposed to smooth the obtained tracking results as a post-processing step [49, 262, 279]. In [7], an approach to camera motion smoothing by minimising an energy functional is proposed. In the following, we will extend our energy function such that it performs online smoothing while tracking the sequence, as described in [218].

To prevent motion jitter, we first predict the object's pose at the beginning of a new frame $\boldsymbol{\chi}_e = (\boldsymbol{\xi}_e, \boldsymbol{\Theta}_e)$ from the poses estimated in the last two frames $\boldsymbol{\chi}^t$ and $\boldsymbol{\chi}^{t-1}$. For the joint angles this is done via the following formula:

$$\boldsymbol{\Theta}_e = \boldsymbol{\Theta}^t + \partial\boldsymbol{\Theta}^t \approx \boldsymbol{\Theta}^t + (\boldsymbol{\Theta}^t - \boldsymbol{\Theta}^{t-1}). \tag{5.33}$$

That is, the change between the last two frames is added to the last known joint configuration.

Similarly, we obtain an estimation for the rigid transformation of the complete object by first computing the motion between the last two frames, and "adding" this motion to the last known rigid body motion. As we use twists to represent these transformations, we first convert them to elements of the Lie group $SE(3)$ via the exponential function (see Section 2.3.1), compute the corresponding motion, and convert this motion back into a twist via the inverse logarithm. This results in the prediction:

$$\boldsymbol{\xi}_e = \log\Big(\exp(\hat{\boldsymbol{\xi}}^t)\exp(\hat{\boldsymbol{\xi}}^{t-1})\exp(\hat{\boldsymbol{\xi}}^t)\Big). \tag{5.34}$$

Now, we can compute the deviation of the current pose from the predicted pose as

$$E_{\text{Smooth}} = \left|\log\left(\exp(\widehat{\boldsymbol{\xi}_e})\exp(\hat{\boldsymbol{\xi}}^{t+1})^{-1}\right)\right|^2 + |\boldsymbol{\Theta}_e - \boldsymbol{\Theta}^{t+1}|^2. \tag{5.35}$$

This term is added to our energy function to penalise poses that deviate from the prediction.

When minimising the new energy function, this results in additional linear equations, which are added to the system of equations to be solved. These equations have the form

$$\nabla_{\boldsymbol{\chi}} E_{\text{Smooth}} = \left(\log\left(\exp(\widehat{\boldsymbol{\xi}_e})\exp(\hat{\boldsymbol{\xi}}^{t+1})^{-1}\right), \boldsymbol{\Theta}_e - \boldsymbol{\Theta}^{t+1}\right)^\top = \mathbf{0} \tag{5.36}$$

and draw the estimation of the next pose towards the predicted pose, which smoothes the tracking results. Moreover, this can also stabilise the tracker: Large deviations from the prediction, which are often due to tracking failures, can be prevented. Moreover, these additional constraints prevent singular equation systems, which can occur if parts of a kinematic chain are not visible.

5.5. Experiments

"Life is too short not to experiment."

Jamelia (Niela Davis) (*1981)

We performed several experiments to illustrate the abilities of the basic tracking approach introduced in this chapter. We start by showing tracking results for rigid objects, followed by tracking results for kinematic chains.

5.5.1. Rigid Objects

The first tracking result obtained with the pose tracking approach presented in this chapter is shown in Figure 5.7. It demonstrates that, despite the problems explained in the last section, our tracking algorithm also works very well without reusing PDFs from the preceding frame. Here, the pose of a tea box was to be tracked in a stereo setup. Figure 5.7 shows tracking results for some particularly difficult frames of the sequence. In the first frames, the tea box does not move while simultaneous occlusions occur in both views. Later, the tea box is turned upside down, and specular highlights occur, while it is turned around two axes simultaneously. Nevertheless, the tracker was able to track the tea box throughout the complete sequence, while the segmentation-based approach explained in the last chapter has various problems with this sequence (see Figure 4.11).

As mentioned before, there is a quality/speed tradeoff between different choices of l (length of the force vectors) and T (threshold of the pose change below which the minimisation is stopped). In Figure 5.8, we use the stereo tea box sequence to illustrate this tradeoff. The results shown in the upper row use $l = 6$, a threshold of 1 for the translational part and 0.01 for the rotational part of the twist, i.e. $T = (1, 0.01)$. In the lower row, the length of the force vectors and the used thresholds were smaller ($l = 2$, $T = (0.1, 0.001)$). Furthermore, only the CIELAB channels were used as features in the upper row, while both CIELAB channels and the four channels from the texture space from [37] were used to create the results shown in the lower row. We will refer to these sets of parameters as set "A" (upper row, fast, inaccurate) and set "B" (lower row, slow, precise).

Although our algorithm was able to track the whole sequence with both parameter sets, there are large differences in quality and speed: Using parameter set "A", tracking results are less accurate, as can be seen in Figure 5.8. Especially note the strong fluctuations during the first 160 frames when using parameter set "A". The only notable fluctuations that can be seen when using parameter set "B" are around frames 100 and 120, in which a significant part of the tea box is occluded. However, tracking was performed over all 395 frames in three minutes and 13 seconds (approximately two frames per second), while more than 28 minutes was necessary using parameter set "B" (less than one frame every four seconds). Due to the quality/speed tradeoff, it is difficult to give exact running times of our algorithm.

To further understand the problem, one should note that the time required in a certain frame depends on the number of iterations necessary to track the object. Figure 5.9 shows a com-

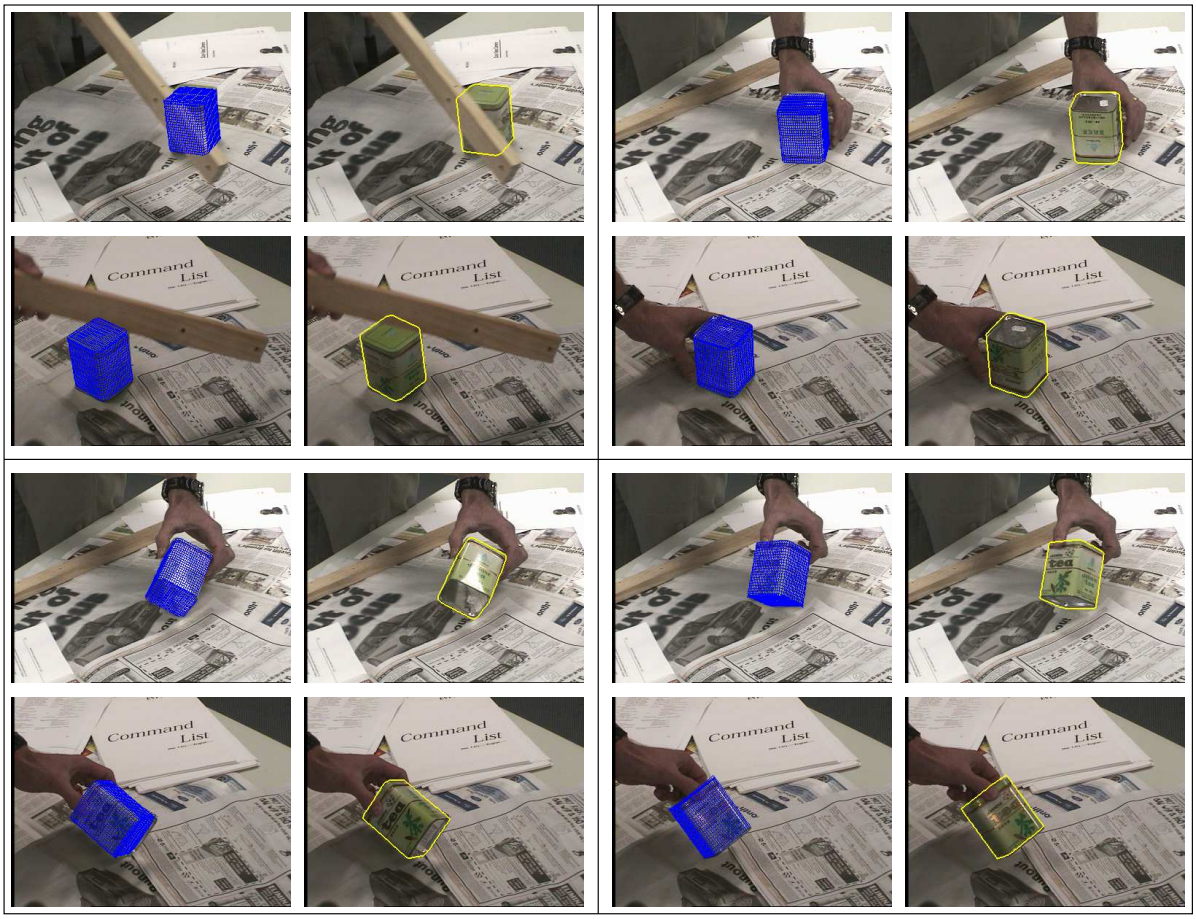


Figure 5.7.: Each block shows the silhouette (yellow) and pose (blue) estimated for both views of a stereo sequence in which a tea box was tracked. Despite partial occlusions (frame 97, top left), complete rotations (frame 230, top right), specular reflections (frame 269, bottom left) and simultaneous rotations around different axes (frame 277, bottom right), tracking was successful.

parison of the time and iterations required per frame when tracking the tea box using the two parameter sets mentioned above.

As can be clearly seen in the graphs shown in Figure 5.9, there are two main reasons why parameter set “B” is much slower than set “A”. First of all, the preprocessing step takes much longer since computing the texture feature space is very expensive. Furthermore, much more iterations are necessary until the movement of the object is below the requested threshold. It can also be seen that more iterations are necessary in some frames. This is due to unexpected movements, or occlusions.

Figure 5.10 shows the first example for which PDFs from the previous frame are reused (see Section 5.4.1). In this scene, a wooden toy giraffe is tracked in a monocular sequence. Due to the cluttered background and since only one view is available, this scene is quite challenging. Nevertheless, our tracking approach was able to estimate good poses.

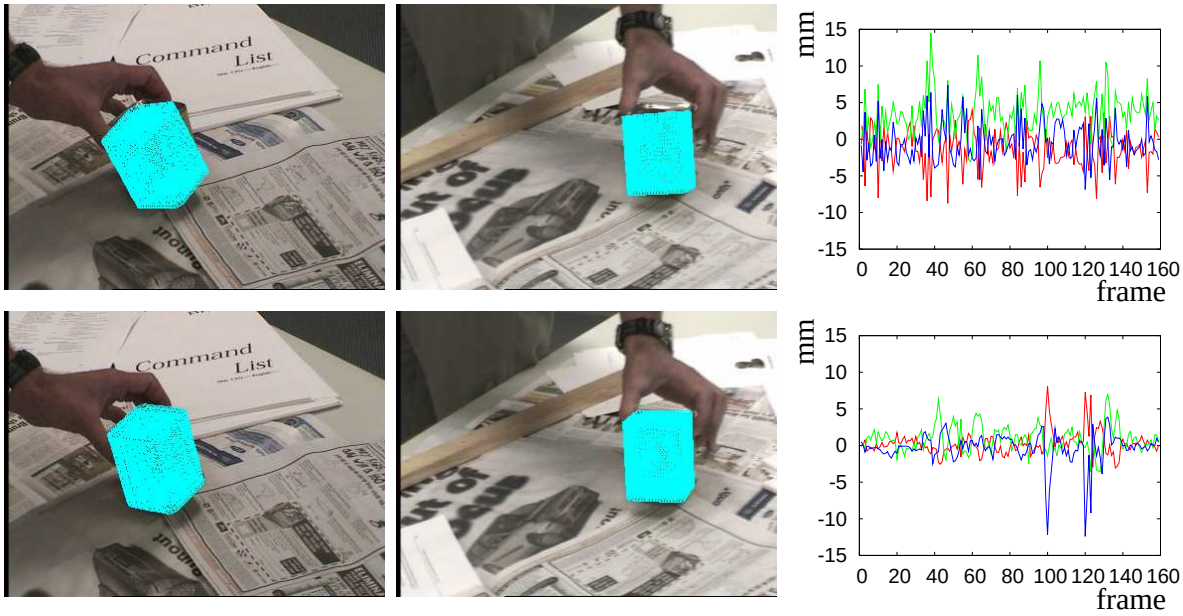


Figure 5.8.: This figure illustrates the quality/speed tradeoff of our pose tracking algorithm. **Upper row**: Inaccurate but fast results. **Lower row**: More precise results that take more than ten times as long. The two images in each row show the tracking results in frame 220 as seen from both cameras, while the plot shows the change in the translational part of the pose in the first 160 frames. Since the tea box did not move in the first 160 frames, no movement would be optimal.

As another example, we show tracking results for a stereo sequence with a tea pot in Figure 5.11. This sequence is not too difficult to track: Despite changes in the lighting conditions, fore- and background can be separated very easily. The only difficulty is the fact that the object model does not perfectly match the object seen in the image. This only causes very slight problems, though.

Figure 5.12 shows tracking results for the puncher sequence already shown in Figure 5.3. As can be seen, there are a multitude of challenges that must be addressed, ranging from large displacements over bad image quality to the fact that only a single view is available. However, as the puncher is rather easy to distinguish from the background, this is not a big problem for our pose tracking algorithm.

To evaluate our tracking approach quantitatively, we also checked our tracking results in a synthetic sequence. The advantage of such a sequence is that the ground truth is available, i.e. it is possible to compare the result of the tracking algorithm against the real position of the object to be tracked. The sequence used here is a monocular sequence with a rolling aeroplane, which was generated and kindly provided by Richard Steffen. As can be seen in Figure 5.13, the aeroplane passes another (static) aeroplane with a similar appearance, making tracking nontrivial.

Considering the translational error (lower right plot in Figure 5.13), one can see that errors in this sequence appear due to two different reasons: At the beginning of the sequence, the

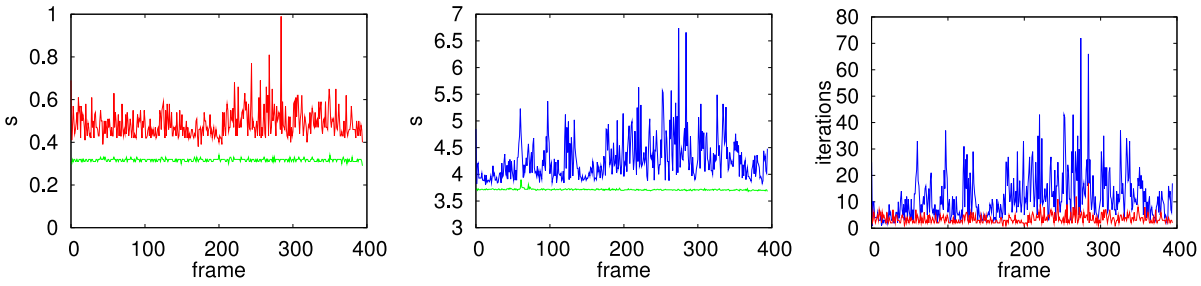


Figure 5.9.: **Left**: Time required per frame when tracking the tea box shown in Figure 5.8 using the fast parameter set "A" (red). The green line corresponds to the time required for preprocessing. **Middle**: The time required per frame using parameter set "B" is shown in blue, while the green line again shows the time required for preprocessing. All times are given in seconds. **Right**: Iterations done per frame using parameter set "A" (red) or parameter set "B" (blue), respectively.

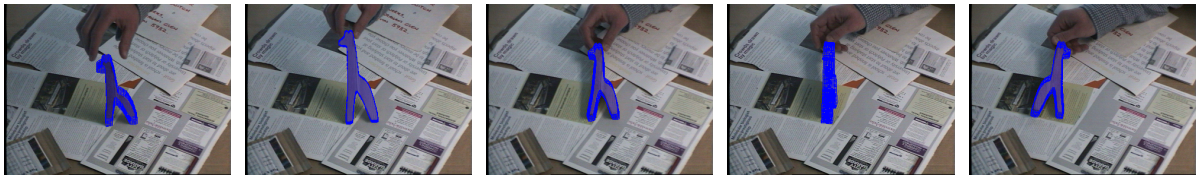


Figure 5.10.: This figure shows the estimated pose of a wooden giraffe in some frames of a monocular image sequence. **From left to right:** Frames 32, 48, 64, 74, and 84. The surface model was given in patch representation (see Section 2.4.1), and consists of a single patch. Thus, it is possible to look through the projected object.

aeroplane is very small. Since there is only one camera view available, this results in depth ambiguities which in turn lead to errors in the X and Z component with different signs (compare the coordinate system shown in the upper right part of the first image in the second row). The error at the last frame of the sequence is due to the fact that the 3-D model of the aeroplane turned incorrectly. Since there is only a tiny part of the aeroplane visible, as seen in the last image shown in Figure 5.13, it is not surprising that our tracking approach has difficulties at this frame.

5.5.2. Kinematic Chains

Next, we evaluate the performance of our algorithm when tracking kinematic chains, i.e. models in which unknown joint angles must be estimated in addition to the global position and orientation of the complete object. Due to the high importance of human tracking, the most common case in which a kinematic chain is used is when tracking a human body. It is quite simple to create sequences in which a human moves, while other objects modelled as kinematic chains are often more difficult to move around in a controlled manner without undesired occlusions. Thus, results for four sequences with humans are presented here. For these sequences, four camera views were available to track the movement of the different humans.

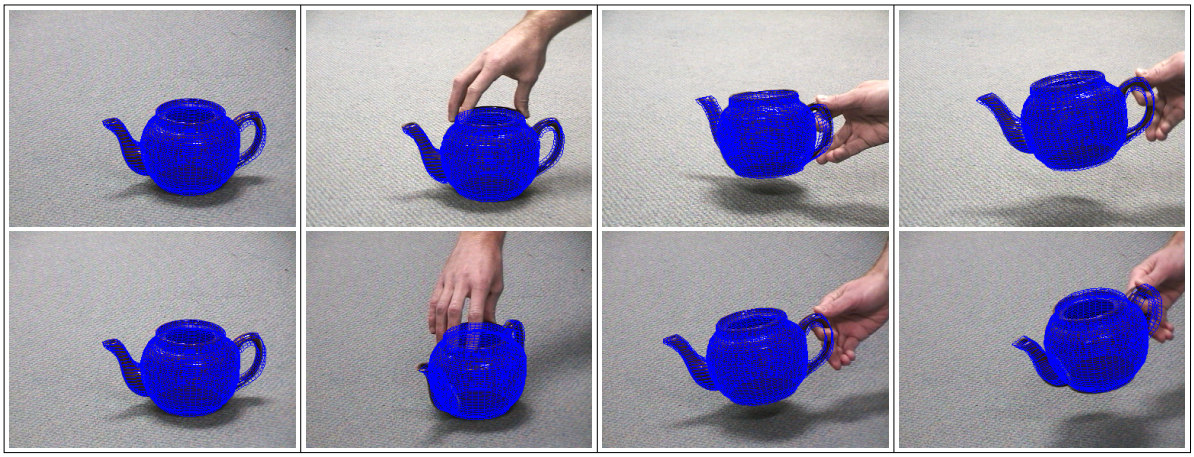


Figure 5.11.: Frames 0, 62, 130, and 240 of a stereo sequence of a moving tea pot. The mesh was in given in patch representation.

The first sequence is the most simple one. It shows a human running in an outdoor environment into a fixed direction. The presented algorithm can easily track this scene, as can be seen in Figure 5.14.

The next two sequences presented are more complex. They show a person turning a cart wheel (Figure 5.15) and a flip (Figure 5.16), respectively. For both sequences, prior knowledge about likely poses is used (see Section 5.4.2). This is necessary because, in some frames, there are no point correspondences for one of the leafs of the kinematic chain, which results in tracking being aborted.

For both sequences, tracking works quite well, apart from smaller problems. Most tracking inaccuracies visible are due to the fact that a human cannot be modelled perfectly by a kinematic chain. Furthermore, the feet are sometimes inaccurate since the feet are white while the remainder of the human is black. Thus, once the feet are not accurate in one frame, the resulting PDF of the inside region does not include white parts and accordingly, the feet are unlikely to enter the white region again. This is a general problem with small parts at the end of the kinematic chain that have an appearance different from the remainder of the object. The extension introduced in Chapter 7 helps to solve this problem.

As a last sequence, we track sequence S4 of the HumanEva-II benchmark [251, 250]. For this sequence, which is provided by Brown University, colour images from four cameras are available. The other eight cameras visible in the images were used to measure ground truth positions of several markers attached to the moving person. Although this ground truth data is not publicly available, tracking results can be evaluated by uploading the estimated marker positions to Brown University, from which the average tracking error per frame is computed. It should be noted that the ground truth of the frames 298 until 325 is corrupted. Thus, we will linearly interpolate the error for these frames.

This sequence consists of three parts: In the first part (approximately up to frame 400), the person is walking. Then, it starts to run until around frame 800. The last part consists of balancing movements. The running part in the middle of the sequence is the most difficult one, as we will see later.

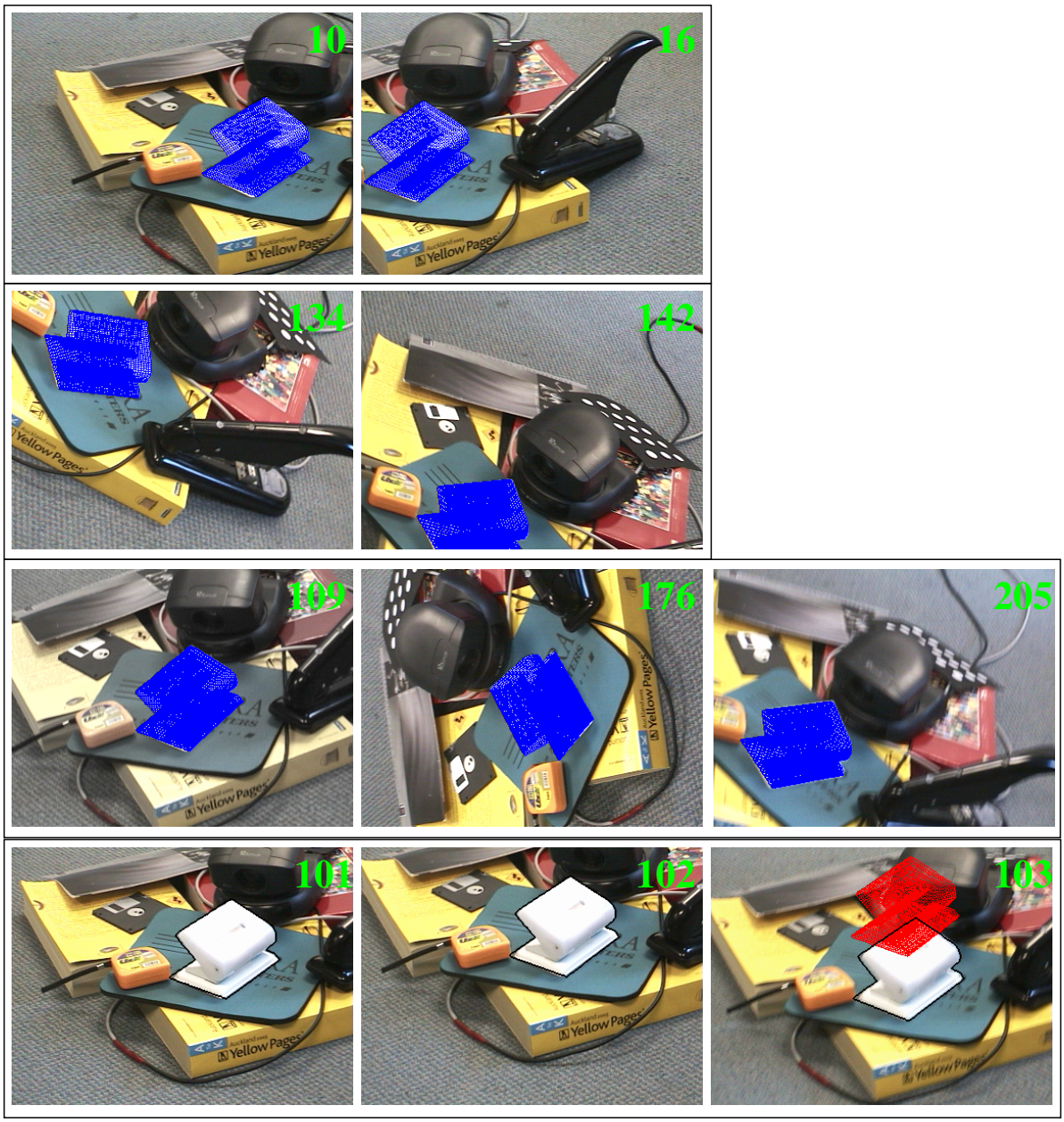


Figure 5.12.: Tracking results of a monocular sequence with a puncher. The green numbers in each image denote the number of the frame shown. Despite fast translational and rotational motion, colour aberrations, large rotations, and motion blur, tracking worked very well. These problems are depicted in the first three rows. In the images shown in the last row, there is a strong acceleration. This lead to the very bad prediction shown in red in frame 103. Still, by using the PDFs from the previous frame, our algorithm can deal with this problem. The final tracking result is shown by black contours in the last line.

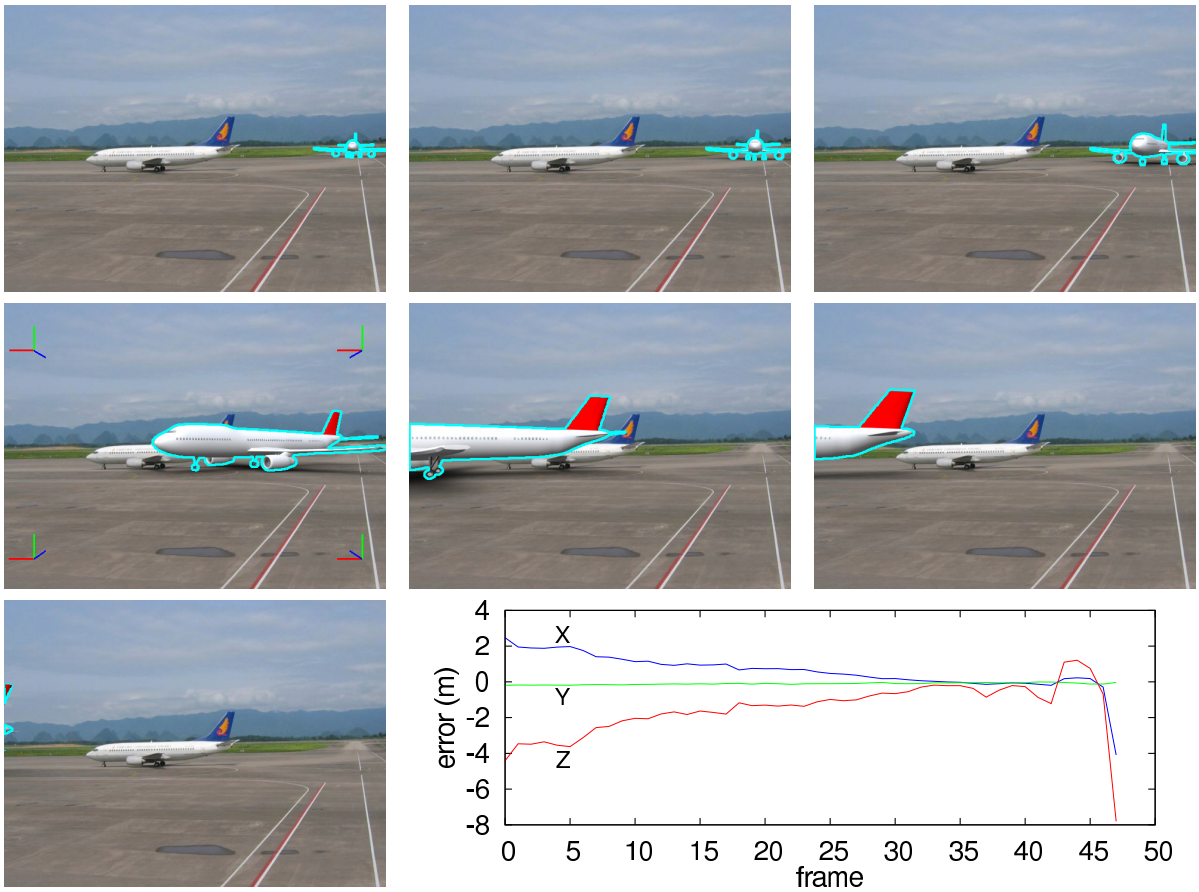


Figure 5.13.: Silhouette of the tracked pose in the frames 0, 10, 20, 30, 40, 44, and 47 of a monocular, synthetic scene with a moving aeroplane. Despite the facts that the aeroplane almost completely left the image, and that there is a similar aeroplane in the background, tracking results look very convincing. The coordinate system used is shown in the first image in the second row (frame 30), while the plots in the lower right corner show the translational tracking errors in each frame, given in meters. The reasons for the errors visible in these graphs are explained in detail in Section 5.5. The input sequence was generated and kindly provided by Richard Steffen.

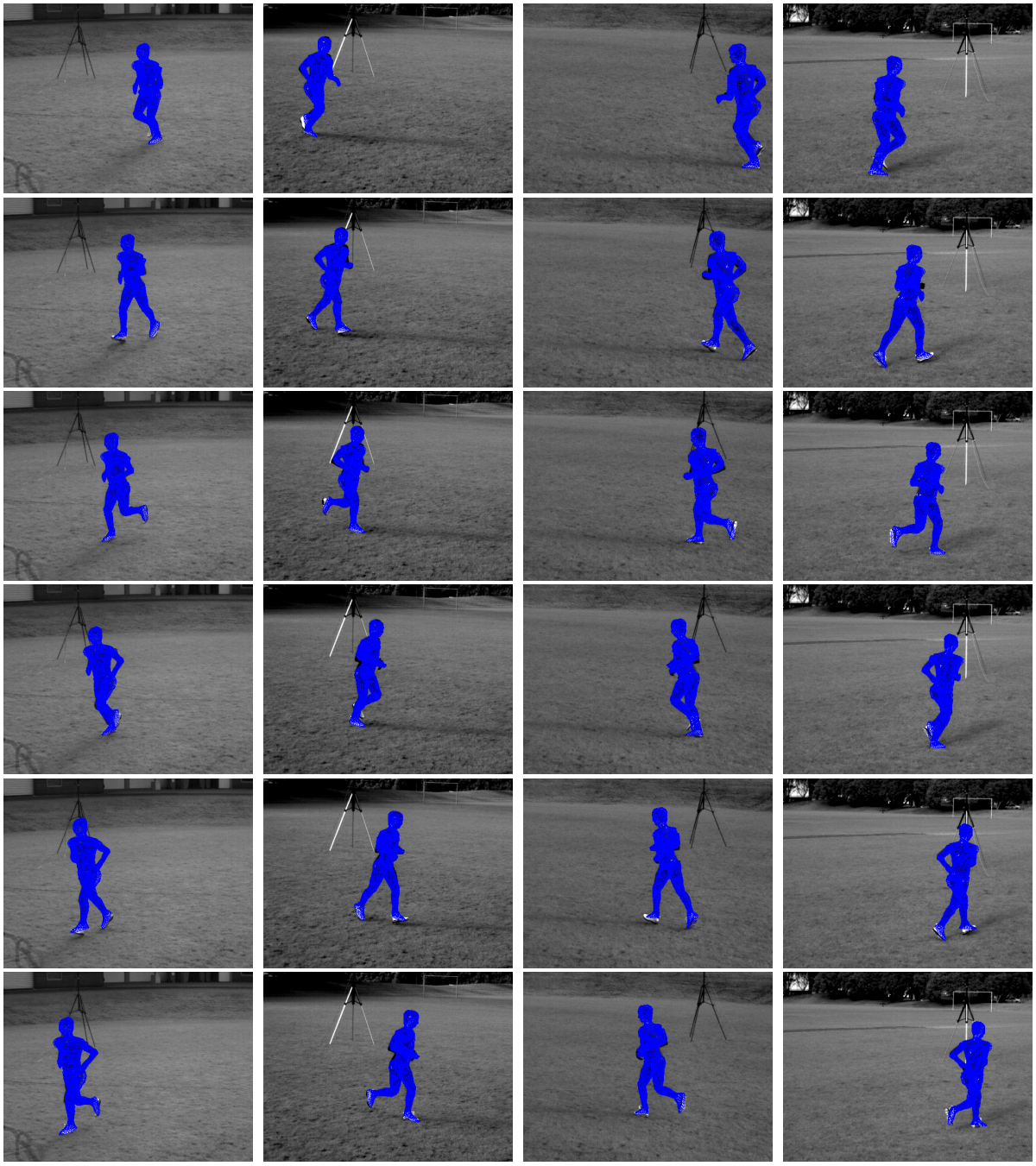


Figure 5.14.: Tracking results of a running person for frame 0, 20, 40, 60, 80, and 97, shown in all four views. Although the images are monochromatic, the whole sequence has been tracked without problems. Note that no prior knowledge about the movement pattern was used.

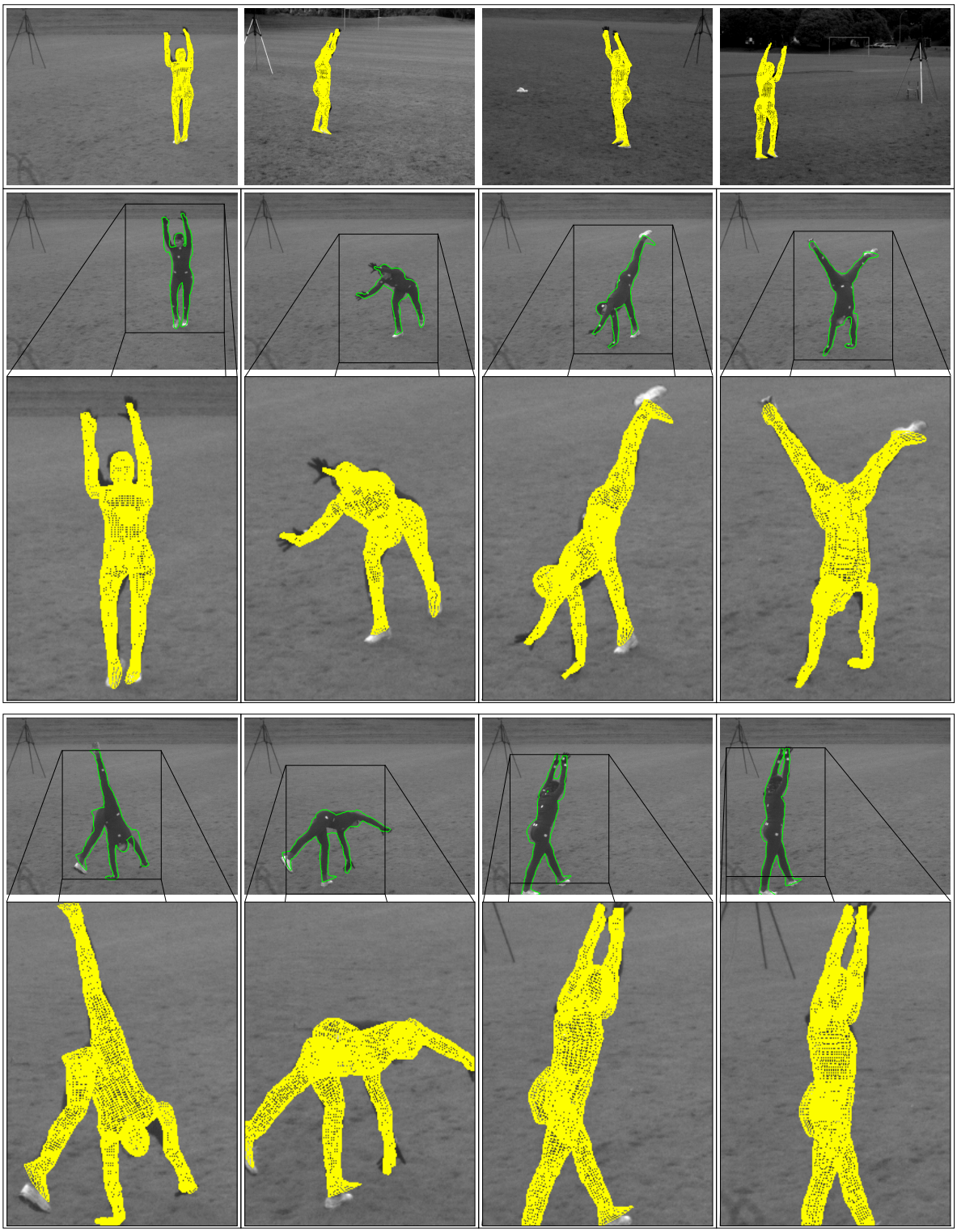


Figure 5.15.: Initialisation in the first frame (upper row) in all four views of the scene, and tracking results shown in the first camera view available. Shown are the frames 90, 180, 220, 260, 300, 340, 400, and 460 of a person doing a cart wheel.

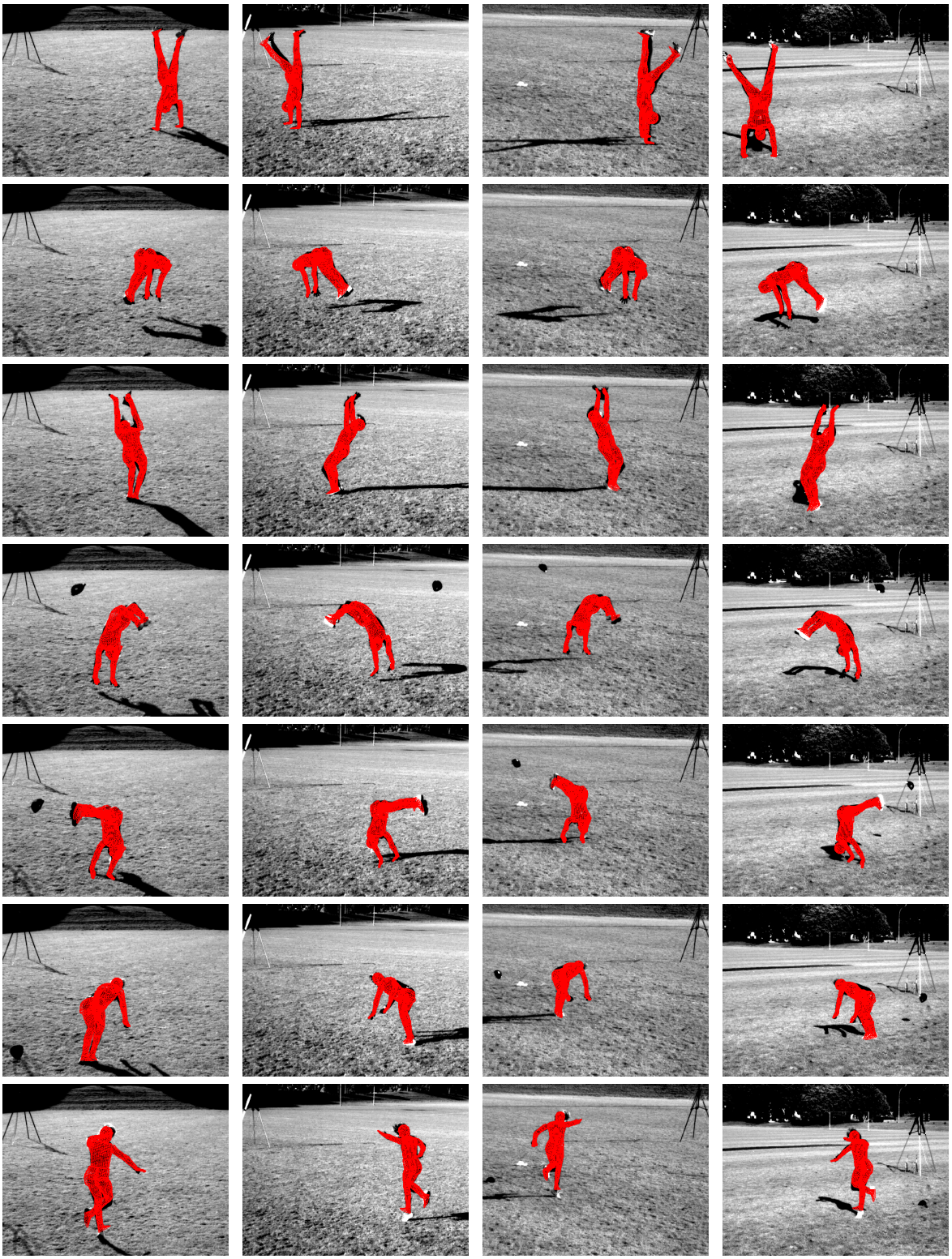


Figure 5.16.: Tracking results for frames 0, 37, 74, 111, 148, 185, and 259 of a person doing flips. All four views are shown.

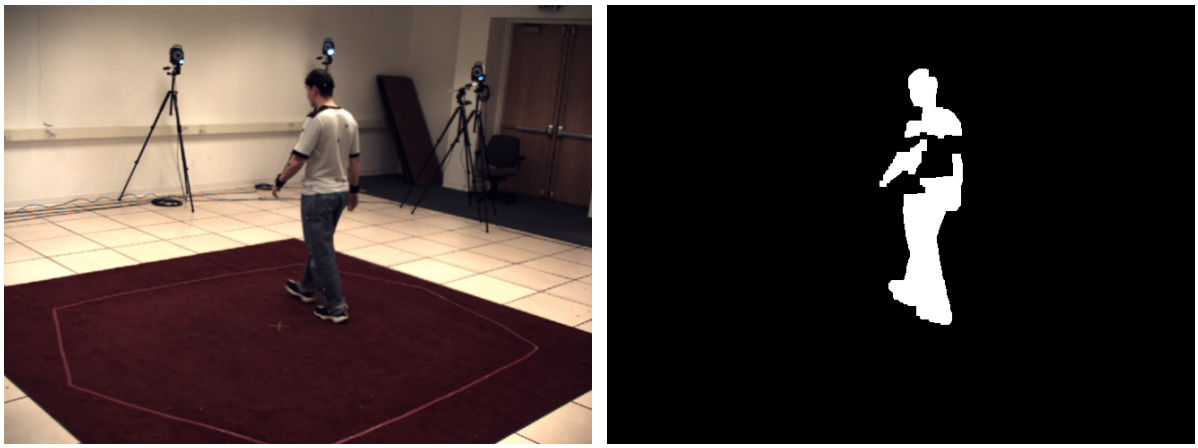


Figure 5.17.: Frame 42 of sequence S4 of the HumanEva-II benchmark, and the silhouette image obtained by performing background subtraction. Since the appearance of the shirt and the wall are too similar, background subtraction results are poor. Also notice that the space between the legs was not correctly classified.

Note that the background of each camera is static. Thus, it is possible to approximately separate the human from the background by performing *background subtraction* [204]. However, background subtraction is a difficult task itself. Thus, the estimated foreground region is often of poor quality, as shown in Figure 5.17. Here, such silhouette images are used as additional feature for the sequence, except for some experiments using the localised mixture models introduced in Chapter 8.

Figure 5.18 shows tracking results for all frames of the sequence, together with the average error reported. For this experiment, we did neither use angle priors from motion databases nor knowledge about the ground plane. Although tracking results are not completely wrong, inaccuracies are easily visible. These inaccuracies are due to several reasons ranging from self-occlusion over object-floor intersections to problems to distinguish fore- and background.

When including the floor constraints explained in Section 5.4.3, the tracker can overcome some of the problems, as illustrated in Figure 5.19. However, serious tracking failures still occur. We will show in later chapters how tracking results can be further improved. A comparison of all results presented for this sequence is given in Table 8.1 in Section 8.3.

5.6. Summary

In this chapter, we introduced our basic tracking algorithm, and evaluated it using a large variety of different object models and video sequences. Furthermore, we explained a number of extensions that are very helpful in some situations. Our algorithm can track both rigid objects as well as kinematic chains and fuse information from several cameras. However, it was also shown to work well for monocular sequences.

However, our tracking algorithm is only able to track a single object so far. In the next chapter, we introduce the necessary concepts to track multiple overlapping objects simultaneously.

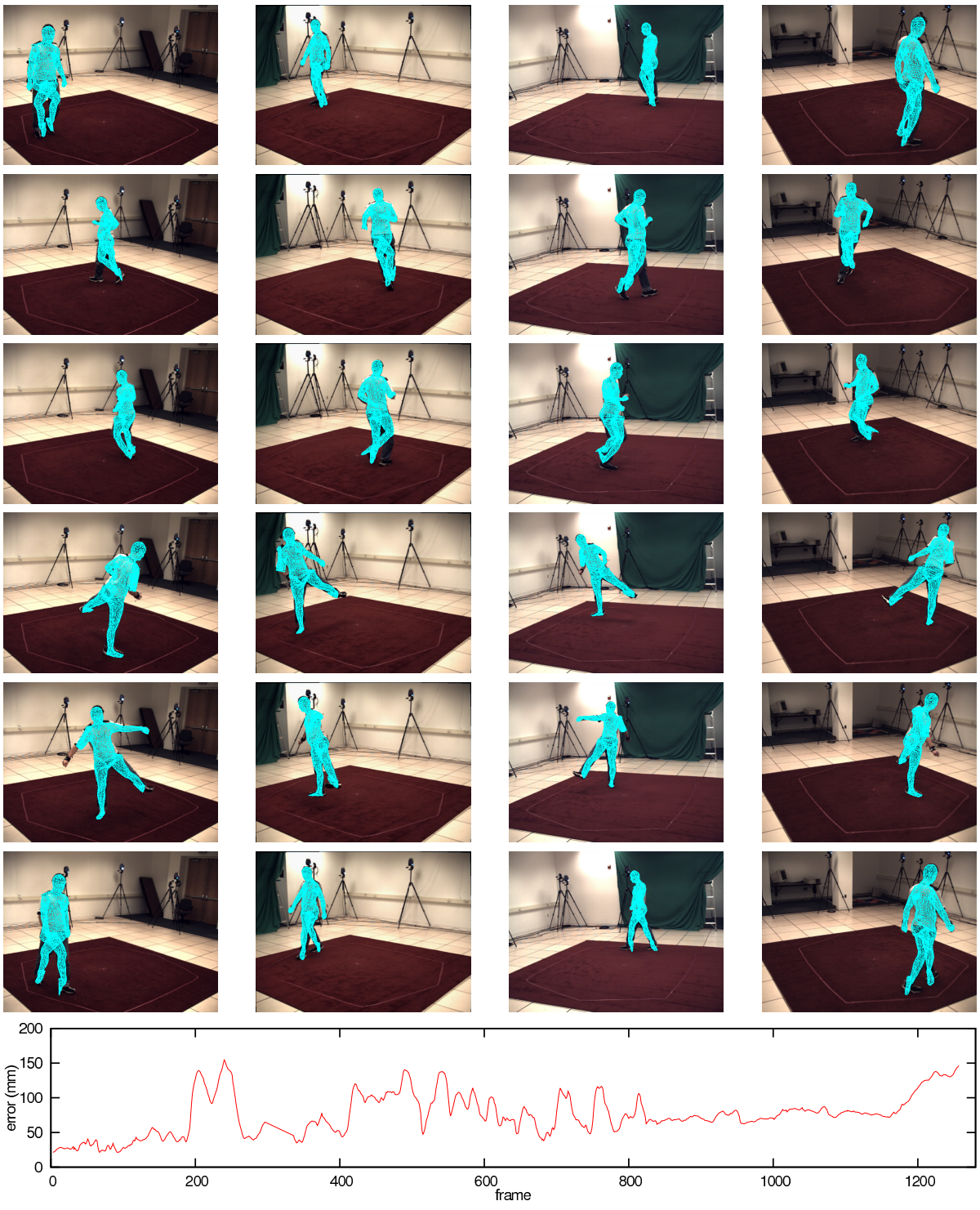


Figure 5.18.: These images show the tracking results for sequence S4 of the HumanEva-II benchmark with the basic tracking approach presented in this chapter. Shown are, from top to bottom, one frame of the walking part (frame 222), two frames in the jogging part (frames 444 and 666), two frames in the balancing part (frame 888 and 1110), and the frame with the highest tracking error (frame 757). The plot at the lower end of the figure shows the average tracking error per frame as reported by Brown university.

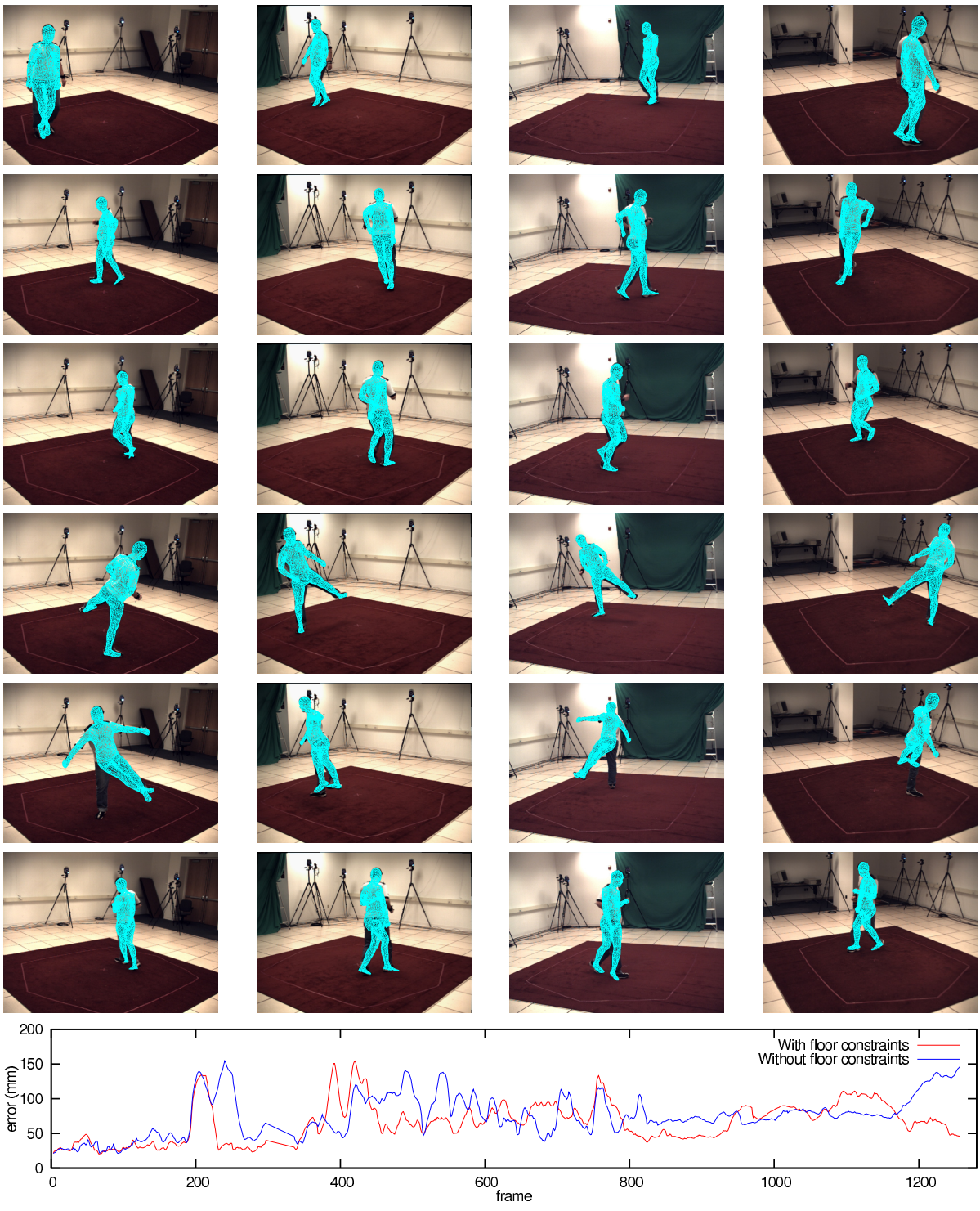


Figure 5.19.: Tracking results using floor constraints. **From top to bottom**: Tracking results in one frame of the walking part (frame 222), two frames in the jogging part (frames 444 and 666), two frames in the balancing part (frame 888 and 1110), and in the frame with the highest tracking error (frame 420). The plot at the lower end of the figure shows the average tracking error per frame achieved with floor constraints (red) and with the basic tracker (see Figure 5.18, blue).

6 Tracking Multiple Objects

"What we call 'Progress' is the exchange of one nuisance for another nuisance."

Havelock Ellis (1859 – 1939)

As shown in Section 5.5, our tracking approach can handle different objects, a varying number of cameras and different kinds of features. It can even handle partial occlusions (see Figure 5.7) and partial self-occlusions (see Section 5.5.2) up to some degree. As soon as too much of the object to be tracked is occluded, tracking will fail, though.

To understand why tracking fails in such a situation, consider the synthetic sequence depicted in Figure 6.1. At first glance, this sequence is very easy to track: The object to track (the green puncher used in Figure 5.12) is very clearly separated from the background, and there are even two views available. Still, tracking fails due to the occlusion, as illustrated in Figure 6.2.

The problem with occlusions is that the occluded parts lead to wrong estimations of the PDF of the inside region. In the example explained above, the percentage of the foreground region that is yellow is larger than the percentage of the yellow background region for some frames. Thus, the tracking algorithm wrongly assumes that yellow points belong to the object instead of to the background. Consequently, it tries to find the pose in such a way that the yellow points belongs to the foreground, which finally leads to a tracking failure.

In such a situation, tracking both the green puncher as well as the yellow tea box can be beneficial even if we are only interested in one of the objects. Moreover, there are also tasks in which the poses of multiple objects are searched for.

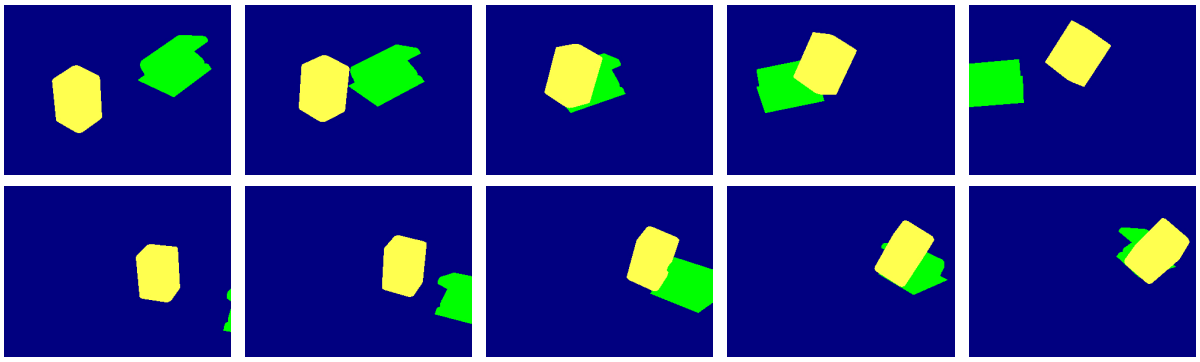


Figure 6.1.: A simple looking synthetic stereo sequence, created by projecting a tea box (yellow) and a puncher (green) onto a blue background. Both objects perform a simple and regular movement in 3-D space. Shown are frames 0, 4, 9, 14 and 19 in both views available for tracking.

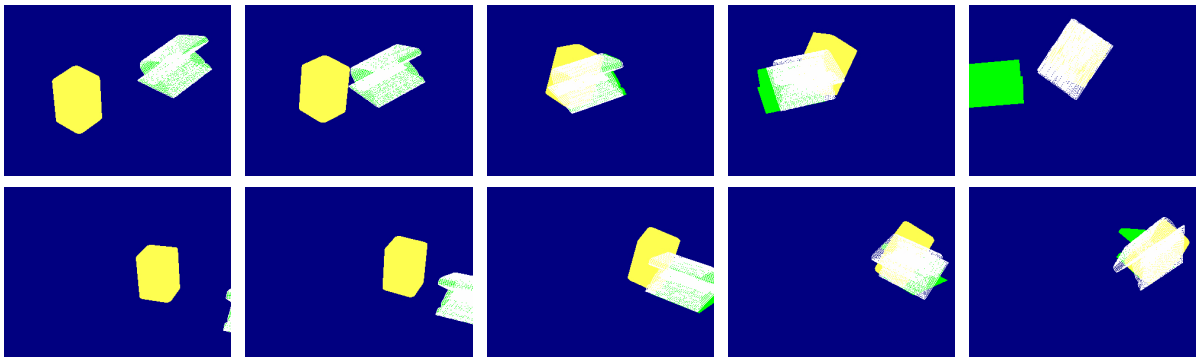


Figure 6.2.: Result when trying to track the puncher in the sequence shown in Figure 6.1. Due to occlusions by the yellow tea box, tracking the green puncher fails.

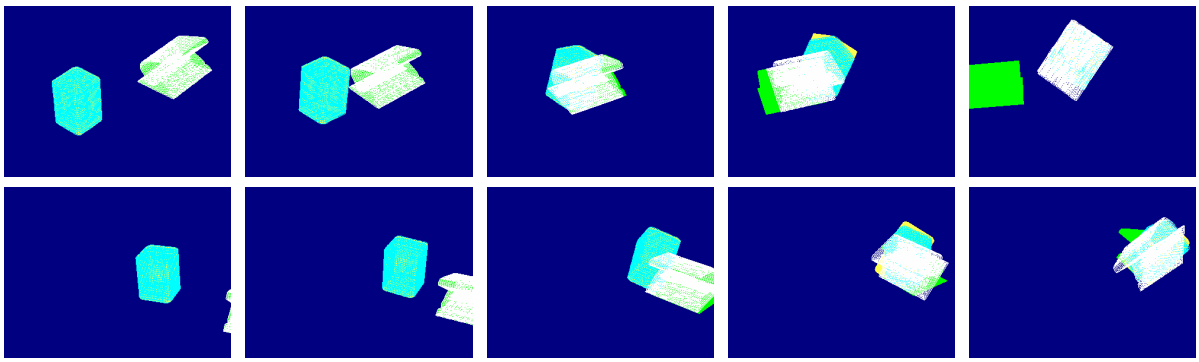


Figure 6.3.: Tracking results of uncoupled tracking. Again, the frames 0, 4, 9, 14 and 19 are shown. It can be seen that both objects are not tracked accurately around frame 14. While the pose of the yellow tea box can be recovered in later frames, tracking the puncher fails completely.

(a) (b) (c) (d) (e)

Figure 6.4.: (a) Magnification of frame 12 of sequence “Synthetic double” shown in Figure 6.1, (b) illustration of the object regions in this image, (c) silhouette of the puncher, (d) non-occluded region of the puncher, and (e) silhouette of the puncher used by the coupled tracker.

In this chapter, we explain how to track multiple objects simultaneously. We start by explaining a simple uncoupled tracking, which does not take occlusions into account. We proceed with the coupled model we proposed in [238], and improved in [237]. Some related work on multiple object tracking is given in [115], where a Gibbs sampler and particle filters are utilised for tracking a variable number of objects. Anther approach is to use a Rao-Blackwellised sequential Monte Carlo method, as demonstrated in [230]. These two approaches only perform 2-D tracking in the image domain, and are thus very restricted in handling mutual occlusions.

After introducing our simple uncoupled tracking algorithm in Section 6.1, we explicate how to couple the tracking of multiple objects in Section 6.2. Section 6.3 further improves this approach. After performing a detailed evaluation in Section 6.4, we conclude the chapter in Section 6.5.

6.1. Uncoupled Tracking of Multiple Objects

"Attempt the impossible in order to improve your work."

Bette Davis (1908 – 1989)

When tracking multiple objects, we assume that a 3-D object model and an initial pose is available for each of the N objects to be tracked. The pose of the i-th object will be denoted by $\boldsymbol{\chi}_i$ even if the object to be tracked is rigid. The task is to find the pose $\boldsymbol{\chi}_i$ of each object i in each frame. Once again, this task is summarised in the following table:

Given:
– One projection matrix for each camera (see Section 2.6)
– One image sequence with domain $\Omega \subset \mathbb{R}$ for each camera
– Models for each object i to track as rigid object or kinematic chain with n_i joints (see Section 2.4)
– Approximate 3-D pose initialisation $\boldsymbol{\chi}_i \in \mathbb{R}^{6+n_i}$ for each object i in the first frame
Assumptions:
– Each object is always (partly) visible
– The appearances of objects and background differ
Task:
– Track the 3-D pose of all objects through all images

To find the pose of each object, the easiest way is to extend the energy function for one object from Equation (5.1) such that it depends on all poses. This can be achieved by adding each individual energy function for each object, resulting in:

$$E(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_N) = -\sum_{i=1}^{N} \int_{\Omega} \Big[c_{\boldsymbol{\chi}_i,i}(\boldsymbol{x}) \log p_{i,\text{in}} + \big(c_{\boldsymbol{\chi}_i,i}(\boldsymbol{x})\big) \log p_{i,\text{out}} \Big] \mathrm{d}\boldsymbol{x}. \tag{6.1}$$

In this equation, $p_{i,\text{in}}$ and $p_{i,\text{out}}$ are the PDFs for the inside region (or outside region, respectively) of the i-th object, and $c_{\boldsymbol{\chi}_i,i}$ is the indicator function of the points in which the i-th object with pose $\boldsymbol{\chi}_i$ can be seen in the image, cf. Equation (5.2). As the different poses correspond to different parts of the energy function, each pose can be optimised independently of the others.

Due to this independence, tracking works fine as long as the objects do not occlude each other. If a large part of one object is occluded, though, the problems described at the beginning of the chapter occur, and tracking fails. This problem is illustrated in Figure 6.3. Note that not only the occluded green puncher is not tracked correctly, even the yellow tea box is tracked inaccurately around frame 14. The reason why the tea box is only tracked inaccurately is either a very slight occlusion (the objects actually pass through each other in 3-D space) or an insufficient number of iterations due to a wrongly chosen pose change threshold $\boldsymbol{T}$. When using the energy function introduced in the next section, these problems disappear.

6.2. Coupled Tracking of Multiple Objects

"Ideas not coupled with action never become bigger than the brain cells they occupied."

Arnold H. Glasgow

In order to understand the problems occurring when tracking occluding objects with the simple energy function (6.1) more clearly, consider the magnification of frame 12 in Figure 6.4(a). Assume that the pose of both the puncher $\boldsymbol{\chi}_1$ and the tea box $\boldsymbol{\chi}_2$ are already perfectly accurate. Then, $p_{1,\text{in}}$ is estimated from the green and yellow area in Figure 6.4(b), while $p_{1,\text{out}}$ is estimated from the blue and the orange area depicted in the same figure. Thus, slightly more than half of the interior of the puncher is yellow in this picture (53.4%), while only very few pixels in the outside region are yellow (3.5%)[1]. Consequently, a yellow point $\boldsymbol{x}$ will be classified as inside the interior of the puncher, as $p_{1,\text{in}}(\boldsymbol{x}) > p_{1,\text{out}}(\boldsymbol{x})$ holds, and yellow silhouette points will be moved outwards by our tracking algorithm. As a result, the puncher moves towards the upper right direction in the image. Shortly after the wrong movement, green points might even be assumed to lie outside the puncher region.

To handle this problem, we have to notice that a part of the puncher is occluded by the tea box, are ignore this part when computing $p_{1,\text{in}}$. That is, only the green part shown in Figure 6.4(d) should be used for this computation. To achieve this goal, we now utilise a *visibility function* $v_i\left(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_n, \boldsymbol{x}\right) \in \mathbb{R}^{6+n_1} \times \ldots \times \mathbb{R}^{6+n_n} \times \Omega$ for each object i [261]. These functions are then used in an energy function designed for a coupled tracking of multiple objects.

First of all, we define the sets $O_i(\boldsymbol{\chi}_i, \boldsymbol{x})$ containing all points on the i-th object model that are projected to a certain image point $\boldsymbol{x}$. Then, we define the distance of the object model to

[1] Remember that only a part of the image is shown in Figure 6.4, and that the orange area in Figure 6.4(b) is actually yellow, and is only shown in orange to show which image regions are occupied by the objects.

this image point as

$$d_i(\boldsymbol{\chi}_i, \boldsymbol{x}) := d(O_i(\boldsymbol{\chi}_i, \boldsymbol{x}), \boldsymbol{C}) = \min_{\boldsymbol{y} \in O_i(\boldsymbol{\chi}_i, \boldsymbol{x})} \{d(\boldsymbol{y}, \boldsymbol{C})\}, \tag{6.2}$$

where $\boldsymbol{C}$ is the optical centre of the camera and d is the standard Euclidean metric. Only the object whose distance to the optical centre is closest will be visible in the image. Thus, the visibility functions are given by

$$v_i(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_n, \boldsymbol{x}) = \begin{cases} 1 & \text{if } d_i(\boldsymbol{\chi}_i, \boldsymbol{x}) = \min\limits_{j \in \{1,\ldots,n\}} \{d_j(\boldsymbol{\chi}_j, \boldsymbol{x})\} \quad , \\ 0 & \text{else} \qquad\qquad\qquad\qquad\qquad\quad . \end{cases} \tag{6.3}$$

Note that visibility is determined on a per-pixel basis. Thus, even if the first object occludes the second in one image region, while it is the other way round in another part of the same image, the proposed tracking approach will be able to deal with both occlusions. As a consequence, this approach is clearly superior to simple approaches in which each model is either completely in front of or behind another. To improve readability, the arguments of the visibility functions will usually be omitted from now on.

When including the visibility functions into the energy function given in Equation (6.1), we obtain:

$$E(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_N) = -\sum_{i=1}^{N} \int_{\Omega} \Big[v_i c_{\boldsymbol{\chi}_i, i}(\boldsymbol{x}) \log p_{i,\text{in}} + \big(1 - v_i c_{\boldsymbol{\chi}_i, i}(\boldsymbol{x})\big) \log p_{i,\text{out}} \Big] \mathrm{d}\boldsymbol{x}. \tag{6.4}$$

The indicator functions $c_{\boldsymbol{\chi}_i, i}(\boldsymbol{x})$ are multiplied by the corresponding visibility functions such that only those points of the object that are actually visible are used for estimating $p_{i,\text{in}}$. This results in a different PDF for inside and outside region. In our example explained above, only the green region shown in Figure 6.4(d) is used to compute $p_{i,\text{in}}$, while the yellow and blue parts lie completely in the region used to estimate $p_{i,\text{out}}$.

To find image-vertex point correspondences necessary for the point-based pose estimation step, we need the silhouettes of the tracked objects. There are two (suboptimal) silhouettes that could be used, namely the silhouette ignoring occlusions shown in Figure 6.4(c), or the silhouette after accounting for occlusions, i.e. the boundary between the green and the blue region in Figure 6.4(d). It is immediately obvious why the first choice should not be used: As a part of the object is occluded, so is a part of the silhouette. Consequently, those occluded points cannot be used, as there is no object boundary visible in the image.

To understand why the second possibility is also not optimal, consider the idea behind the pose tracking algorithm: Each silhouette point casts a vote for the direction in which the projection of the object should move; A point which seems to belong to the object tries to have nearby points included as well, while points that seem to belong to the background try to move the object away such that these points actually belong to the background region. Therefore, if the movement proposed by each silhouette point is performed, each silhouette point benefits from this movement as it gets closer to the actual contour seen in the image. However, this is not the case for those points which are only on the silhouette due to an occlusion. No matter how the object moves, this part of silhouette will stay the same, at least for small movements.

In fact, the other object must be moved to change this part of the silhouette. Thus, only those points which are on the silhouette both before and after considering occlusions are used to estimate the motion of the puncher. In our example, those points are shown in Figure 6.4(e).

As each part of the energy function (potentially) depends on each pose, searching the poses independently of each other is no longer possible. In our implementation, we simultaneously perform one gradient descent step for each object until all objects move less than the given threshold $\boldsymbol{T}$ from Equation (5.9).

This concludes the introduction of the energy function for coupled tracking of multiple objects introduced in [238]. This method still has a major drawback, which will be removed in the next section. Thus, experiments are postponed until Section 6.4.

6.3. Improved Tracking of Multiple Objects

"First ask yourself: What is the worst that can happen?
Then prepare to accept it. Then proceed to improve on the worst."

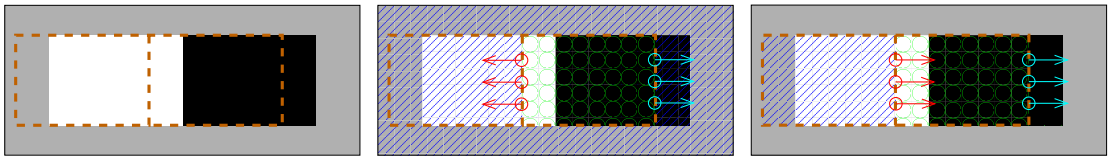
Dale B. Carnegie (1888 – 1955))

Minimising the energy function (6.4) for coupled tracking of multiple objects often yields good results. However, there are situations in which problems can occur. In this section, we explain why and when problems appear, and answer the question how the energy function can be improved such that these problems do not occur any more.

Consider Figure 6.4 to understand the possible problem discussed now. This time, assume that the 3-D poses of the two objects are in such a way that each object exactly occupies the image region it should, but the puncher is actually in front of the tea box. That is, the puncher is visible in the green and yellow part in Figure 6.4(b), while the tea box is visible only in the orange part. Obviously, when tracking the puncher, we face same problems as in the uncoupled case, as it is not occluded.

To illustrate that the problem we described can even occur when there are no occlusions, consider Figure 6.5(a) as a second example. The black and the white rectangle are to be tracked, while the dashed brown lines indicate the current position of the object models. Using the coupled energy function from the last section, the regions marked by the green circles and by the blue lines in Figure 6.5(b) are compared to decide in which direction the points on the silhouette of the right object move. As 20% of the green "inside" region is white, but only around 14% of the blue "outside" region, white points are classified as belonging to the object. Thus, the points marked in red move in outwards normal direction, i.e. to the left. The black points marked by cyan circles on the right side are correctly classified as inside the object, and thus move in outwards normal direction, i.e. to the right. As a result, the induced forces cancel each other and the right object stays at the incorrect position.

The reason why a simple coupled tracking fails is the following: Although each image point is assigned to at most one object region, it is still assigned to multiple background regions.



(a) (b) (c)

Figure 6.5.: **Left**: Image in which the white and the black object should be tracked. The dashed brown lines indicate the currently estimated position of the two objects. **Middle**: Situation when using a simple coupled tracking. **Right**: Situation with an improved coupling. The arrows indicate the displacement of some example points on the black object. The areas marked by green circles indicate the regions used for estimating the appearance of the interior of the right object, while the blue lines show the region whose appearance is compared against the appearance of the appropriate inside regions for the marked points.

That is, although the PDFs for the interiors of the object regions are accurate, those for the background regions still overlap, and are thus inaccurate.

To eliminate this problem, we propose to use a single background region instead of an individual background region for each object tracked. More precisely, we define a visibility function v_0 for the background as

$$v_0(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_N, \boldsymbol{x}) := \prod_{i=1}^{N} (1 - v_i)\,, \tag{6.5}$$

since the background is visible if and only if no object is visible. Furthermore, we define an indicator function c_0 for the background by setting

$$c_0(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_N, \boldsymbol{x}) := 1, \tag{6.6}$$

as the background covers the whole image due to the fact that occlusions are already handled. Furthermore, we set p_0 as the (unknown) PDF which describes the complete background region.

Then, the improved energy function is given by:

$$E(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_N) = -\sum_{i=0}^{N} \int_{\Omega} \Big(v_i c_{\boldsymbol{\chi}_i, i}(\boldsymbol{x}) \log p_i \Big) \mathrm{d}\boldsymbol{x}. \tag{6.7}$$

Using this energy function, each image point is assigned to exactly one PDF.

Minimising this energy function is done in a similar way as before. However, there are some important differences. First off all, the visibility functions are computed simultaneously by projecting each object with OpenGL [247], where a different colour is used for each object. The rendered image is then used to compute all visibility functions. As before, object regions are computed from which we collect the points lying on the silhouettes, i.e. those points which are on the silhouette before and after considering occlusions. Each of these points is then moved in inwards or outwards normal direction, depending on whether it fits better to the PDF

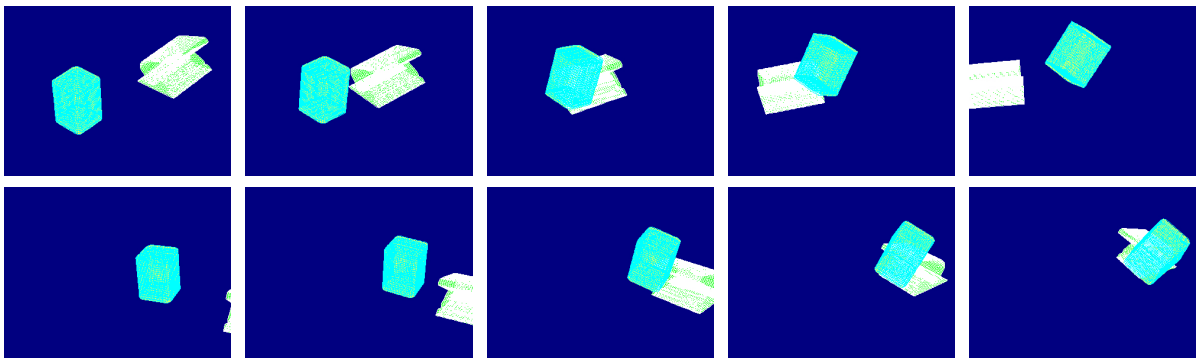


Figure 6.6.: Result when tracking the sequence shown in Figure 6.1 with the energy function for coupled tracking of multiple objects. As can be seen, the problems visible in Figure 6.3 do not occur any more.

of the object it belongs to, or to the PDF of the region next to it. The shifted points are then used to estimate the pose of each object using the point-based pose estimation algorithm from Section 4.1. This minimisation step is then repeated until each object motion is small enough, cf. Equation 5.9.

In the example explained at the beginning of this section, we will get three PDFs: The PDF p_1 for the interior of the puncher is estimated from green and yellow points, the PDF p_2 for the tea box indicates that it is completely yellow, and the PDF p_0 of the background region states that only blue points are in the background. For the points on the silhouette of the puncher which are adjoined to the background, i.e. for the silhouette points shown in Figure 6.4(e), the PDFs p_1 and p_0 are compared to decide to which region the points belong, while the PDFs p_1 and p_2 are used for the remainder of the points. In both cases, all points are correctly classified, which was not the case before.

When using the improved energy function in the second example, we are in the situation depicted in Figure 6.5(c): Again, 20% of the inside region are white. However, the points marked in red will compare their appearance against those of the left object, i.e. against the region marked by the blue lines. As 75% of this region is white, the points are now correctly classified and move to the right. Thus, the right object is moved into the correct direction.

6.4. Experiments

"I really like to experiment. That's the only way I can work. It's instinctive."

Fahrid Murray Abraham (*1939)

As a first experiment, we consider the sequence used throughout this chapter to illustrate shortcomings and improvements of our energy functions, namely the sequence depicted in Figure 6.1. Tracking results for this sequence are shown in Figure 6.6. It can be observed that coupled tracking works very well.

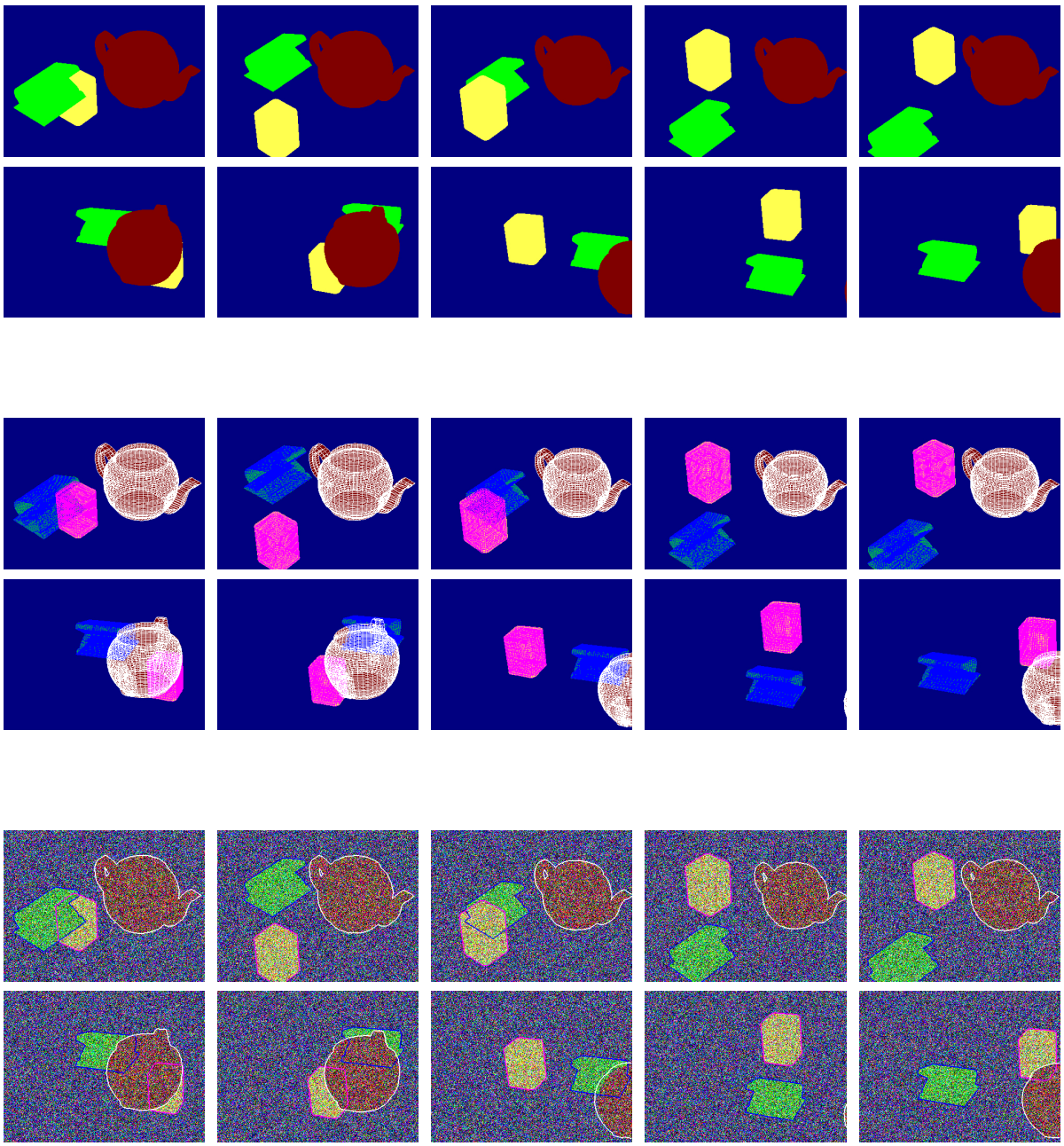


Figure 6.7.: Tracking results for a synthetic scene with three objects. The first two rows show the two views available, the third and fourth row the tracking results using the coupled energy function, and the last two rows the results when adding uncorrelated Gaussian noise with a standard deviation of 256 before tracking. Each row shows the frames 4, 9, 14, 19, and 24.

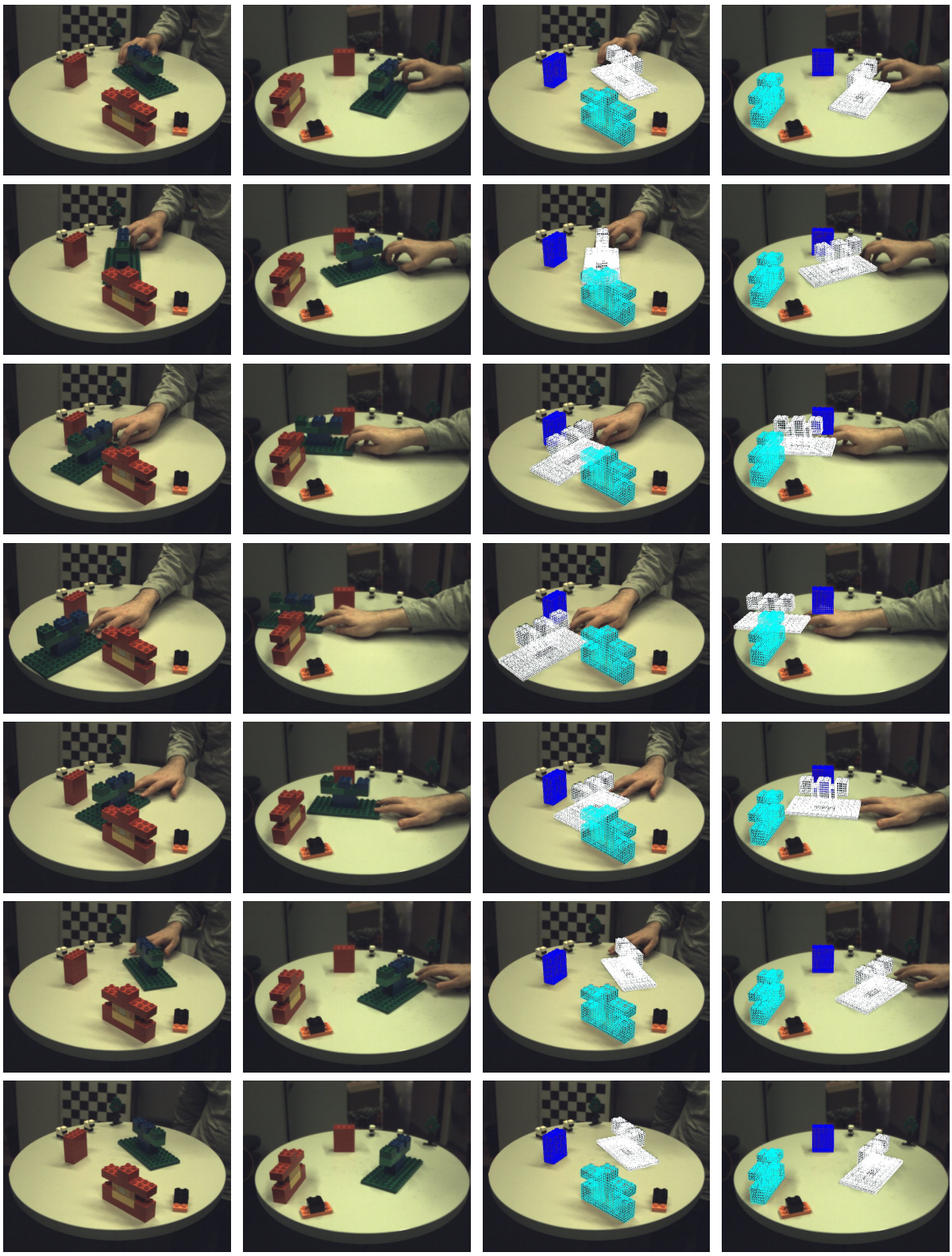


Figure 6.8.: Easy real-world sequence with tracking results. Shown are, from top to bottom, the frames 20, 60, 100, 140, 180, 220, and 260 in both views (left columns), and the tracking results in these frames (right columns).

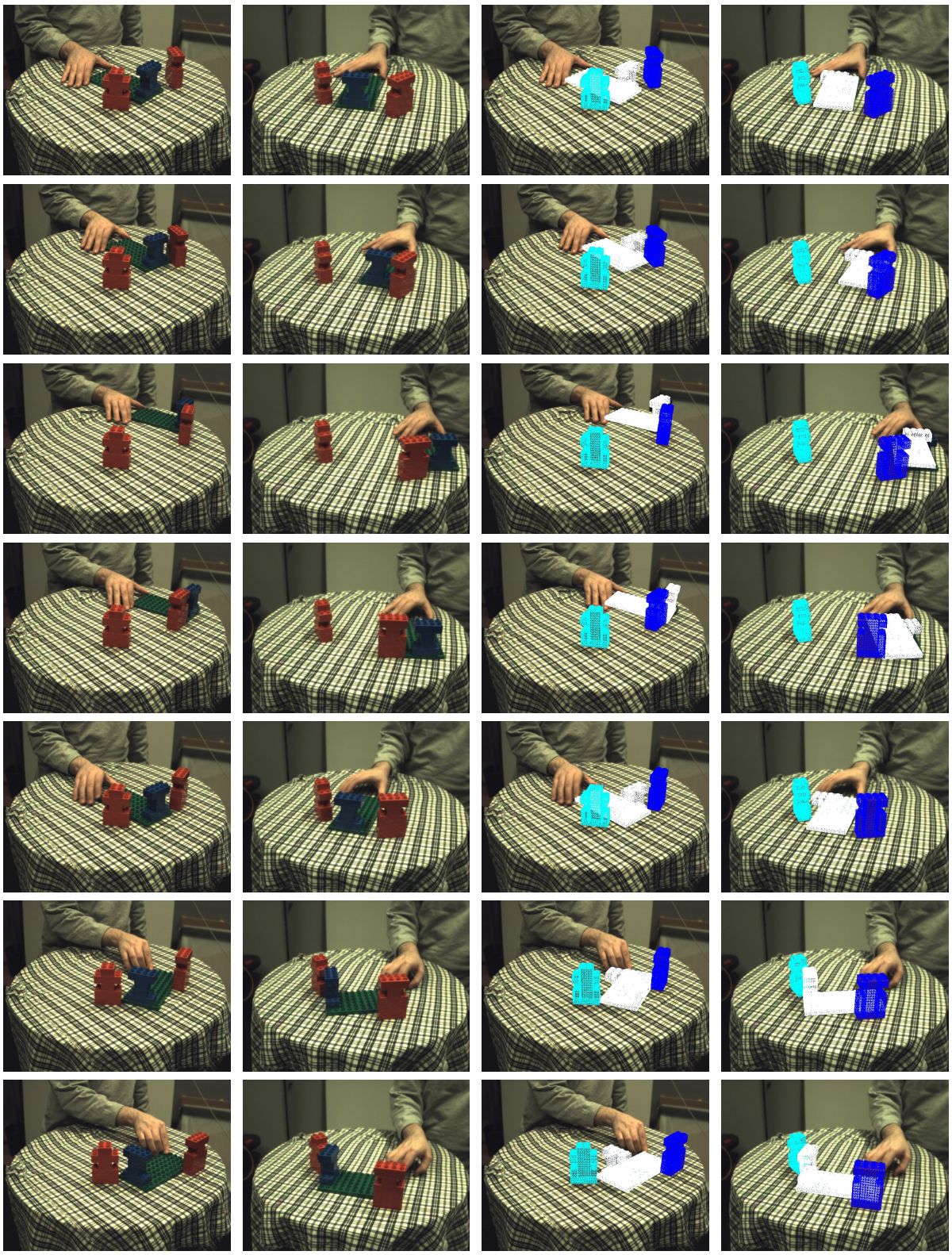


Figure 6.9.: In this sequence, three objects are moving. Although there are some problems around frame 130 and 190, tracking works well in general. Shown are, from top to bottom, the frames 10, 70, 130, 190, 250, 310, and 340 in both views (left columns), and the tracking results in these frames (right columns).

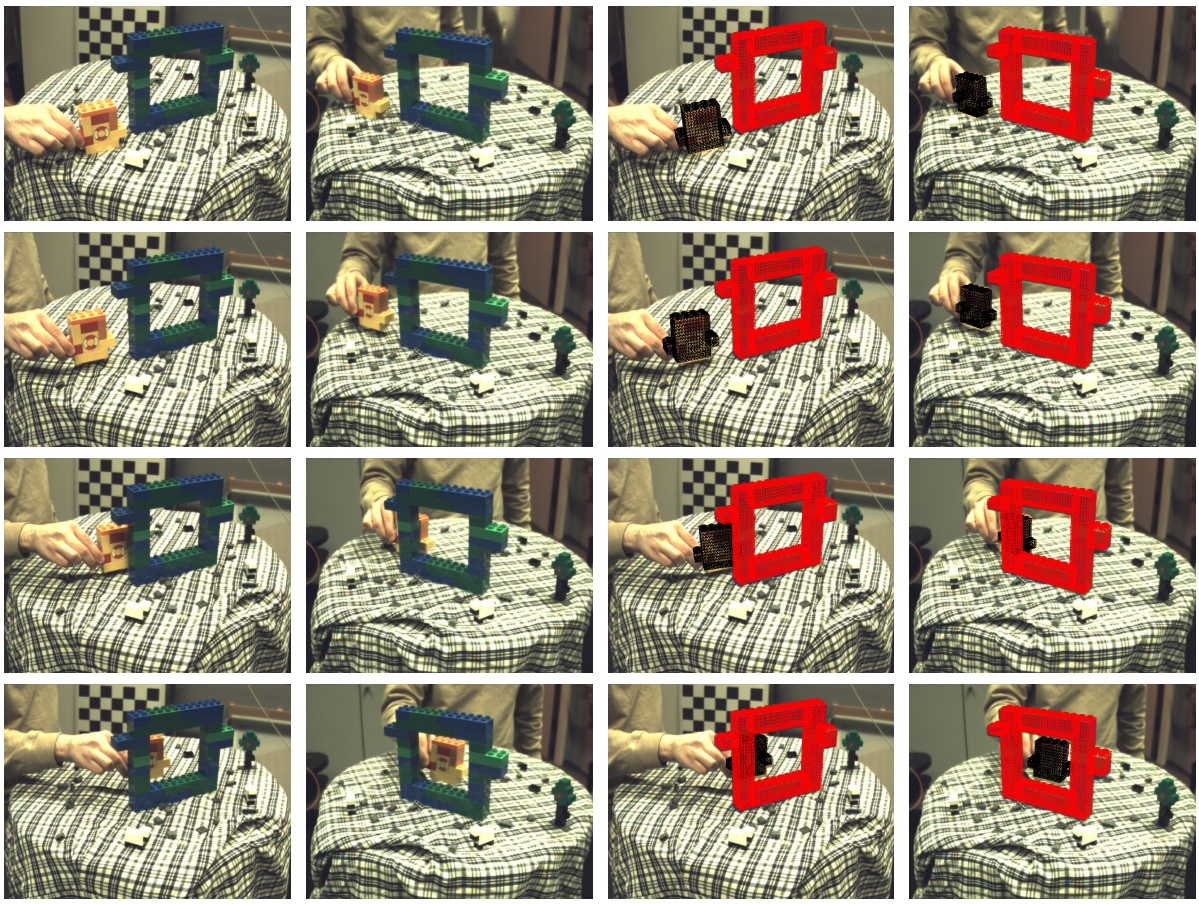


Figure 6.10.: Real-world sequence in which one object is only visible through a hole of the other. As can be seen, tracking also works in such a difficult situation. Shown are, from top to bottom, the frames 15, 45, 75, and 105 in both views (left columns), and the tracking results in these frames (right columns).

We also evaluate a more complex synthetic sequence with three objects, and multiple partial occlusions. This sequence, which is depicted in the first two rows in Figure 6.7 is tracked twice: Once without noise, and once after uncorrelated Gaussian noise with a standard deviation of 256 has been added. For the noisy sequence, our tracking algorithm has smoothed all images using a Gaussian kernel before estimating the PDFs. Both results are shown in Figure 6.7. There are some minor inaccuracies visible when tracking the noisy sequence, e.g. see the red teapot in the second view of frame 14. However, this is not surprising due to the amount of noise added.

In Figure 6.8, a real-world sequence with three Lego Duplo® objects and the corresponding tracking results are shown. Even though only the object built from green and blue bricks moves, all three objects have been tracked. However, tracking only the moving object also works, since the appearance of the background is homogeneous close to the objects and since only a small part of the moving object is occluded.

This is not the case for the second experiment we performed with Lego Duplo bricks, as this sequence is far more complicated: All three objects move, the background is not homogeneous, and a large part of one object is occluded in some frames, see Figure 6.9. These problems result in inaccurate tracking results for some frames around frame 130. In Section 7.3, we will show improved tracking results for this sequence using the extension to multiple internal object regions introduced in Section 7.1.

Although there are only two interesting objects used in the sequence shown in Figure 6.10 only one of which is moving, this sequence is even more complicated. For example, the yellow object has an appearance that is close to the hand moving it. Furthermore, there are simultaneous occlusions of significant parts of the yellow object in both views while it is moved behind the large one, and the appearance of the background is less uniform than before, as the table-cloth is wrinkled. Nevertheless, tracking results are only slightly inaccurate. Also note that the small object is only visible through a hole in the big one. This is no problem for our tracking algorithm, though.

As a last experiment, we combined tracking of multiple objects with the soft constraints introduced in Section 5.4.3 to track a cycling athlete. As explained before, the hands of the cyclist are required to stay at the handle of the bicycle, the pedals should rotate around a fixed axis, and right foot should stay on the right pedal. Note that the left foot is rigidly attached to the left pedal. Due to the soft constraints and due to the fact that mutual occlusions can be handled since both objects are tracked, tracking results look very good. They are shown in Figure 6.11. The second row in this figure also illustrates the different complex mutual occlusions: Areas where the bike is occluded by the cyclist have been drawn in green, while regions where it is vice versa are shown in dark blue. It can easily be seen that there are several mutual occlusions of each type in each view. Nevertheless, our general visibility functions automatically handle these occlusions.

6.5. Summary

Using the concepts introduced in this chapter, we extended the basic tracking algorithm from the last chapter such that it can track multiple object simultaneously. Instead of only assuming that one object is completely in front of another, our model can deal with situations in which different objects mutually occlude each other. Where occlusion occur is decided purely on the tracked objects on a per-pixel basis, i.e. without using prior knowledge of the scene, and may change in every iteration of our algorithm.

The ideas presented in this chapter are reused in the following chapter to deal with occlusions occurring in kinematic chains.

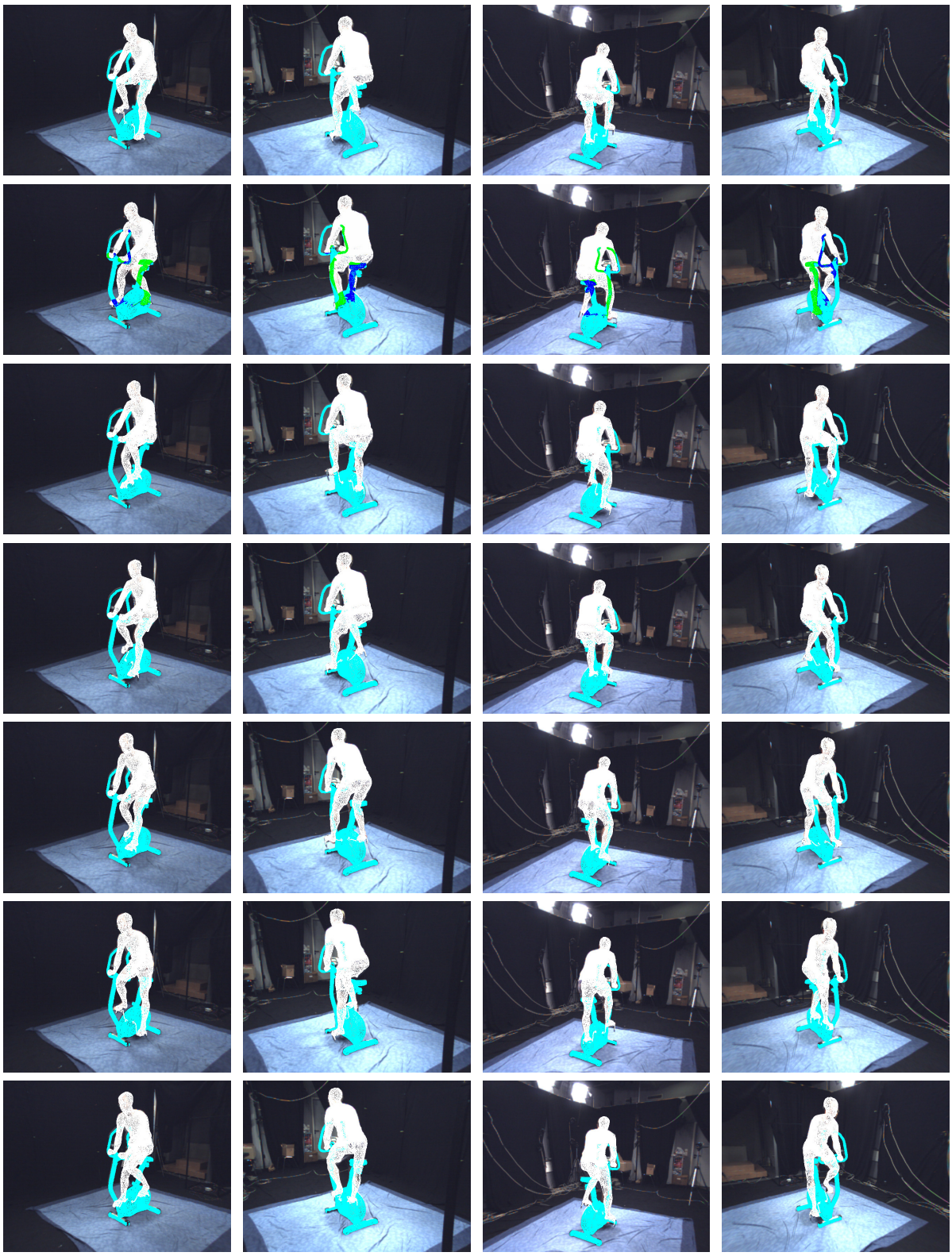


Figure 6.11.: Tracking result when tracking a static training bike and a cyclist using soft constraints. Shown are frame 210, 240, 270, 300, 470, 500, and 530 in each of the four available views. The green and dark blue regions in frame 240 illustrate the occluded regions of the bike and the cyclist, respectively.

7 Tracking Models with Multiple Components

"Much learning does not teach understanding."

Heraclitus (540 BC – 480 BC)

So far, we have demonstrated how to track multiple objects by dividing the image into several non-overlapping regions for each of which a separate PDF is estimated. In this chapter, we show that this concept can be extended naturally to use additional internal silhouettes. This is especially favourable for tracking articulated objects, but can also be beneficial for rigid objects.

As a simple example why additional internal silhouettes are advantageous consider the synthetic sequence shown in Figure 7.1. This sequence was created by projecting each rigid part of a human (modelled as kinematic chain) in a unique colour onto a fixed background. The only exception are the thighs, which have been omitted. Afterwards, uncorrelated Gaussian noise with standard deviation 15 was added. The movement of the person was modelled in such a way that the right arm of the person moves in front of its body. Thus, the arm is completely inside the object region, and there are no image-vertex point correspondences on this arm. In the resulting system of equations, there is no equation in which the angles corresponding to the right arm or the right hand occur. Thus, these angles cannot be estimated at all using the tracking approaches introduced so far as there is no cue for the pose of that part.

One possibility to cope with such problems is to use a bottom-up approach that first searches possible positions of different parts of the object. Out of these possible positions, one position for each part is picked such that all parts together result in the complete object. The appearance of the different object parts can be learned using Gibbs distributions [252] or using AdaBoost [175], for example. Another approach is to average pixel intensities inside parts of the object as done in [84].

Furthermore, it is possible to deal with such situations by using angle priors (see Section 5.4.2). This may introduce an undesired bias towards the used training data, though.

Figure 7.1.: Frame 0, 10, 20, and 29 of a synthetic sequence in which a human moves one arm in front of its body. Furthermore, the upper parts of the legs are transparent to simulate body parts whose appearances are indistinguishable from the background.

Here, we deal with such problems by using internal silhouettes. As only information provided by the input images is used, this will not introduce a bias. We first presented this approach in [239].

We first extend our tracking algorithm such that it can use multiple components per object. In Section 7.2, we explain how to create those components automatically. The chapter is finished with synthetic and real-world experiments in Section 7.3, and a conclusion in Section 7.4.

7.1. Multi-component Models

"I love modeling."

Brooke Burke (*1971)

The task we are going to solve in this chapter is very similar to the task from the last section. We only make the additional assumption that, for at least one object to track, there are some parts of this object whose appearance differs from the appearance of other parts of the object. This is a reasonable assumption since tracking is infeasible if two overlapping object parts have the same appearance. Even humans cannot distinguish equally looking parts after all.

Thus, the task to be solved now is given by:

Given:
– One projection matrix for each camera (see Section 2.6)
– One image sequence with domain $\Omega \subset \mathbb{R}$ for each camera
– Models for each object i to track as rigid object or kinematic chain with n_i joints (see Section 2.4)
– Approximate 3-D pose initialisation $\boldsymbol{\chi}_i \in \mathbb{R}^{6+n_i}$ for each object i in the first frame
Assumptions:
– Each object is always (partly) visible
– The appearances of objects and background differ
– Different object parts have different appearances
Task:
– Track the 3-D pose of all objects through all images

Let $M^j, j = 1, \ldots, l$ be l different components of one object model M. Note that we do not require each vertex of M to be in exactly one component M^j, i.e. there might be vertices that are in several components, or in none at all. In the synthetic example illustrated in Figure 7.1, the arm on the left side of the image could be one region, while the remainder of the visible body is the second component. That is, the thighs do not belong to any region in this case. As a real-world example in which some parts of the object model are omitted, consider a person wearing shorts whose colour matches those of the background. We presume that this splitting

is given for the remainder of this section. Details on two possible heuristics how a given object model can be split automatically are given in Section 7.2.

We follow the ideas explained in the last chapter, i.e. to use *visibility functions* to find out which portions of the object are actually visible in a certain image. To this end, we define $O^j(\boldsymbol{\chi}, \boldsymbol{x})$ as the set of 3-D points on M^j which are projected to the image point $\boldsymbol{x}$ when the current pose of the complete object is given by $\boldsymbol{\chi}$. Next, we define $d^j(\boldsymbol{\chi}, \boldsymbol{x})$ as the smallest distance of all points on M^j which are projected to $\boldsymbol{x}$ to the optical centre $\boldsymbol{C}$ as:

$$d^j(\boldsymbol{\chi}, \boldsymbol{x}) := d(O^j(\boldsymbol{\chi}, \boldsymbol{x}), \boldsymbol{C}) = \min_{\boldsymbol{y} \in O^j(\boldsymbol{\chi}, \boldsymbol{x})} \{d(\boldsymbol{y}, \boldsymbol{C}).\} \tag{7.1}$$

The visibility functions v^j are then given by:

$$v^j(\boldsymbol{\chi}, \boldsymbol{x}) := \begin{cases} 1 & \text{if} \quad d^j(\boldsymbol{\chi}, \boldsymbol{x}) = \min\limits_{j \in \{1,\ldots,l\}} \{d^j(\boldsymbol{\chi}, \boldsymbol{x})\} \quad , \\ 0 & \text{else} \qquad\qquad\qquad\qquad\qquad\quad . \end{cases} \tag{7.2}$$

For the background region, we set

$$v^0(\boldsymbol{\chi}, \boldsymbol{x}) := \prod_{j=1}^{l} \left(1 - v^j(\boldsymbol{\chi}, \boldsymbol{x})\right), \tag{7.3}$$

as the background is visible if no part of the object can be seen, and $c_0(\boldsymbol{\chi}, \boldsymbol{x}) := 1$ as the background is everywhere in the image where it is not occluded. The energy function for minimisation with multiple internal silhouettes is then given by:

$$E(\boldsymbol{\chi}) = -\sum_{j=0}^{l} \int_{\Omega} \Big(v^j(\boldsymbol{\chi}, \boldsymbol{x}) c^j_{\boldsymbol{\chi}}(\boldsymbol{x}) \log p^j \Big) \mathrm{d}\boldsymbol{x}. \tag{7.4}$$

Here, we used the notation $p^j := p^j(\boldsymbol{\chi}, \boldsymbol{x})$ for the estimated PDF of the j-th component M^j and $c^j_{\boldsymbol{\chi}}(\boldsymbol{x}) : \Omega \to \{0, 1\}$ for the indicator function that is one if and only if any point on M^j is projected to the image point $\boldsymbol{x}$.

If multiple objects are to be tracked, defining an appropriate energy function can be done in a similar way: For each of the i objects, we denote the number of parts considered by l_i, and the j-th component of the i-th object model as M^j_i. The corresponding PDFs and indicator functions are denoted by p^j_i and $c^j_{\boldsymbol{\chi}_i, i}(\boldsymbol{x})$, respectively. As before, the pose of the i-th object is called $\boldsymbol{\chi}_i$. Using the set $O^j_i(\boldsymbol{\chi}_i, \boldsymbol{x})$ of those points on M^j_i that are projected onto the image point $\boldsymbol{x}$ and the distance

$$d^j_i(\boldsymbol{\chi}_i, \boldsymbol{x}) := d(O^j_i(\boldsymbol{\chi}_i, \boldsymbol{x}), \boldsymbol{C}) = \min_{\boldsymbol{y} \in O^j_i(\boldsymbol{\chi}_i, \boldsymbol{x})} \{d(\boldsymbol{y}, \boldsymbol{C})\}, \tag{7.5}$$

we can define the visibility function v^j_i for the j-th component of the i-th object as

$$v^j_i(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_n, \boldsymbol{x}) := \begin{cases} 1 & \text{if } d^j_i(\boldsymbol{\chi}_i, \boldsymbol{x}) = \min\limits_{i' \in \{1,\ldots N\}} \min\limits_{j' \in \{1,\ldots,l_i\}} \{d^{j'}_{i'}(\boldsymbol{\chi}_i, \boldsymbol{x})\} \\ 0 & \text{else} \end{cases} . \tag{7.6}$$

For the background region, we set $l_0 = 1$, define the visibility function as

$$v_0^1(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_n, \boldsymbol{x}) = \prod_{i=1}^{n} \prod_{j=1}^{l_i} \left(1 - v_i^j(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_n, \boldsymbol{x})\right), \tag{7.7}$$

and set $c_0^1(\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_n, \boldsymbol{x}) = 1$

Then, the energy function to be minimised reads:

$$E(\boldsymbol{\chi}_1, \ldots \boldsymbol{\chi}_N) = -\sum_{i=0}^{N} \sum_{j=1}^{l_i} \int_{\Omega} \left[v_i^j c_{\boldsymbol{\chi}_i,i}^j(\boldsymbol{x}) \log p_i^j\right] \mathrm{d}\boldsymbol{x}. \tag{7.8}$$

Minimising the energy functions given in Equations (7.4) and (7.8) can be done similarly to the minimisation of Equation (6.7), i.e. when tracking multiple objects: First of all, each component of each object is projected simultaneously using OpenGL. In this projection step each component has a unique colour. After reading back the rendered image from the graphics card, the colour at an image point indicates which of the object components is visible at that specific point. This way, the visibility functions v_i^j for the current iteration are obtained. After that, each component is projected individually to compute the indicator functions $c_{\boldsymbol{\chi}_i,i}^j(\boldsymbol{x})$. Using these two functions, we compute the silhouettes of each region before and after considering occlusions, thereby gathering a set of 2-D silhouette points lying on both silhouettes of each component. Using a lookup-table created while projecting the object components, we determine the 3-D points on the object model corresponding to the 2-D silhouette points, resulting in a set of image-vertex point correspondences. The 2-D part is then shifted by a force vector perpendicular to the object boundary. To decide whether this force vector points towards the inside or the outside of the object component, the PDF of the current component and the next component in outside normal direction are compared in the same way as in the minimisation of the basic tracking approach presented in Section 5.2. Once the movement of all objects is below the requested threshold, the current poses are considered final for this frame, and tracking continues with the next frame.

7.2. Automatic Component Generation

"Just because it's automatic doesn't mean it works."

Daniel J. Bernstein (*1971)

Up to now, we have explained how to improve tracking by using models with multiple components. However, we have not explained how these components are chosen. This will be done in this section.

The tracking approach described in the last section only works well if the appearance of those model components M^j which are close together can be distinguished, as it compares the

estimated PDFs of these components. Thus, it is reasonable to assume that object parts with similar appearance should be in the same component.

Our automatic component generation algorithm starts by assigning each rigid part of the kinematic chain into a component of its own, resulting in as many components as object parts. This results in 14 components in the example shown in Figure 7.2: head, torso, three for each arm (upper arm, lower arm, and the hand), and three for each leg (thigh, lower leg, and foot). Note that it can also be useful to start with a different assignment, e.g. if a rigid part has two completely different appearances. We will see an example for this situation in Figure 7.8.

Once the initial assignment is known, a PDF is estimated for each component by projecting them into the image plane using the initial pose estimation, accounting for occlusions and estimating the PDF from the visible pixels. Next, the most similar components are merged into a new component, and the process is iterated until all components are different enough.

In this algorithm, it is necessary to compare two PDFs $p(i)$ and $q(i)$. For this task, there are several possible distance measures that can be used, e.g. summing the squares of the differences,

$$J_1(p,q) := \sum_i \left(p(i) - q(i)\right)^2, \tag{7.9}$$

or the *Kullback-Leibler divergence* [147] given by

$$J_2(p,q) := \sum_i p(i) \log \frac{p(i)}{q(i)}. \tag{7.10}$$

In our tests, we obtained the best results with the *Jensen-Shannon divergence* [297]. It is a symmetric and smooth variant of the Kullback-Leibler divergence and is given by

$$J_3(p,q) := \frac{J_2(p,M) + J_2(q,M)}{2}, \tag{7.11}$$

where M is the average of p and q, i.e. $M = \frac{p+q}{2}$. Once there is no pair of components for which $J_3(p,q)$ is smaller that a given merging threshold α, the algorithm has converged. The remaining components are fixed and used in our multi-component tracking algorithm. For all our experiments, we set $\alpha := 0.25$.

If some parts of the object are to be omitted, the background region can be added as an additional component in the initial assignment. All rigid model parts merged with the background region will not be present in a component while tracking. However, no leaf of the kinematic chain should ever be added to the background region, as it would be impossible to estimate the joint angles of these parts.

This process is illustrated in Figure 7.2. Using the top left image, the model of a woman wearing an orange shirt and dark trousers is to be divided automatically. In the third image in the upper row, the 14 initial components are shown. Note that the initial pose is inaccurate, and that there is no clear boundary between the joints, as explained in Section 2.4.2. Nevertheless, the merging steps performed are reasonable: The image in the second row show the pairwise differences between the PDFs estimated for the different components in the first three steps of the algorithm. In the first step, the difference between the PDFs corresponding to the two lower legs is smallest. Thus, these the components are merged into a new component called

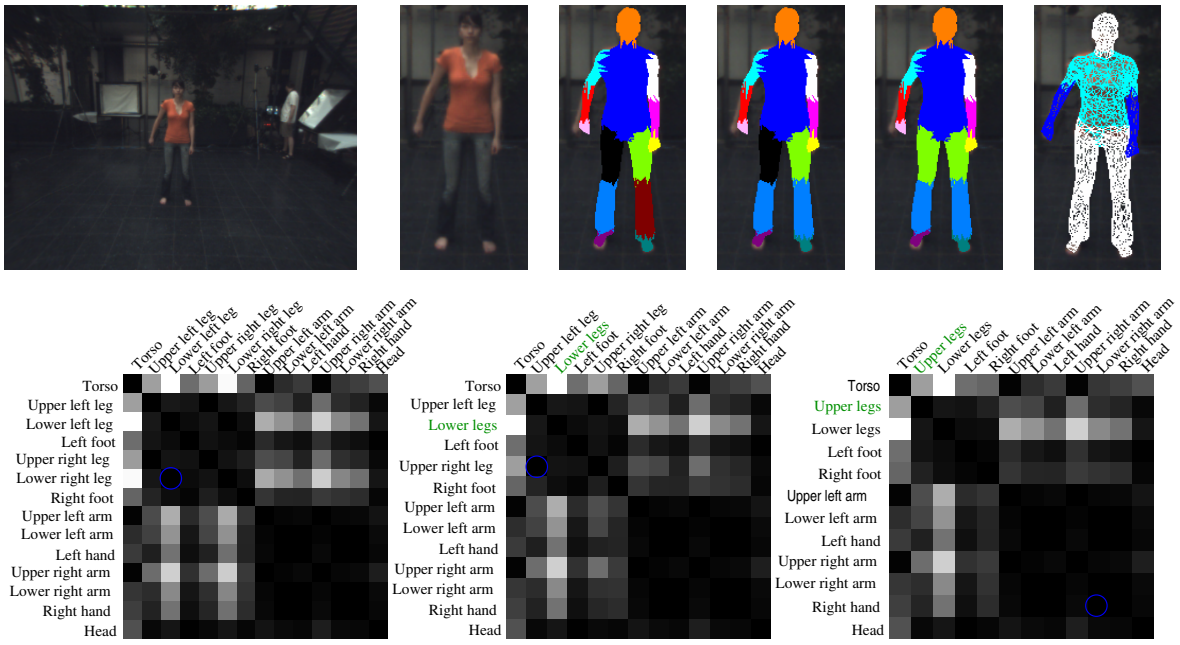


Figure 7.2.: Illustration of the automatic component generation algorithm. **Upper row, first two images**: Initial image, and magnification of the person to be tracked. **Upper row, third to fifth image**: initial components, and components after one and two merging steps, illustrated by projecting each component with a different colour. **Upper row, last image**: Components at the end of the merging steps after tracking the first frame. **Lower row, from left to right**: Matrix indicating the similarity of all pairs of components in the initial assignment, and after one and two merging steps. The darker a square, the more similar the corresponding components are. The blue circles indicate the regions merged next, and the names written in green indicate new components.

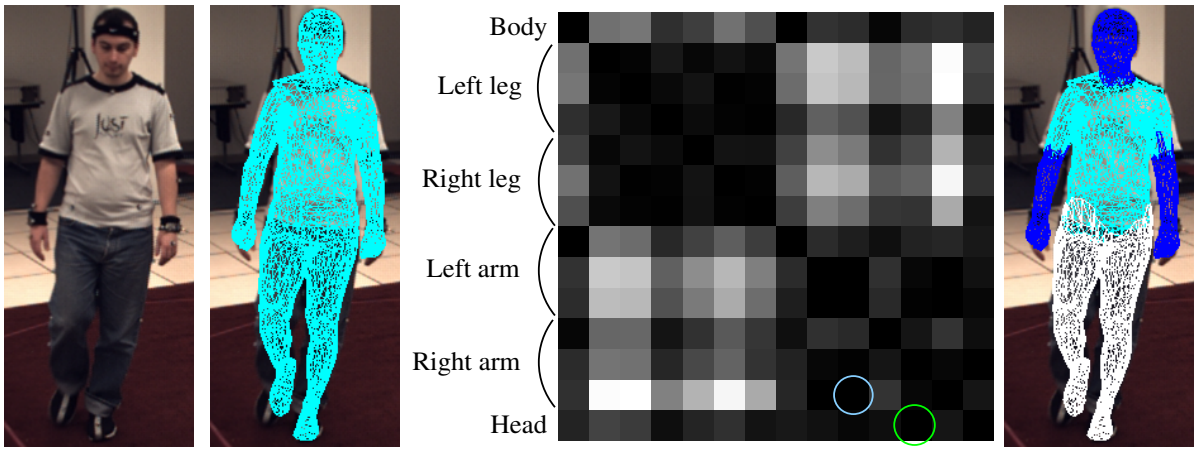


Figure 7.3.: Second example for our automatic component generation algorithm. Shown are a magnification of the input image (leftmost), the initial pose (left), the similarity matrix (right), and the final components (rightmost). The two circles in the similarity matrix indicate the first components merged, namely head and lower right arm (green) and both hands (blue).

Figure 7.4.: Tracking result for frame 0, 10, 20, and 29 of the sequence shown in Figure 7.1. For the shown tracking results, a model with two components was used: One component consists of the right arm, and the other of the remainder of the body except for the thighs.

"lower legs" in the second image. Next, the two thighs are merged, followed by the right forearm and the right hand. In the end, we get the three components shown in the last image of the first row in Figure 7.2: One for the torso and the upper arms, one for the legs, the feet and the head, and one for the lower arms including the hands.

This assignment might seem surprising at first, as it does not assign the upper and lower arms to the same component. When looking carefully at the image, one can see that the upper arms are partly orange and have partly the colour of skin. The same situation is given for the torso, while only skin can be seen for the lower arms and hands. Consequently, the upper arms in fact belong to the same component as the torso, according to their appearance.

As a second example, we consider sequence S4 of the HumanEva-II benchmark (see Section 5.5.2). The first similarity matrix and the final components obtained using our algorithm are shown in Figure 7.3. As can be seen, the results are similar to the first example: The only difference is that the head is assigned to different components. This happens because the head region mainly contains skin tone and a dark colour, which also appear at the trousers or the arms, respectively.

7.3. Experiments

"When your work speaks for itself, don't interrupt."

Henry J. Kaiser (1882 – 1967)

The first experiment for which we present results is the synthetic sequence shown in Figure 7.1. For this first initial experiments, the model components were created manually. As shown in Figure 7.4, all 30 frames were tracked without problems even though one arm is completely in front of the remainder of the body. Consequently, the basic tracking approach introduced in Section 5 is unable to track this sequence.

Further note that the upper legs are automatically tracked in this sequence even though they are not part of any component, as the lower legs and torso determine the position of the thighs. More precisely, the image-vertex point correspondences from lower legs and feet result in

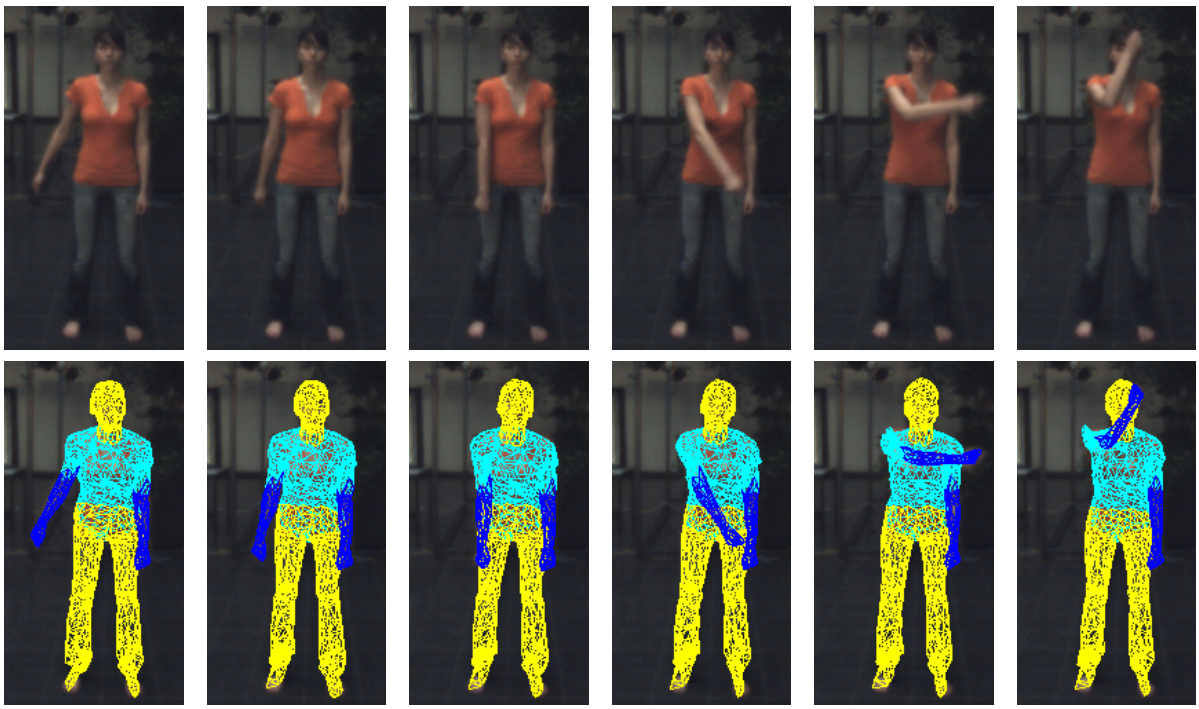


Figure 7.5.: **Upper row**: Frames 8, 18, 28, 38, 48, and 58 of the input image (cropped). **Lower row**: Tracking results for the same frames.

equations which include all joint angles of the legs. Thus, the position of the upper legs is also estimated when solving the resulting system of equations.

In Figure 7.5, a monocular real-world sequence with a person performing a similar motion is shown. As can be seen, the tracker found poses such that the projected object model matches the object seen in the image quite well for the first 58 frames. This changes in later frames, though.

When looking at the scene in frame 58 from other cameras not used for tracking (see Figure 7.6, one can immediately see why tracking fails after this frame: Theoretically, the exact distance of the model from the camera centre can be derived by the tracker. However, due to small errors in the model, camera aberrations, and other otherwise minor problems, there is a depth ambiguity. That is, the estimated model position is closer to the camera than it should be. As a result, the moving arm is actually estimated to be behind the head, resulting in wrong visibility functions. As the right arm is believed to be completely occluded, tracking fails.

Once again, we use sequence S4 from the HumanEva-II benchmark to evaluate our tracker. Example frames are shown in Figure 7.7. Using multiple regions, the average tracking error decreases from the 66.78mm obtained using soft floor constraints in Section 5.4.3 to 48.87mm when additionally using multiple regions. The improvements can be seen especially well around frames 200 and 400: Here, the legs flip without internal regions (see first images in Figure 5.19). Additionally, the arms are more accurate during the balancing part (compare fourth row of images in Figures 5.19 and 7.7), and the partial tracking failure of one leg from frame 930 to frame 1185 has disappeared (fifth row of images in these figures).

Figure 7.6.: Tracking results for frame 58 of the sequence shown in Figure 7.5 from two additional views not used for tracking. It can be seen that the 3-D pose is much worse than expected when looking only from the view shown in Figure 7.5.

As last experiment in this chapter, we revisit the Lego Duplo sequence from Figure 6.9 in Chapter 6. This sequence has three moving objects, two of which were tracked without noticeable problems. However, tracking results for the third object were inaccurate in some frames. Two example frames already shown in Figure 6.9 are magnified in Figure 7.8. When splitting one object into the blue and the green part depicted in that figure, tracking results get noticeably more accurate. This is demonstrated in Figure 7.8. In those frames not shown, tracking still works nicely.

7.4. Summary

This chapter introduced our approach for using multiple internal regions to improve the tracking of kinematic chains. This especially improves the tracking results in monocular scenes, but also helps in sequences with several views, as parts of the kinematic chain might be occluded in some views. The necessary splitting of the object model can either be done manually, or our automatic splitting approach can be employed.

In the next chapter, we introduce the last improvement of our tracking algorithm, namely an enhanced approach how to improve tracking in case of complicated and possibly varying backgrounds.

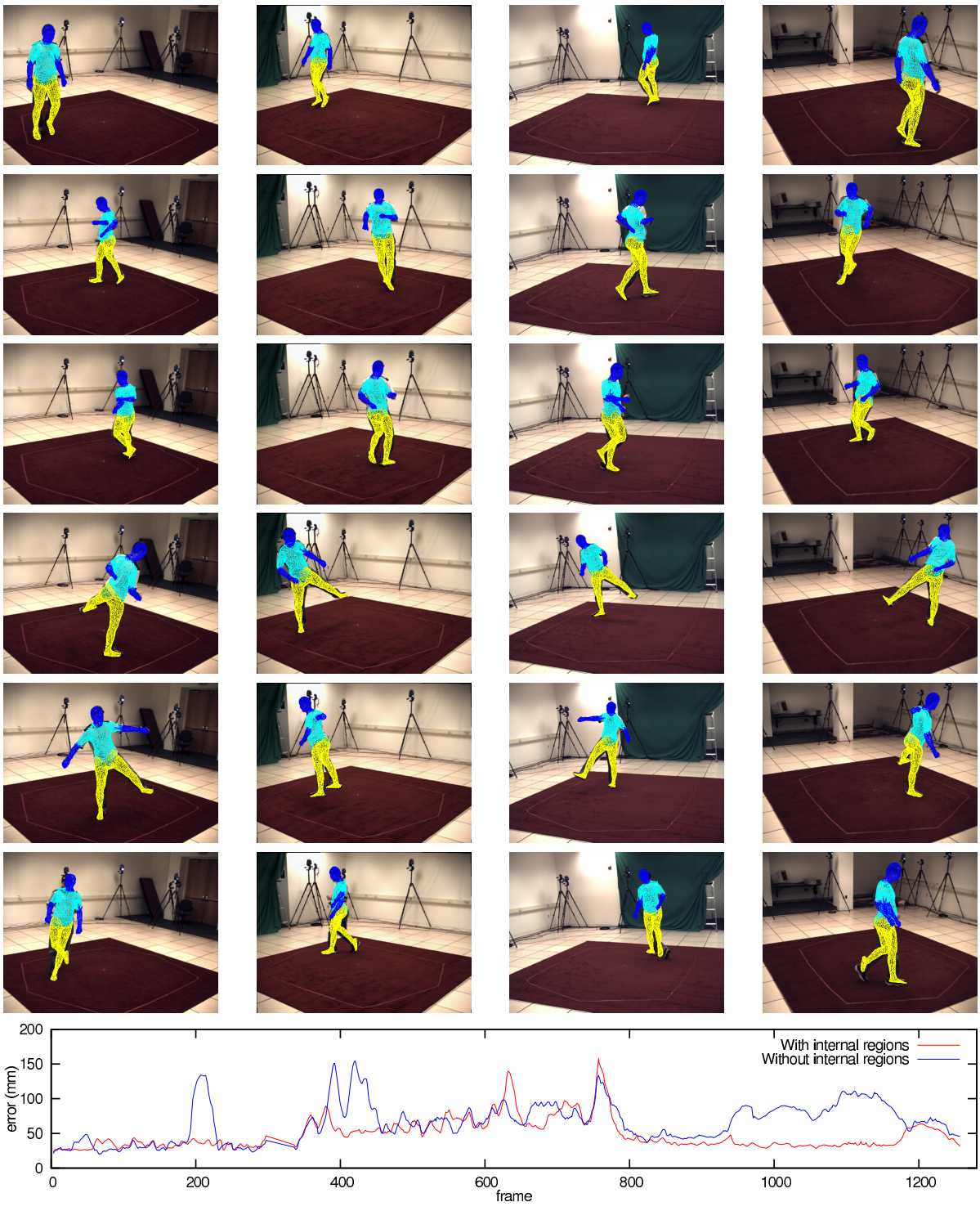


Figure 7.7.: Tracking results using multiple internal regions. **From top to bottom**: Tracking results in one frame of the walking part (frame 222), two frames in the jogging part (frames 444 and 666), two frames in the balancing part (frame 888 and 1110), and in the frame with the highest tracking error (frame 757). The plot at the lower end of the figure shows the average tracking error per frame achieved with multiple regions (red), and the best tracking error obtained without multiple regions (blue) shown in Figure 5.19 in Section 5.4.3.

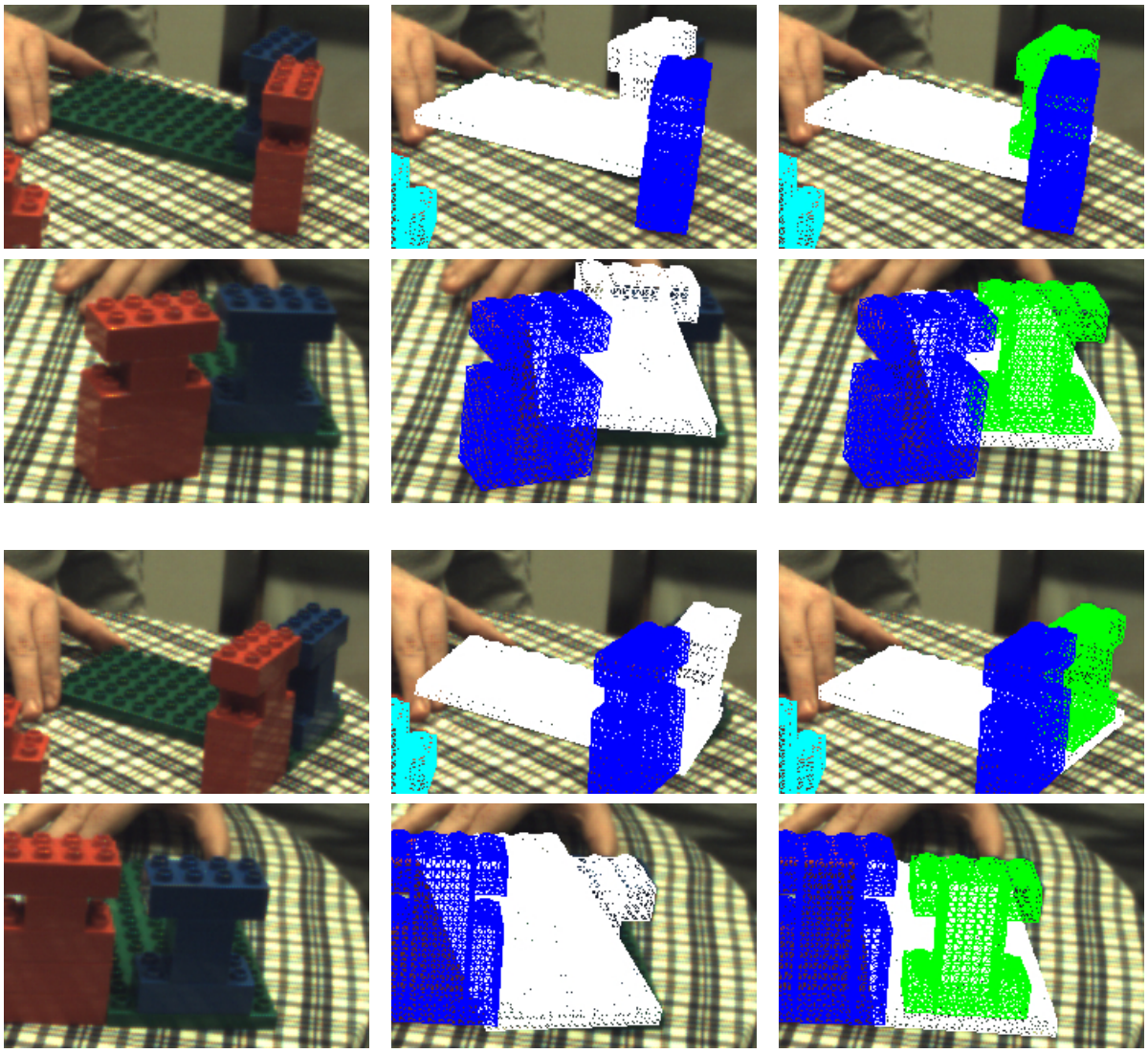


Figure 7.8.: Tracking experiment with multiple objects and multiple regions. **From left to right**: Input image, tracking result from Chapter 6 (see Figure 6.9), and tracking results using two internal components for one object. **Upper two row**: Frame 130 from both camera views. **Lower two row**: Frame 190 from both camera views.

8 Localised Mixture Models

"Essentially, all models are wrong, but some are useful"

George E. P. Box (*1919)

In the last chapters, we explained how to handle occlusions between different object parts or different objects by increasing the number of image regions considered. However, the more regions are used in such approaches, the more likely it is that the appearance of one object part is similar to some part of the background. In such situations, distinguishing fore- and background region may fail.

Two example images in which such situations may occur are shown in Figure 8.1. In the synthetic image to the left, a blue giraffe with red hooves is to be segmented or tracked. As half of the background is red, the red pixels in the hooves are classified as background points. Consequently, results will be inaccurate. Here, using the local Gaussian model presented in Section 2.7.2 is sufficient to solve this problem. However, this is not the case for the image on the right, which shows a man with black hair in front of a mostly black background.

A different local statistical model was proposed in [179], which blurs across discontinuities inside the region. This blurring can be quite significant, as a large neighbourhood is necessary for a reliable estimation.

Thus, we proposed in [236] to segment the background region into different subregions in each of which a separate PDF is estimated. Therefore, our approach uses local region statistics, i.e. it uses the position of the pixels in addition to their colour. Simultaneously, it

Figure 8.1.: Two examples for which modelling fore- and background by independent and identically distributed random variables would yield suboptimal tracking results. **Left**: Synthetic image of a giraffe with red hooves. As red points are more likely in the background region, the hooves are not classified correctly. **Right**: Black and white regions appear frequently in fore- and background.

utilises a large part of the image to accurately estimate the occurring PDFs. This is particularly necessary for non-parametric approaches such as kernel density estimations, see Section 2.7.2. Our approach will be called *localised mixture model*, and is abbreviated as *LMM*.

There is a similar concept in the context of background subtraction, namely so-called Gaussian mixture models. When using such a model, the estimations are performed along the temporal axis instead of in a spatial neighbourhood, resulting in accurate positional information [103]. An overview of both simple and complicated background models is given in [205]. It is also possible to combine learned statistics with a conventional spatially global model, see [265]. Learning the background in advance is still necessary for background subtraction. Thus, a background image or images without the objects to track and without strong motions must be available. This is not the case when using LMMs. There are also image labelling approaches that segment the background into several regions such as [246]. In contrast to such approaches, no learning step is necessary with LMMs.

There are many more approaches to distinguish foreground from background using techniques such as graph cuts [225, 57], active contours [24, 12, 139, 151], segmentation using K-means clustering [74], or Markov random fields [57]. The details of these approaches are beyond the scope of this thesis.

In the remainder of this chapter, we first introduce a localised background model for a known background image. Afterwards, this model is extended to the case of an unknown and possibly changing background in Section 8.2. Furthermore, we evaluate two segmentation algorithms for the necessary splitting of the background. After presenting tracking results both with and without localised models in Section 8.3, the chapter is concluded in Section 8.4 with a summary containing all tracking results presented in this thesis.

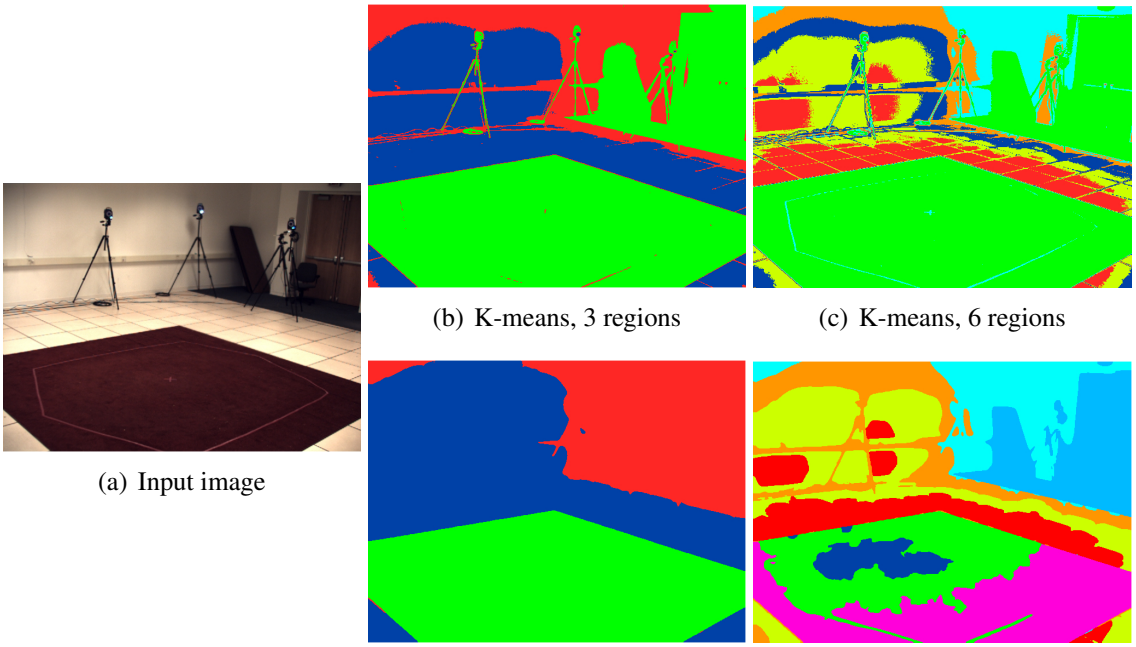
8.1. LMMs using Static Background Images

"In the midst of movement and chaos, keep stillness inside of you."

Deepak Chopra (*1946)

We first consider the case that a static background image is given. As we do not require the different background regions to correspond to different objects in the scene, this task is much easier than the top-level task of object-background separation. Thus, any segmentation algorithm which supports multiple regions may be used. Here, we test an extension of the level-set-based segmentation algorithm presented in Section 4.2, which has been introduced in [36]. Additionally, a simple segmentation using the K-means algorithm [164, 158] is evaluated. We employ the implementation from [92], which implements a fast variant of the K-means algorithm [74]. Note that this algorithm is also called *Lloyd's algorithm*, especially in computer science. As mentioned on page 149 in Section 9, it is used in [241] and [17] for stippling.

Notice the difference between these two segmentation algorithms: K-means only clusters the pixels in the image according to their appearance, but independent of their position in the



(a) Input image (b) K-means, 3 regions (c) K-means, 6 regions (d) Level-set-based, 3 regions (e) Level-set-based, 8 regions

Figure 8.2.: Example segmentations of the image shown on the left (a). The images on the upper row (b,c) show segmentation results with K-means clustering with three and six clusters, respectively, while level-set-based segmentation with two different parameter settings is used for the images on the lower row (d,e). As can be seen, K-means results in more fuzzy region boundaries.

image. The more sophisticated level-set-based approach considers multiple image scales, and includes a smoothness prior on the contour. Note that the number of regions a background image is segmented into is fixed when using the K-means algorithm, while the level-set-based segmentation algorithm utilises a homogeneity criterion to find the optimal number of regions, depending on a tuning parameter. Thus, the number of regions can vary between different frames or views.

The differences can be seen in Figure 8.2 in which the same background image is segmented with both approaches and with different parameters. Due to the boundary length constraint, the region boundaries are much smoother when using the level-set-based segmentation approach. Furthermore, we get a lot of small disconnected regions when using K-means. Especially when considering a moving background, as introduced in the next section, fuzzy boundaries and small regions can be harmful.

Once the background is segmented, a separate PDF p_i is estimated for each segment in the background image. These PDFs are then combined to obtain the LMM of the complete background image. More precisely, let $L(\boldsymbol{x}) : \Omega \mapsto \mathbb{N}$ denote the segmentation that maps each point in the background image to the number of its region, and let s be the set of image

features used for tracking. The PDF of the background is then given by

$$p(\boldsymbol{x}, s(\boldsymbol{x})) = p_{L(\boldsymbol{x})}(s(\boldsymbol{x})). \tag{8.1}$$

It is reasonable – but not obligatory – to use the same density models for the subregions that were used for segmentation. For K-means, this results in Gaussian distributions with fixed variance (see Section 2.7.2):

$$p_j^{\text{kmeans}}(s) \propto \exp\left(-\frac{(s-\mu_j)^2}{2}\right), \tag{8.2}$$

where μ_j is the cluster centre of cluster j. For the level-set-based segmentation approach, a kernel density estimator is employed (see Section 2.7.2):

$$p_j^{\text{levelset}}(s) = K_\sigma * \frac{\sum_{\boldsymbol{x}\in\Omega_j} \delta(s, I(\boldsymbol{x}))}{|\Omega_j|}, \tag{8.3}$$

where δ is the Dirac delta distribution and K_σ is a Gaussian kernel with standard deviation $\sigma = \sqrt{30}$. For tracking, these PDFs can be computed once at the beginning of the sequence and the LMM can then be used as PDF p_0 in one of the Equations (6.7), (7.4), or (7.8). Note that p_0 is fixed for all frames, while the PDFs for the different object parts are still recomputed once each frame.

8.2. LMMs with Varying Backgrounds

"To be suspicious is not a fault.
To be suspicious all the time without coming to a conclusion is the defect."

Lu Xun (1881 – 1936)

The approach presented in the last section can only be used if there is a static background image given as input data. However, this is usually not the case. Especially in outdoor scenarios, the background constantly changes due to moving plants, clouds, people passing by, shadows, changing lighting conditions, or camera motion. Note that the last problems can also appear in indoor environments.

To extend LMMs to non-static images and images in which the object is already present, we use the facts that the background usually changes slowly between successive frames. Note that, at the time when the new PDFs are estimated, tracking the current frame is already finished when information reusing as introduced in Section 5.4.1 is applied. Thus, we know where the object currently is and can exclude the occluded regions from the segmentation step.

Example segmentation results for which the object region is omitted are shown in Figure 8.3. The image shown is frame 42 of sequence S4 of the HumanEva-II benchmark, which was already used several times in earlier chapters. As can be seen, both approaches still yield

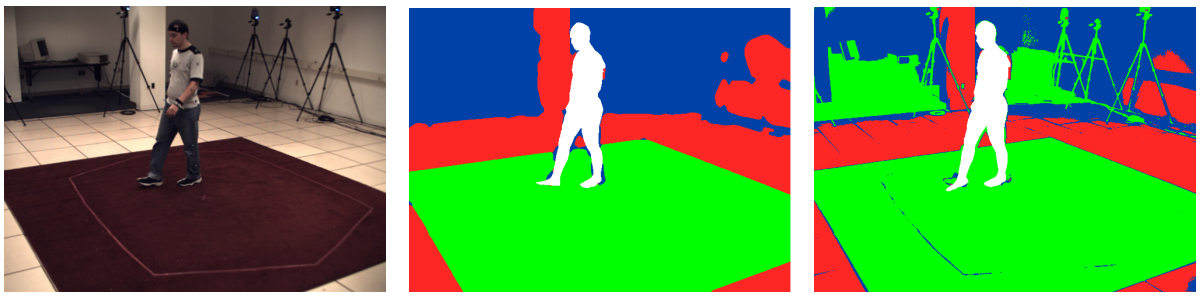


Figure 8.3.: Here, segmentation results for frame 42 of the HumanEva-II sequence S4 obtained with the level-set-based segmentation approach (middle) and the K-means-based segmentation approach (right) are presented. The white region is occluded, every other colour denotes a segmented region.

reasonable segmentations. However, there are some faults close to the object due to tracking inaccuracies and due to the fact that a human cannot be modelled accurately using kinematic chains. This is especially visible in the right foot in the right image, for which tracking results are quite bad.

In those parts of the background image in which the object is not visible, the LMM is created as before. However, as the object's position in the image will change during tracking, it is necessary to evaluate the PDF of the background region at points occluded while the segmentation was performed. At those points, we use the PDF of the subregion closest to this point. Thus, the LMM is given by:

$$p(\boldsymbol{x}, s) = p_{j^*}(\boldsymbol{x}) \qquad \text{with} \qquad j^* = \underset{j}{\operatorname{argmin}} \left(\operatorname{dist}(\boldsymbol{x}, \Omega_j) \right), \tag{8.4}$$

where Ω_j is the jth subregion computed by the segmentation step:

$$\Omega_j := \{\boldsymbol{x} \in \Omega | L(\boldsymbol{x}) = j\}. \tag{8.5}$$

Although this is just an approximation of an ideal segmentation, we will see in the next section that this approximation is good enough to obtain much better tracking results than with a global PDF.

In order to account for small tracking inaccuracies, and thus dubious segmentation results close to the object, it can make sense to increase the area of the occluded background region before segmentation, e.g. by dilating the occluded region. However, this has not been done for the experiments shown in this thesis.

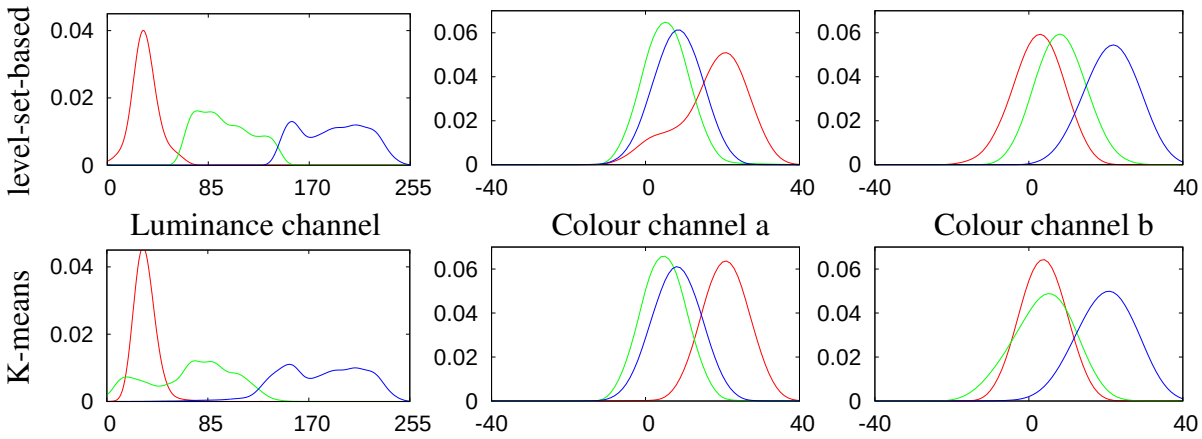


Figure 8.4.: PDFs for the three subregions in each of the segmented backgrounds shown in Figure 8.3. Shown are the PDFs for the luminance channel (left), and the two colour channels (middle and right) for the level-set-based approach (top) and for K-means (bottom).

8.3. Experiments

"All life is an experiment. The more experiments you make the better."

Ralph Waldo Emerson (1803 – 1882)

Once again, we use sequence S4 from the HumanEva-II benchmark to evaluate the new tracking approach. All experiments performed with this sequence are summarised in Table 8.1. The first three lines show the results obtained with the basic tracking approach introduced in Sections 5.1, the improved version using floor constraints (see Section 5.4.3), and the extension to multiple internal components explained in Section 7.1. The remaining rows show results using the localised mixture models explained in this chapter, with and without background knowledge, and with different segmentation algorithms. More details will be given in the remainder of this section.

Since we noticed an increased tracking error around frame 1200 without any visible reason for this behaviour (see graph in Figure 7.7), we decided to use only the frames up to and including frame 1170 for the comparison in this section. Thus, the numbers given in Table 8.1 which are used as comparisons in this section slightly differ from those values given in previous chapters. The frames 298–325, for which the ground truth is corrupt, are still omitted.

As a first experiment, we use one of the available static background images for each view to segment the background, as shown in Figure 8.2 and explained in Section 8.1. For both segmentation approaches, we optimised the involved parameters such that the best tracking results are obtained. That is, the number of clusters is optimised for K-means, and the parameter determining the additional energy for each region is optimised for the level-set-based approach. In contrast to the previous chapters, the inaccurate silhouette images from background subtraction (see Figure 5.17) are not used as additional features.

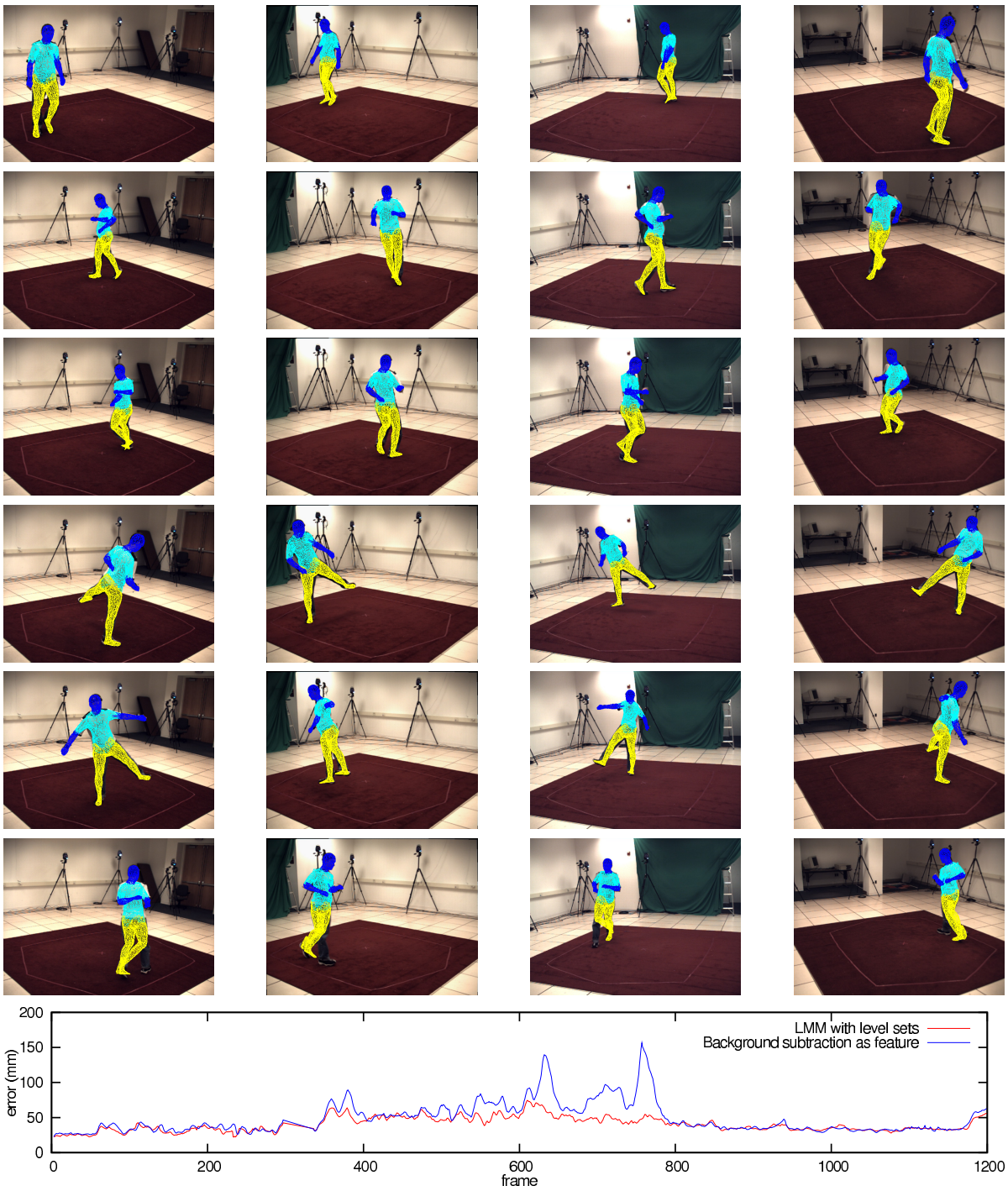


Figure 8.5.: Tracking results using LMMs instead of background subtracted silhouette images. **From top to bottom**: Tracking results in one frame of the walking part (frame 222), two frames in the jogging part (frames 444 and 666), two frames in the balancing part (frame 888 and 1110), and in the frame with the highest tracking error (frame 610). The plot at the lower end of the figure shows the average tracking error per frame when using a localised region model of the known background image (red) instead of using silhouette images obtained by background subtraction as additional feature. Again, the best tracking error obtained before (see Figure 7.7 in Section 7.3) is shown in blue.

Using LMMs with K-means (fourth row in Table 8.1), an average error of 49.61mm is obtained, which is slightly higher than the average error of 48.63mm obtained with the method from the last chapter (third row). However, the maximal tracking error dropped from 156.5mm to 114.2mm. With the level-set based approach, the results are even much better: The average tracking error drops to 40.25mm, and the maximal tracking error drops to 74.5mm, as shown in the fifth row of Table 8.1. A comparison of the tracking error for each frame is shown in Figure 8.5. One can see that the jogging part between frames 370 and 780 is tracked much better when using level-set-based LMMs than when using silhouette images obtained from background subtraction as additional features.

Next, we evaluate if using both silhouettes extracted from background subtraction and LMMs computed on given background images further improves the tracking results. Thus, we performed experiments in which LMMs and background subtraction are used simultaneously. The corresponding experiments are given in the last two lines of the first block in Table 8.1 (lines six and seven).

When using K-means in the LMMs, tracking results are further improved when additionally using silhouettes extracted with background subtraction. With the level-set based approach this is not the case. However, this approach still yields better results than the one using K-means. It seems that the erroneous silhouettes obtained by background subtraction (see Figure 5.17) can only improve the results if an improper segmentation algorithm is used in LMMs. Note that we have claimed in [236] that using both LMMs and silhouettes always improves the results. However, we merely got this impression due to a minor bug that has been corrected after the paper was published. Figure 8.6 shows a comparison per frame when using LMMs with and without background subtraction as additional feature (lines five and seven in Table 8.1).

Finally, we performed several experiments to evaluate our tracking algorithm without using the background images or any information derived from them. The results from those experiments are summarised in the second block of Table 8.1. That is, we use the LMM variant designed for varying, unknown backgrounds from Section 8.2 and did not perform background subtraction to obtain additional silhouettes images. For these experiments, the same internal regions as before have been used.

In such a situation, our tracking approach fails when only using a simple Parzen model, see line eight of Table 8.1. Nevertheless, our approach using LMMs with unknown background images (lines nine and ten) yields results which are very close to those obtained with background subtraction (line three), especially when using the level-set-based approach: The average error is only 0.17mm worse (48.80mm versus 48.63mm), while the maximal error is even significantly slower (82.8mm instead of 157mm). This shows that using LMMs significantly stabilises tracking if no background images are given.

Note that the tracking results for this sequence show that the results obtained with our algorithm are very accurate compared with other approaches from the literature.

In [113], segmentation and feature learning is used for monocular pose tracking. The tracking errors obtained range are in the range 111-146mm for the walking part, and 106-155mm for the jogging part, depending on the view used. No results for the balancing part are given, as balancing was not part of the training set.

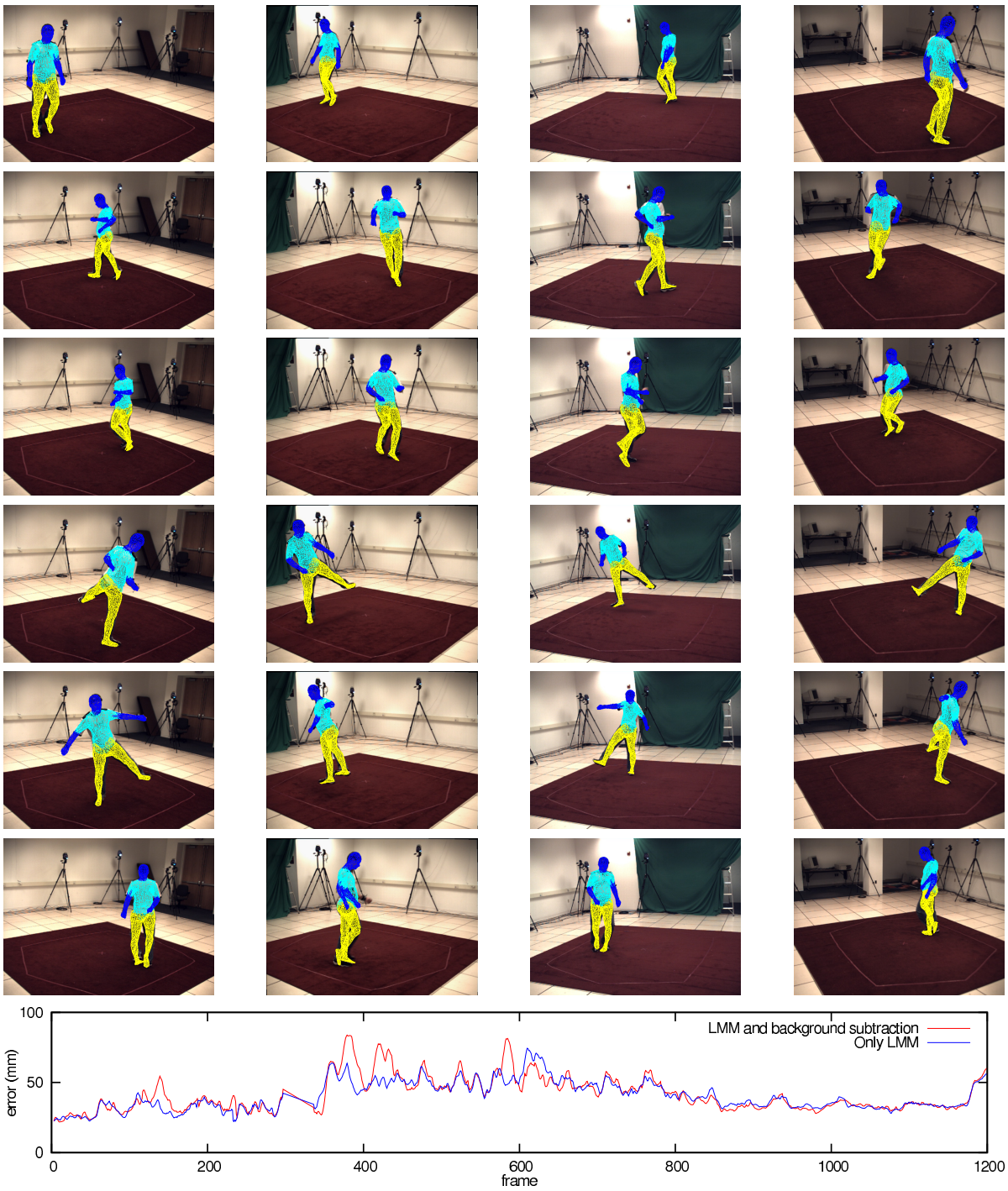


Figure 8.6.: Tracking results when using both LMMs and silhouette images from background subtraction. **From top to bottom**: Tracking results in one frame of the walking part (frame 222), two frames in the jogging part (frames 444 and 666), two frames in the balancing part (frame 888 and 1110), and in the frame with the highest tracking error (frame 379). The plot at the lower end of the figure shows the average tracking error per frame when using both a localised region model of the known background image all well as silhouette images obtained by background subtraction (red). The results obtained when using only LMMs (see Figure 8.5) is shown in blue.

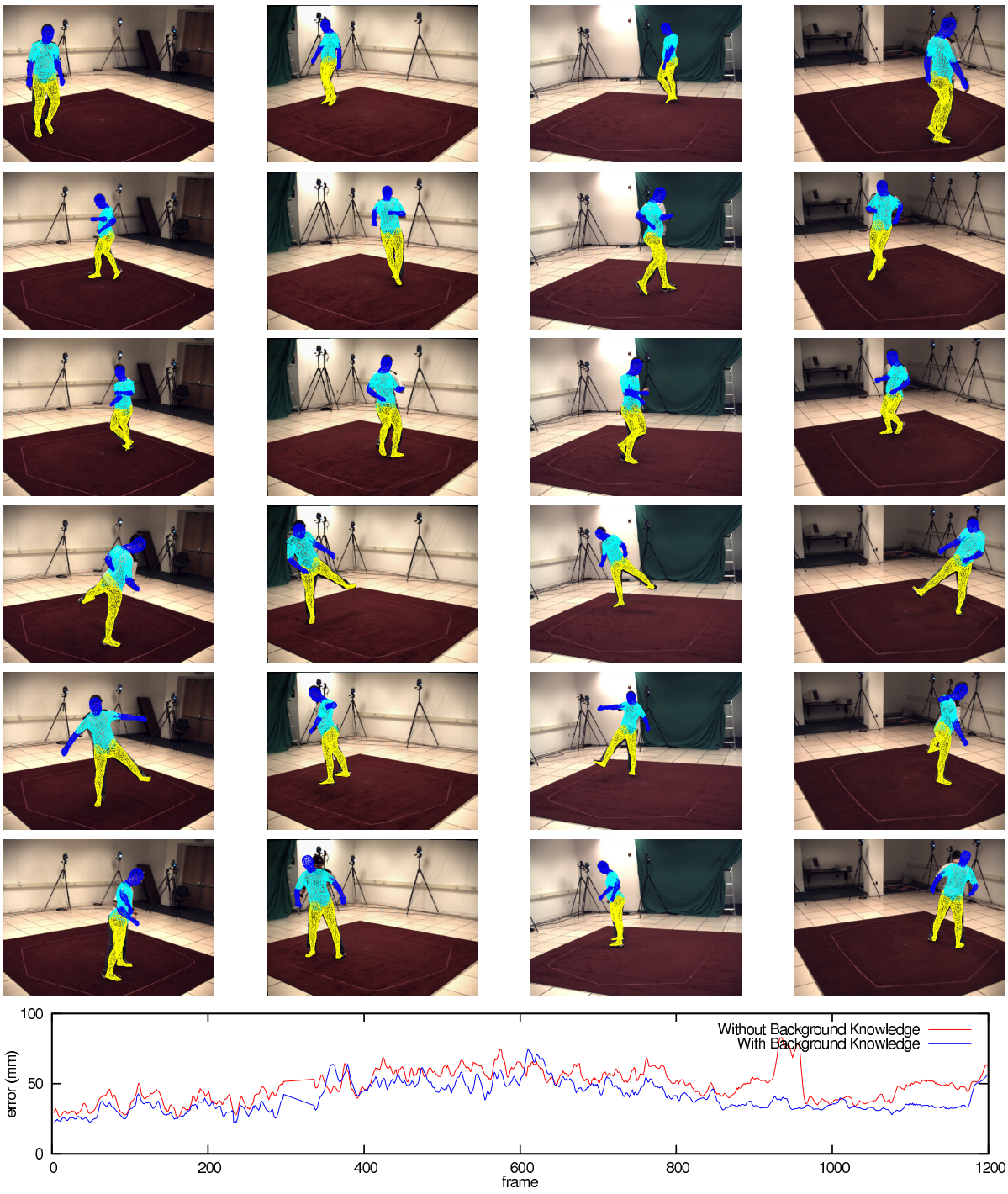


Figure 8.7.: Tracking results using LMMs without knowledge about the background image. **From top to bottom**: Tracking results in one frame of the walking part (frame 222), two frames in the jogging part (frames 444 and 666), two frames in the balancing part (frame 888 and 1110), and in the frame with the highest tracking error (frame 933). The plot at the lower end of the figure shows the average tracking error per frame when using both a localised region model of the known background image as well as silhouette images obtained by background subtraction (red). The best tracking error obtained when using background knowledge (see Figure 8.5) is shown in blue.

Table 8.1.: Summary of all tracking results of sequence S4 of the HumanEva-II benchmark. The first seven results use background images in tracking, either by using silhouette images obtained from background subtraction as additional features (Lines 1, 2, 3, 6, and 7), or by using LMMs with fixed background image (Lines 4–7). For the last three results, the background images are not used, nor is any information derived from them. Shown are the number given to the method for referencing (No.), a short description of the method itself (Method), the average tracking (Avg. error), the variance of the error (Variance), the maximal tracking error obtained (Max. error), and the figures in which results from the experiments are shown (Figure). All numbers are in millimetres. As explained in the text, only the frames up to frame 1170, but excluding frames 298 to 325, were used to compute the average tracking error.

No.	Method		Avg. error	Variance	Max. error	Figure
1	Basic approach		72.78	756.60	154.79	5.18
2	Floor constraints		66.86	847.97	154.32	5.19
3	Internal regions		48.63	511.23	156.53	7.7
4	LMM (K-means)		49.61	485.69	114.17	–
5	LMM (LS segm.)		40.25	107.51	74.46	8.5
6	LMM (K-means)	+ BG subtraction	42.75	179.75	92.56	–
7	LMM (LS segm.)	+ BG subtraction	41.44	154.70	83.78	8.6
8	Parzen Model		451.18	24706.77	728.45	–
9	LMM (K-means)		52.66	603.67	162.70	–
10	LMM (LS segm.)		48.80	127.15	82.82	8.7

Table 8.2.: Comparison of results from our tracking approach with different approaches from the literature. Given are the average tracking error of Sequence S4 of the HumanEva-II benchmark for the walking part, the jogging part, the balancing part, and the complete sequence. All results are given in millimetres.

Paper	walking	jogging	balancing	Total
Poppe [207]	121–183	112–156	185–266	148–204
Howe [113]	111–146	106–155	–	–
Cheng and Trivedi [50]	161	259	134	177
Husz *et al.* [118]	180	126	161	157
Corazza *et al.* [55]	80	–	–	–
Brubaker *et al.* [38]	52	–	–	–
Gall *et al.* [89]	28	37	31	32
Our approach	31	52	37	40

Hierarchical partitioned particle filters specially designed for human tracking are employed in [118]. Using a motion model and prior knowledge of the structure of the human body, average tracking errors of 180mm (walking part), 126mm (jogging part), and 161mm (balancing part) are obtained.

A volumetric reconstruction, combined with a single Gaussian mixture model for each rigid object part, was used in [50]. The mean tracking error reported is 177mm. The example-based pose estimation in [207] uses histograms of oriented gradients as image descriptors to estimate the pose from a single view. Depending on the view used, the average tracking error lies between 148mm and 204mm.

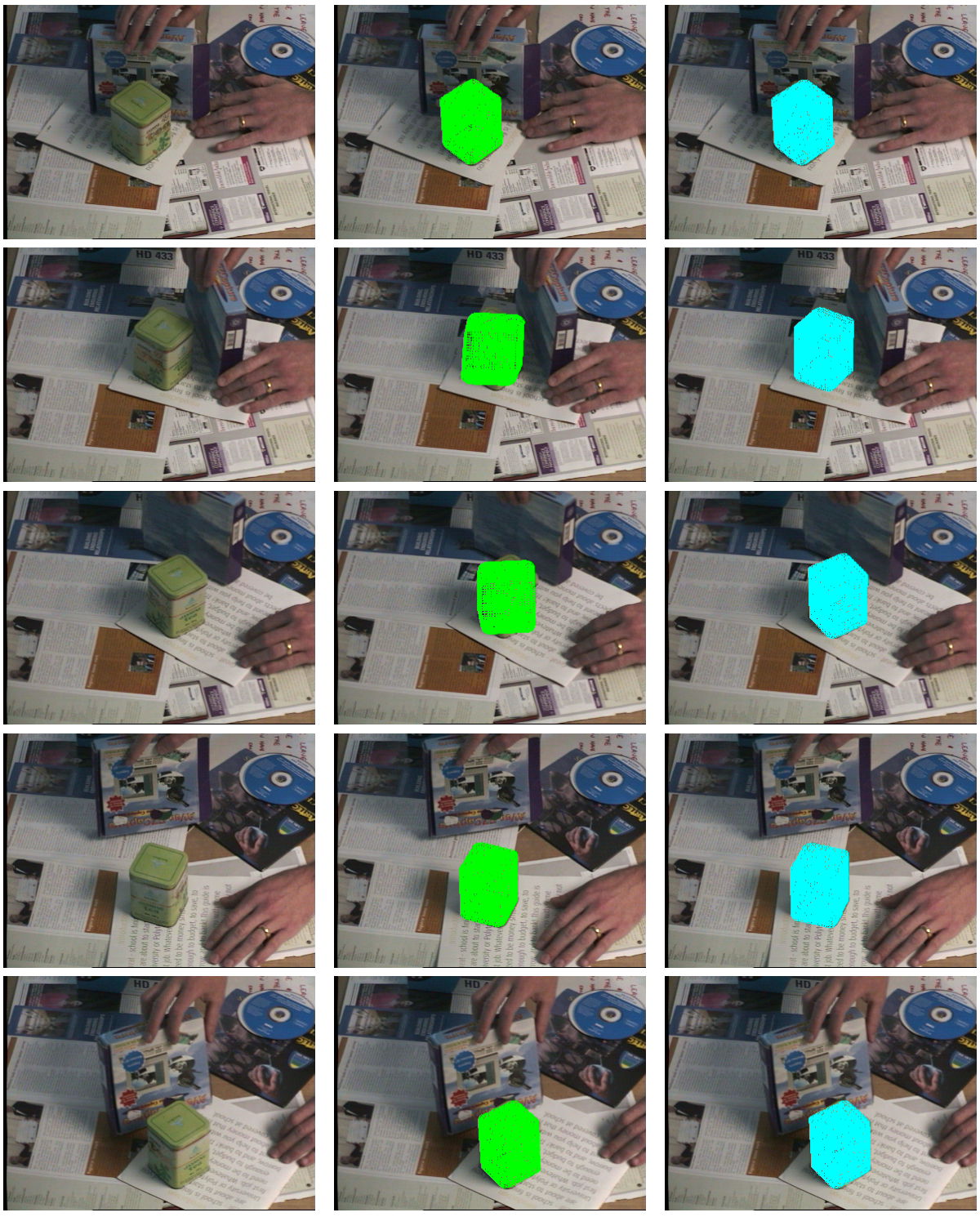
Corazza *et al.* report a mean tracking error of 80mm [55]. However, they used only the first 150 frames, as tracking became less robust and started failing afterwards. In [38], an average error of 54mm is obtained by employing a physical model of walking and two of the four views. However, only the lower body part is tracked, and tracking was only done for frames 15 to 350, i.e. for the walking part of the sequence.

An impressive average tracking error of 32mm was reported in [89] by combining a global optimisation based on interacting simulated annealing with a particle filter, followed by a local optimisation similar to the approach presented in this thesis. Due to the global optimisation, a tracking failure in one frame is very unlikely to propagate into the following frames. Obviously, this stabilises tracking considerably. Such an approach can also be used as first step in our approach. One should note that using such a global approach is quite costly, and would increase the run time of our tracking algorithm by a factor of five. Also note that we also receive very good results without using the provided background images, while the method by Gall *et al.* requires background subtracted images for its global optimisation step.

To verify that LMMs also work for non-static backgrounds, we show tracking results with a partially moving background in Figure 8.8. Note that there are no background images available for this sequence. Thus, background subtraction is impossible. As can be seen in the middle column, the algorithm fails to track a 90° rotation of the tea box around frame 90 if the Parzen model is used. However, the LMM can deal with the changing background and tracks the tea box successfully.

When adding uncorrelated Gaussian noise with standard deviation 10 before tracking this sequence, the difference is even more striking. This is demonstrated in Figure 8.9: With the Parzen model, tracking completely fails beginning with frame 75, while the LMM can track a longer part of the sequence.

Even with LMMs, the tracker still fails to track the rotation around frame 200. This is due to two facts: We are dealing with a monocular setting, and two planes of the tea box are projected (approximately) to lines in the image. In such a situation, rotating the object a little more makes this plane appear or disappear. As the points on the plane are not part of the silhouette yet, they do not influence the pose. It was thus proposed in [211] to use also the furthest points projected onto the silhouette in such a situation. Using this extension, tracking the tea box will probably succeed. This is part of our future work.



Input images Parzen model LMM

Figure 8.8.: A monocular sequence of a tea box with partially varying background (left) tracked with a global Parzen model (middle, green) and with LMMs based on level-sets (right, cyan). Shown are, from top to bottom, the frames 20, 80, 90, 160, and 230. In contrast to the LMM, the global Parzen model misses a $90°$ rotation of the tea box around frame 90.

Table 8.3.: Summary of all experiments shown in this thesis. Shown are the number of views used for pose tracking (Views), the number of objects to track (Objects), the model representation ('P' for patch-based models, 'T' for triangle-based models, Type), the total number of vertices of all models used (Vertices), the number of unknowns that are to be estimated per object (Unknowns), the image resolutions (Resolution), the length of the sequence (Frames), whether the images are in colour or grey-valued (Colour), whether texture features were used for tracking (Texture), if the 'old' PDFs from the previous frame were used (PDFs), and in which figures results from this sequence are shown (Figure(s)).

Sequence	Views	Objects	Type	Vertices	Unknowns	Resolution	Frames	Colour	Texture	PDFs	Figure(s)
Tea box	2	1	T	1694	6	384×288	395	yes	yes	new	5.7,5.8,5.9
Tea box 2	2	1	T	1694	6	384×288	395	yes	no	new	5.8,5.9
Giraffe	1	1	P	1183	6	384×288	84	yes	yes	old	5.10
Teapot	2	1	P	9764	6	376×284	347	yes	no	old	5.11
Puncher	1	1	T	5053	6	376×284	171	yes	no	old	5.12
Aeroplane	1	1	T	86922	6	640×480	48	yes	no	old	5.13
Paul	4	1	P	6201	26	500×380	98	no	no	old	5.14
Cart	4	1	P	5671	27	500×380	464	no	no	old	5.15
Flip	4	1	P	5671	27	500×380	260	no	no	old	5.16
Bike	4	2	T	20434	6/28	640×480	550	yes	no	old	6.11
Sync 1	2	2	T	6747	6/6	384×288	20	yes	no	old	6.1, 6.6
Sync 2	2	3	P,T	16511	6/6/6	384×288	40	yes	no	old	6.7
Duplo1	2	3	P	8578	6/6/6	640×480	264	yes	no	old	6.8
Duplo2	2	3	P	7374	6/6/6	640×480	380	yes	no	old	6.9, 7.8
Duplo3	2	2	T	5192	6/6	640×480	110	yes	yes	old	6.10
Palma	1	1	T	999	30	640×480	30	yes	no	old	7.1, 7.4
Arm	1	1	T	999	30	640×480	59	yes	no	old	7.2, 7.5, 7.6
Tea box mono	1	1	T	1694	6	384×288	277	yes	no	old	8.9, 8.8
HumanEva	4	1	T	2502	28	656×490	1257	yes	no	old	5.18,5.19,7.7 8.5,8.6,8.7

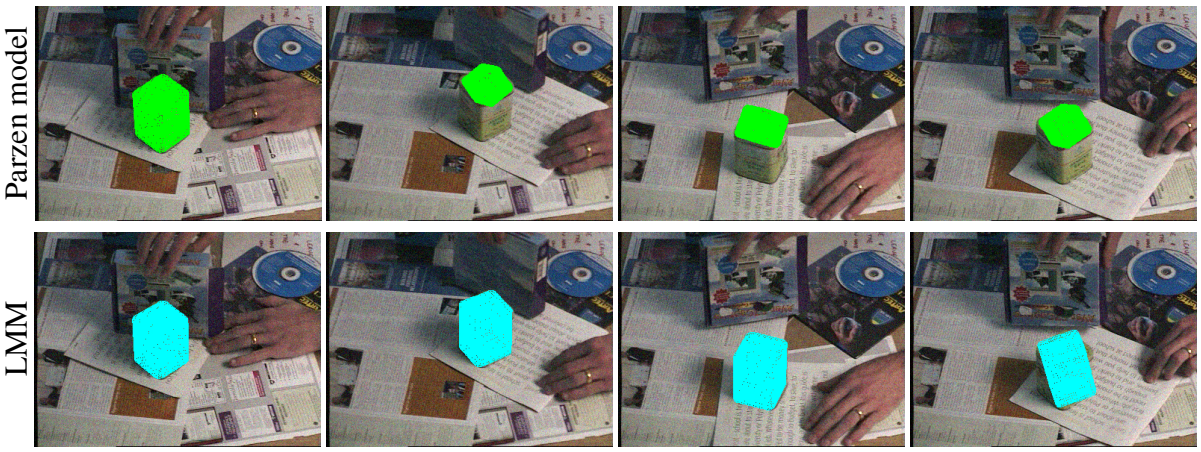


Figure 8.9.: In this figure, uncorrelated Gaussian noise with standard deviation 10 was added before tracking the sequence from Figure 8.8. Shown are, from left to right, the frames 20, 90, 160, 200. While tracking with the Parzen model fails quickly, the LMM can track the sequence quite long.

8.4. Summary

In this chapter, we presented our localised mixture model, and showed that this model can noticeably improve the obtained tracking results. We further illustrated that our model can use the background image to improve the tracking results, but even works very well in case of unknown or varying background images.

Table 8.3 shows a summary of all sequences tracked in this thesis. As can be seen, the presented tracking approach can deal with both monocular and multi-view sequences, and can track several completely different objects with different model representations in the same sequence. Furthermore, it can deal with simple and complex models, can be used for rigid and articulated models with a large amount of unknown joint angles, use different image features, and incorporate soft constraints and angle priors.

This concludes the part of this thesis that deals with 3-D pose tracking. The next chapter introduces the third topic discussed in this thesis, namely the novel halftoning algorithm we developed.

9 Electrostatic Halftoning

"I like physics. I think it is the best science out of all three of them, because generally it's more useful. You learn about speed and velocity and time, and that's all clever stuff."

Thomas Andrew "Tom" Felton (*1987)

In Chapter 2.7.2 we have seen how to estimate a probability density function (PDF) from random samples drawn from this PDF. In *sampling*, the inverse problem must be solved: Given a PDF, the task is to choose a predefined number of sample points such that these points best represent the underlying PDF. This problem arises in different tasks such as *stippling* [241], object placement [63], multi-class object placement [286], ray-tracing [203], image-based lighting using high dynamic range images [142], geometry processing [269], or numerical integration such as Quasi-Monte Carlo methods [105, 54].

The general problem to approximate images with continuous tones using a finite number of dots is called *halftoning*. Here, only few different colours may be used for each dot. Common examples are the preparation of grey-scale images for printers or fax machines, which can only create black dots onto a white piece of paper due to technical reasons. Similarly, offset printers used to create colour images also use dots with a limited amount of pure colours, typically cyan, magenta, and yellow. Black is often added in addition as a fourth colour. Through mixing dots with these colours, a human viewer gets the impression to see a much larger amount of tones.

Typically, but not always, these dots are required to be on some regular grid, which is usually quadratic. In this case, the specific problem is called *dithering*, see Figure 9.1. In one of the first methods proposed for dithering, Goodall first added Gaussian noise to an image and used a simple thresholding afterwards [97]. This can also be regarded as adapting the threshold instead of adding noise. The next notable approach, which uses predefined threshold matrices,

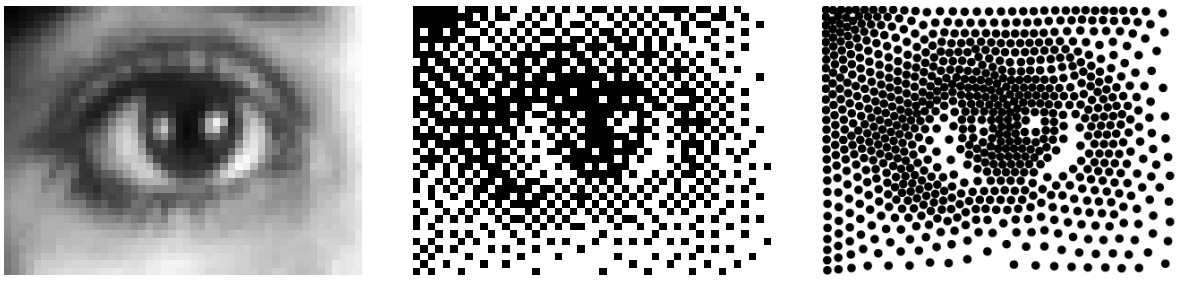


Figure 9.1.: Illustration of dithering and stippling results. **Left**: Initial image. **Middle**: Result using our dithering algorithm. **Right**: Result using our stippling algorithm.

$$\frac{1}{5}\begin{pmatrix}1 & 3\\ 2 & 4\end{pmatrix} \quad \frac{1}{17}\begin{pmatrix}1 & 9 & 3 & 11\\ 13 & 5 & 15 & 7\\ 4 & 12 & 2 & 10\\ 16 & 8 & 14 & 6\end{pmatrix} \quad \frac{1}{17}\begin{pmatrix}1 & 8 & 3 & 5\\ 13 & 9 & 11 & 15\\ 7 & 4 & 6 & 2\\ 13 & 12 & 16 & 10\end{pmatrix} \quad \frac{1}{17}\begin{pmatrix}1 & 4 & 5 & 14\\ 13 & 10 & 15 & 8\\ 3 & 7 & 2 & 6\\ 16 & 12 & 11 & 9\end{pmatrix}$$

(a) (b) (c) (d)

Figure 9.2.: **(a)+(b)**: Bayer matrices used in the methods of Bayer [19] for size 2 and 4. **(c)+(d)**: Two threshold matrices created with the method of Purgathofer *et al.* [212] with size 4.

$$\frac{1}{16}\begin{pmatrix}0 & 0 & 0\\ 0 & X & 7\\ 3 & 5 & 1\end{pmatrix} \quad \frac{1}{48}\begin{pmatrix}0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & X & 7 & 5\\ 3 & 5 & 7 & 5 & 3\\ 1 & 3 & 5 & 3 & 1\end{pmatrix} \quad \frac{1}{42}\begin{pmatrix}0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & X & 8 & 4\\ 2 & 4 & 8 & 4 & 2\\ 1 & 2 & 4 & 2 & 1\end{pmatrix}$$

(a) (b) (c)

Figure 9.3.: Error diffusion filter proposed by Floyd and Steinberg [80] (a), by Jarvis, Judice, and Ninke [131] (b), and by Stucki [260] (c). The X indicates the pixel currently being processed, and the remainder of the matrix shows which portion of the quantisation error is spread to which pixel. A processing order from top to bottom and left to right was assumed.

was introduced by Bayer [19]. In this approach, the image is tiled into parts having the same size as the given threshold matrix, which is also called *Bayer matrix*, and every image value is increased by the corresponding matrix entry before thresholding. Later, Purgathofer *et al.* proposed to substitute the optimal pattern used by Bayer with less regular matrices [212]. Even nowadays, these two methods are still used in printing due to their efficiency. Example matrices are shown in Figure 9.2.

Another well-known idea in dithering is to spread the local image error obtained by dithering the current pixel to those surrounding pixels which are not dithered, yet. This idea is called *error diffusion*. The most popular dithering approach based on error diffusion is the method by Floyd and Steinberg [80]. When using this method, the image is scanned from left to right and top to bottom. Each pixel encountered is quantised, and the quantisation error is distributed to the surrounding pixels according to the *error diffusion matrix* shown in Figure 9.3(a). At image boundaries, the diffused errors are either "moved out" of the image, or the matrix entries are re-weighted such that the error is only distributed to those places inside the image. When using the first approach, the average grey value of the image is not preserved, while the second method creates more artefacts. It is also possible to swap the processing order in each line, such that every odd line is processed from left to right while every even line is processed from right to left. This is called *serpentine scanning*, and can reduce some of the artefacts that occur for the different methods. More complex processing orders, e.g. a Hilbert curve, have also been proposed [213].

There are many related methods that use the same idea as the original method by Floyd and Steinberg, but introduce different error diffusion matrices. Examples include the stencils

proposed by Jarvis, Judice, and Ninke [131], and by Stucki [260], which are both shown in Figure 9.3. Stevenson and Arce introduced an error filter for images on a hexagonal grid [256]. Further improvements were proposed by Ostromoukhov, who used a different error diffusion matrix for each grey value encountered [188], and by Zhou and Fang, who combined the ideas by Goodall and Ostromoukhov by adapting the thresholding with random noise whose strength depends on the grey value in the image [306]. Furthermore, they used optimised error diffusion matrices afterwards. In such a context, the adaptation of the threshold is called *threshold modulation* [141]. One recent approach even uses the local frequency, orientation, and contrast of the image to choose appropriate thresholds and error diffusion matrices [47]. The five lookup tables used in their method were created manually, and require more than one megabyte in total. In another method, the image is covered with polyomino tilings, which are then used to create threshold matrices for dithering [283].

The methods described so far only consider each pixel once. Thus, they have a linear runtime complexity but cannot undo suboptimal decisions. As a remedy, methods that perform a global optimisation have been proposed. Some first global approaches were presented by Geist *et al.* [93], who proposed two models using Markov simulations and neural networks, respectively. The results of those two approaches are equivalent. As they only consider a local neighbourhood to judge the reconstruction quality, they handle midtones accurately, but have problems with dark or bright image parts. The method patented by Ilbery has similar problems since it only locally optimises space distance measures based on a physical model [120]. However, it performs no global search, but has a processing order as the method by Floyd and Steinberg. Another global approach which minimises a structure-preserving energy functional was introduced by Pang *et al.* in [194]. However, artefacts present in the initialisation often stay visible with this approach. Thus, the authors even propose to initialise this dithering scheme using the method of Ostromoukhov *et al.* explained in [188].

Other modern approaches aim at creating artistic images, e.g. to reproduce traditional drawing or printing techniques [191]. Here, the output of the halftoning algorithm may also consist of other primitives, such as lines, or complex patterns such as a set of Kanjis. In this situation, the primitives are not restricted to only occur at grid locations. Another noteworthy approach is the work by Pnueli and Bruckstein, which uses lines obtained from an Eikonal equation in the resulting image [206].

Stippling is a task similar to sampling and dithering. Here one tries to visualise images by drawing dots of a predefined size whose positions are not restricted to the grid, as illustrated in the right image in Figure 9.1. However, the number of points that may be used is fixed, such that one dot typically has a different size than a pixel in the initial image. One technique to create such images is to apply *Lloyd's algorithm* [158, 171] to create (weighted) centroidal Voronoi diagrams, whose centres are the positions of the dots in the final image. The first such approach that is not interactive was proposed by Second [241]. An approach recently proposed by Balzer *et al.* significantly improves these results [17], and was accelerated by Li *et al.* [154, 155]. Yet other approaches are based on shortest paths in weighted graphs [180], or on the approximation of the image by sums of Gaussians [77].

In the context of sampling, there are many more approaches building upon ideas such as *dart throwing* [69], *tiled blue noise samples* [110], Poisson disks [148, 149, 90], or methods based on aperiodic tilings such as Penrose tiles [190] or Wang tiles [244, 51]. Since introducing

all these methods is beyond the scope of this thesis, we refer to the survey by Lagae and Dutré [150].

The method presented here [233] is motivated by principles from electrostatics, and can be used for stippling, screening, halftoning, or sampling. Using the extension presented in Section 9.3.2, it can also create dithered images and hatching results.

Many recent algorithms aim to achieve specific application-dependent features such as structure-enhancement or prevention of aliasing artefacts, and include these characteristics as integral and mandatory part of their model. However, our approach is much more general: In its basic form, it tries to represent the underlying image or PDF as good as possible, while intuitive and flexible extensions are used to achieve specific characteristics. Thus, the presented algorithm directly adapts to the requested number of dots in the output image, is edge enhancing to a freely adjustable degree, works well on colour and grey-valued images, and can adjust between energetically optimal and visually pleasing results. As we will show in Section 9.4, these extensions are completely optional, and constant parameters already yield very precise results that clearly surpass those of other state-of-the-art algorithms. We work presented in this chapter was first published in [233] and [234]. For our faster optimisation scheme not described in this thesis, we refer to [273] and [104].

The remainder of this chapter is organised as follows: First, we introduce our basic halftoning algorithm and the particle model it is based upon. Section 9.2 gives more details on how to perform the three major steps of our implementation. A number of extensions to the basic algorithm, e.g. for varying dot sizes, colour images, or hatching, are explained in Section 9.3 and thoroughly evaluated in Section 9.4. Section 9.5 concludes this chapter with a summary.

9.1. Electrostatic Model

"When I'm working on a problem, I never think about beauty.
I think only how to solve the problem.
But when I have finished, if the solution is not beautiful, I know it is wrong."

Richard Buckminster Fuller (1895 – 1983)

The basic idea behind our halftoning approach is that, in regions of constant density in the input image, black points should be equally distributed after halftoning. To achieve this, we model black points as (negatively) charged particles in a global particle system. Due to the repulsive forces between the particles, an arrangement fulfilling this condition is obtained when the particle evolution is finished.

The idea described above was first mentioned in [106]. However, the approach from this paper was not extended to arbitrary images, and requires an unintuitive force repelling particles from boundary locations. Another related approach was introduced by Meyer *et al.* [174], which proposed a curvature dependent sampling of implicit surfaces using a particle system. Since particles should only have a local influence according to their approach, it does not use electrostatic potentials, but an alternative energy function utilising an additional parameter

that must be optimised during particle evolution. Such problems do not occur in our approach, which is based on two simple ingredients: A repulsive force between particles, and an attractive force from the underlying image. The first force takes care of a good distribution of the particles, while the second attracts particle to dark image areas. In this section, we introduce these two forces, and explain how both are combined to obtain halftoned images.

For the moment, we assume that the number of particles linearly depends on the grey values in the input image. That is, for a discrete $n_x \times n_y$ image $u : \Gamma \mapsto [0, 1]$, where 0 corresponds to black and 1 to a white and where

$$\Gamma := \left\{ (i, j)^\top \;\middle|\; i \in \{1, \dots, n_x\}, j \in \{1, \dots, n_y\} \right\}, \tag{9.1}$$

we compute the number of black particles in the output image as

$$|\mathcal{P}| := \text{round} \left(\sum_{\boldsymbol{x} \in \Gamma} (1 - u(\boldsymbol{x})) \right), \tag{9.2}$$

where $\text{round}(x) : \mathbb{R} \mapsto \mathbb{N}$ returns the integer closest to x. For dithering, this is basically the only reasonable choice. For stippling, it is possible to choose the number of particles arbitrarily since the size of the particles is not predefined. We will show in Section 9.3.1 how to extent our approach such that it uses another number of particles.

For the remainder of this section, we will identify each particle $m \in \mathcal{P}$ by its charge q_m and its 2-D position $\boldsymbol{p}_m$. Let m and n be two such particles. As shown in Equation (2.29), the repulsive force $\boldsymbol{F}_{r,n,m}$ acting on particle n due to the electric field induced by particle m is given by

$$\boldsymbol{F}_{r,n,m} = -\frac{k \cdot q_n \cdot q_m}{\|\boldsymbol{p}_m - \boldsymbol{p}_n\|} \boldsymbol{e}_{n,m}, \tag{9.3}$$

where $\boldsymbol{e}_{n,m}$ is the unit vector from p_n to p_m. To obtain the total force $\boldsymbol{F}_{r,n}$ acting on particle n due to all other particles, we sum up all forces on this particle:

$$\boldsymbol{F}_{r,n} = \sum_{\substack{m \in \mathcal{P} \\ m \neq n}} \boldsymbol{F}_{r,n,m} = -\sum_{\substack{m \in \mathcal{P} \\ m \neq n}} \frac{k \cdot q_n \cdot q_m}{\|\boldsymbol{p}_m - \boldsymbol{p}_n\|} \boldsymbol{e}_{n,m}. \tag{9.4}$$

This concludes the description of the first force, which is responsible for a good distribution of particles over the image domain. For constant images, this is already sufficient to obtain a reasonable halftoning result, provided that particles may not leave the image domain. However, too many particles will arrange on the boundary when only using this force.

Instead of using a force repelling particles from the boundary as in [106], e.g. by modelling a charged boundary, we now introduce an attractive force that even makes it possible to halftone arbitrary images. This force prevents that particles accumulate at the boundary, and automatically ties particle locations to the image domain. This is done by modelling each pixel $\boldsymbol{x} \in \Gamma$ with grey value $u(\boldsymbol{x})$ as a particle with a positive charge $(1 - u(\boldsymbol{x})) \cdot q$, where q is the charge of the darkest possible pixel. As these positive charges attract the negatively charged particles, this results in the attractive force

$$\boldsymbol{F}_{a,n} = \sum_{\substack{\boldsymbol{x} \in \Gamma \\ \boldsymbol{x} \neq \boldsymbol{p}_n}} \frac{k \cdot q_n \cdot (1 - u(\boldsymbol{x})) \cdot q}{\|\boldsymbol{x} - \boldsymbol{p}_n\|} \boldsymbol{e}_{n,x}, \tag{9.5}$$

where $\boldsymbol{e}_{n,\boldsymbol{x}}$ is the unit vector from $\boldsymbol{p}_n$ to the grid location $\boldsymbol{x} \in \Gamma$.

To ensure that the particles are distributed over the complete non-white image domain, the two forces from Equations (9.4) and (9.5) must be balanced correctly. That is, there must be an equilibrium between attractive and repulsive forces. Except for the inaccuracies introduced by rounding in Equation (9.2), this holds automatically when all involved charges are equal:

$$\forall n \in \mathcal{P}: \quad q_n := q. \tag{9.6}$$

The global force $\boldsymbol{F}_n$ acting on each particle $n \in \mathcal{P}$ is obtained by combining Equations (9.4), (9.5), and (9.6):

$$\boldsymbol{F}_n = k \cdot q^2 \cdot \left(\sum_{\substack{\boldsymbol{x} \in \Gamma \\ \boldsymbol{x} \neq \boldsymbol{p}_n}} \frac{1 - u(\boldsymbol{x})}{\|\boldsymbol{x} - \boldsymbol{p}_n\|} \boldsymbol{e}_{n,x} - \sum_{\substack{m \in \mathcal{P} \\ m \neq n}} \frac{1}{\|\boldsymbol{p}_m - \boldsymbol{p}_n\|} \boldsymbol{e}_{n,m} \right). \tag{9.7}$$

Using these forces, we perform a particle evolution until a situation is reached in which an equilibrium of forces is obtained. As in image inpainting, such a situation is called a *steady-state*. Since we are interested in the steady-state, and not in the evolution in between, we do not consider the acceleration induced by the forces resulting in accelerated particles, but perform a numerically more stable artificial time evolution also resulting in the desired steady-state. In this evolution, each particle is moved according to the weighted force acting on it. This results in the following update equation:

$$\boldsymbol{p}_n^{k+1} = \boldsymbol{p}_n^k + \tau \left(\sum_{\substack{\boldsymbol{x} \in \Gamma \\ \boldsymbol{x} \neq \boldsymbol{p}_n}} \frac{1 - u(\boldsymbol{x})}{\|\boldsymbol{x} - \boldsymbol{p}_n\|} \boldsymbol{e}_{n,x} - \sum_{\substack{m \in \mathcal{P} \\ m \neq n}} \frac{1}{\|\boldsymbol{p}_m - \boldsymbol{p}_n\|} \boldsymbol{e}_{n,m} \right), \tag{9.8}$$

where $\boldsymbol{p}_n^k$ is the position of the particle n at time k, and where τ denotes an artificial time step parameter which also includes the constants q and k. That is, the imbalance of forces is reduced in each time step by repositioning the particles in accordance with the forces induced by the image and the particles. In our experiments, we always use $\tau := 0.1$.

Alternatively, this update equation can also be derived by performing a gradient descent of the potential energy in the particle system. For a particle system without attractive forces, this energy is well known [79]. When including the attractive forces, we get the following potential energy:

$$W = k \cdot q^2 \cdot \sum_{n \in \mathcal{P}} \left(\sum_{\substack{\boldsymbol{x} \in \Gamma \\ \boldsymbol{x} \neq \boldsymbol{p}_n}} (1 - u(\boldsymbol{x})) \cdot \log(\|\boldsymbol{x} - \boldsymbol{p}_n\|) - \sum_{\substack{m \in \mathcal{P} \\ m \neq n}} \cdot \log(\|\boldsymbol{p}_m - \boldsymbol{p}_n\|) \right). \tag{9.9}$$

Using Equation (9.8), each part of the image exactly attracts the number of particles it should contain according to its average grey value. That is, once the correct amount of particles is present in a certain part of the image, this part is electrically neutral and does neither attract nor repel other particles. Especially, this holds for the complete image domain. Thus, if a

particle leaves the image domain, the attractive image forces will be stronger than the repulsive forces, moving the particle back into the image. Therefore, it is not necessary to restrict particle locations to the image domain. To speed up the convergence and for technical reasons, we still prevent particles from leaving the image domain, though.

As attractive forces are uniformly distributed in regions of homogeneous grey values, the optimal particle positions are still those maximising the particle distances. Thus, this approach still follows our basic idea explained above.

9.2. Algorithm

"You can't trust code that you did not totally create yourself."

Ken Thompson (*1943)

Our implementation of the algorithm explained in the last section performs three major steps: The initialisation of the particle locations, a precomputation of the attractive image forces, and the particle evolution. The first two steps are initialisations only performed once, while the last step must be performed several times. In the remainder of this section, we detail on these steps.

As the main parts of our algorithm are perfectly suited for parallelisation, we have implemented them on *Graphics Processing Units* (*GPUs*) using *CUDA* (*Compute Unified Device Architecture*), which is a programming language for NVidia GPUs. More precisely, we use an adapted version of the fast parallelisation technique proposed by Nyland *et al.* [185]. We will leave out most of the technical details of this method, and refer the reader to [185] for a detailed explanation.

9.2.1. Particle Initialisation

The first step done by our algorithm is the initialisation of the particle locations. For this step, any initialisation with the correct number of particles can be used, as long as all particle locations are distinct. However, as we will see in Section 9.4.5, the closer the initialisation is to the optimal solution, the faster the particles will converge. Thus, using the result of another halftoning algorithm as initialisation can be beneficial. Here, we perform a more simple initialisation in order to allow a fair comparison against other methods, and to prevent a biasing of our algorithm: We use a random initialisation in which the probability that a certain position is chosen is proportional to $1 - u(\boldsymbol{x})$. More precisely, the particles are placed one after the other. For each particle, a random grid position and a random number between zero and one is chosen. If the number is smaller than the image brightness at that position, or if there is already a particle at that position, a new position and a new random number are chosen. This is done for each particle until convergence. If there are more particles than possible grid positions, the particles may also be placed at positions between the grid.

As the particle positions are chosen one after the other, this step is done on the CPU, while most of the remainder of the algorithm is implemented on a GPU. As the time required by the particle initialisation step is negligible, this does not have a noticeable negative impact on the running time of the complete algorithm.

9.2.2. Force Field Computation

As second step, we precompute the attractive forces $\boldsymbol{F}_{a,n}$ given in Equation (9.5). This is possible since the term $\boldsymbol{F}_{a,n}$ only depends on the current position, and is thus independent from other particles.

First of all, we compute the attractive image forces in each grid point. For particle locations not coinciding with grid locations, bilinear interpolation will be used in the particle evolution. This way, a smooth representation of the attractive force field is obtained. Note that using a smoothed version of the attractive field automatically prevents attractive forces which are too strong and could bind particles to undesired grid locations.

To compute the attractive forces at grid locations, we move a single test charge over each of the $n_x \cdot n_y = M$ grid position. Then, the force acting on this test charge due to the other $M-1$ point charges residing on the other image grid points is computed. This resembles an M-body particle system in which each grid point interacts with each other grid point. Thus, a modified version of the fast parallel M-body particle system on CUDA is well suited to compute these forces.

Once the attractive forces for the grid positions have been computed, these forces are stored in texture memory on the graphics card. Consequently, the bilinear interpolation is automatically performed in hardware by the GPU using an advanced caching strategy.

9.2.3. Particle Evolution

The movement of each particle depends on the attractive and the repulsive forces. The first force can be obtained by a simple texture lookup (see last section), while the second force again corresponds to an N-body particle system, where N is the number of particles. As for the force field computation, we use an adapted variant of the fast parallelisation technique proposed by Nyland *et al.* [185] to compute the forces acting on each particle.

After each iteration of the particle evolution, we check for convergence by comparing each particle displacement with a given threshold. For all our experiments, this threshold was 10^{-10}. Note that, since all computations on the GPU are performed with single-precision floating point numbers, this threshold might be reached much faster than when using double precision floating point numbers on a CPU. The information if a particle has moved more than this threshold is reduced on a per-block basis, and sent to the CPU. The particle evolution is stopped if there is no particle whose displacement exceeds the threshold, or until a certain number of iterations is reached.

In particular when adapting our algorithm such that it performs dithering (see Section 9.3.2), it is possible that the particle evolution gets stuck in a local minimum. Thus, we also use a procedure we call “shaking”: Every ten iterations, each particle is randomly displaced by a small amount. The direction and strength of this displacement is uniformly distributed,

whereby the maximal possible movement slowly decreases: For the i-th of NOI iterations, the maximal displacement is given by

$$\max\left(0, \frac{\log_2(NOI) - 6}{10}\right) \cdot \exp\left(\frac{-i}{1000}\right). \tag{9.10}$$

That is, most shaking is performed at the beginning of the particle evolutions, and when many iterations are requested. Seen from a physical standpoint, the particles perform a Brownian motion, whereas the temperature of the particles – and therefore the amount of motion – decreases during the particle evolution. This strategy is known as *simulated annealing*, and is well known in literature.

Note that there was no random number generator available in CUDA when we implemented our algorithm. Thus, the particle locations were transferred to the CPU, perturbed, and moved back to the GPU.

9.3. Extensions

"Ideas are like rabbits. You get a couple and learn how to handle them, and pretty soon you have a dozen."

John Steinbeck (1902 – 1968)

In the last section, we introduced our halftoning algorithm based on electrostatic forces. Due to its flexibility, this basic algorithm can be easily extended. We present several such extensions in this section.

9.3.1. Different Dot Sizes

When computing the number of black particles in Equation (9.2), we assumed that one dot in the output has the same size as one pixel in the initial image. Depending on the application, this is not always the case, e.g. when sampling a density distribution or when placing objects. One possibility to overcome this problem is to create a large point cloud which is sparsified recursively, as proposed in [143]. However, it is also application dependent whether such post-processing steps are possible. Furthermore, computing large point clouds can be computationally expensive.

With the proposed algorithm, using a different dot size is straightforward: If we denote the ratio between the size of one pixel in the image and one dot in the output by r, all that has to be done is to divide the brightness of the pixels in the input image by a factor of r. Note that this is the same as multiplying the charge of each particle by r while simultaneously scaling the number of particles down with the same factor.

Further note that, using this extension, it is even possible to account for the slight inaccuracy introduced by rounding in Equation (9.2): After the number of particles $|P|$ has been

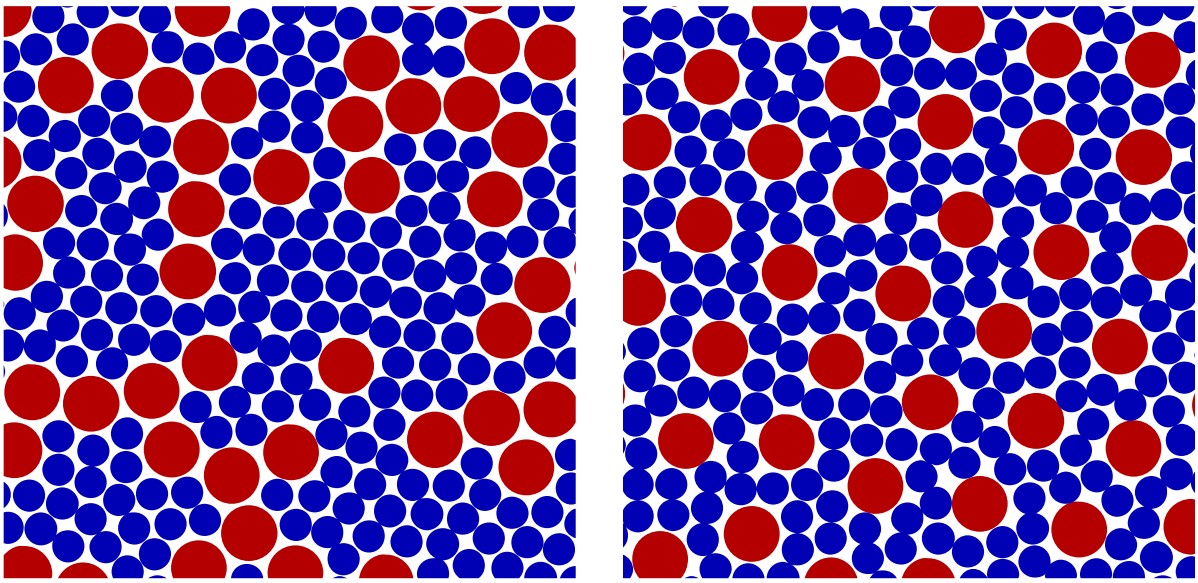


Figure 9.4.: Halftoning results of a constant grey value without (left, $C = 1$) and with (right, $C = 2$) the additional force responsible for an equal distribution of large particles. The colours were only used for illustration purposes.

computed, we determine the ratio r with the formula

$$r = \frac{\sum_{\boldsymbol{x}\in\Gamma}(1-u(\boldsymbol{x}))}{|P|} = \frac{\sum_{\boldsymbol{x}\in\Gamma}(1-u(\boldsymbol{x}))}{\text{round}\left(\sum_{\boldsymbol{x}\in\Gamma}(1-u(\boldsymbol{x}))\right)}, \tag{9.11}$$

and divide the brightness values in the image by this factor. Then, the complete image domain is exactly electrically neutral, which was only approximately true before.

Furthermore, it is even possible to use dots of varying sizes within one image: In fact, the only change necessary is to assign charges proportional to the dot sizes to the corresponding particles. Then, all dots automatically approximate the image. However, the larger dots may accumulate in some image regions, as illustrated in the left image in Figure 9.4. Typically, such clusters are undesired. In printing applications, for example, some of the smaller dots may not be printed due to technical reasons. However, the remaining large dots alone do not approximate the underlying image well.

Let us first consider the case of only two different dot sizes. In this situation, we introduce a supplementary force that encourages a good distribution of the large particles such that a result as shown in the right image of Figure 9.4 is obtained. Consequently, even if the small dots are removed, the remaining particles still approximate the image well. This supplementary force only affects the large particles, and is, except for a multiplicative constant $C - 1$, exactly the same force that occurs in our particle system when only large particles are present. Thereby, the parameter $C \geq 1$ deals as a tradeoff between the two conflicting requirements imposed on the system: The larger C, the better the approximation of the original image when considering only large particles, but the worse the approximation of all particles.

	S	**L**
S	$A_{\mathrm{S}}A_{\mathrm{S}}$	$A_{\mathrm{S}}A_{\mathrm{L}}$
L	$A_{\mathrm{S}}A_{\mathrm{L}}$	$CA_{\mathrm{L}}A_{\mathrm{L}}$
u	A_{S}	$(w_S + Cw_L)A_{\mathrm{L}}$

Figure 9.5.: This *interaction matrix* shows the total repulsion strength between small particles (S) with area A_{S} and large particles (L) with area A_{L}. The attraction strength for each particle size from the underlying image is shown in the last line. The constant C is a user-defined variable explained in detail in the text. The variables w_S and w_L correspond to the fraction of charges located in small and large particles, respectively.

	S	**M**	**L**
S	$A_{\mathrm{S}}A_{\mathrm{S}}$	$A_{\mathrm{S}}A_{\mathrm{M}}$	$A_{\mathrm{S}}A_{\mathrm{L}}$
M	$A_{\mathrm{S}}A_{\mathrm{M}}$	$C_1A_{\mathrm{M}}A_{\mathrm{M}}$	$C_1A_{\mathrm{M}}A_{\mathrm{L}}$
L	$A_{\mathrm{S}}A_{\mathrm{L}}$	$C_1A_{\mathrm{M}}A_{\mathrm{L}}$	$C_2\ A_{\mathrm{L}}A_{\mathrm{L}}$
u	A_{S}	$(w_{\mathrm{S}} + C_1(w_{\mathrm{M}} + w_{\mathrm{L}}))A_{\mathrm{M}}$	$(w_{\mathrm{S}} + C_1w_{\mathrm{M}} + C_2w_{\mathrm{L}})A_{\mathrm{L}}$

Figure 9.6.: Interaction matrix i for three particle sizes. Each entry in the table shows the repulsion strength between pairs of consisting of small particles (S) with area A_{S}, medium sized particles (M) with area A_{M}, or large particles (L) with area A_{L}. In the last line, the interaction strength of each particle class with the attractive image field is given. The variables w_S, w_M, and w_L correspond to the fraction of charges located in small, medium, and large particles, respectively.

However, there are now more forces repelling large particles than attracting them, i.e. the complete image is not electrically neutral any more. As this leads to problem, we need additional attractive forces to maintain the electric neutrality of the overall image. To solve this problem, we first denote the fraction of the charges located in large particles by w_L, and the fraction of the charges located in small particles by w_S. If we multiply the force attracting large particles by the factor $w_S + Cw_L$, the electric neutrality is restored, as large particles are attracted C times as strong from the fraction of the image corresponding to the large particles, and as strong as before from the remainder.

All these concepts are summarised in the *interaction matrix* i depicted in Figure 9.5. Each entry $i(A_j, A_k)$ of the matrix i indicates how strong the interaction between particles from the classes A_j and A_k are.

This additional force slightly reduces the approximation quality of all particles. However, this problem arising from the fact that a tradeoff between two conflicting requirements is searched for. Thus, it is a unavoidable problem, which must occur in any algorithm trying to solve this problem. As shown in the experiments section, this impact is typically negligible in real-world images.

These ideas can be easily extended to more than two dot sizes. Then, the property that all particles approximate the image as good as possible should also hold for each subset containing only the particles up to a certain size. The resulting interaction matrix for three particle sizes is shown in Figure 9.6. For the experiments shown in this thesis, we choose the necessary parameters C_i in a multiplicative manner, i.e. $C_i = C_1^i = 2^i$. This is only one of several possible choices, though. In the general setting, the attractive image force $f(s)$ acting on the particle s is given by

$$f(s) = \frac{\sum_{t \in \mathcal{P}} i(A_s, A_t) A_t}{\sum_{t \in \mathcal{P}} A_t}. \tag{9.12}$$

We can even deal with situations in which there are as many classes as particles. That is, even if each particle has a unique size, halftoning is possible when choosing the interaction matrix appropriately. An example is given in the experiments section.

9.3.2. Dithering

The extension introduced in this section is complicated, but very important: We will extend the algorithm such that the computed particle locations at the end of the evolution are at one of a predefined set of possible locations. Here, this will always be the rectangular grid given by the initial image. That is, the extension explained in this section allows to use the proposed algorithm for dithering.

The most simple idea how to achieve this goal is to first compute the unrestricted result, followed by assigning each particle to the closest grid location. As can be seen in Figure 9.7, this yields very bad results. This is due to several reasons. First of all, multiple particles can be assigned to the same grid location. In this example, this happened over 400 times and noticeably reduced the image brightness. Furthermore, hexagonal structures occur in the continuous setting due to their optimality. Those structures cannot be adequately represented on the regular rectangular grid imposed by the given image domain. This leads to clearly visible artefacts.

To deal with these problems, we include the rectangular grid as additional constraint into the model by adding three additional ingredients. First, we add another force term that pulls particles to grid locations. This force is given by

$$\boldsymbol{F}_{\boldsymbol{g},n} = \beta \cdot \frac{1}{1 + \frac{\|\boldsymbol{x}_{n\boldsymbol{g}}\|^8}{\lambda^8}} \cdot \frac{\boldsymbol{x}_{n\boldsymbol{g}}}{\|\boldsymbol{x}_{n\boldsymbol{g}}\|}, \tag{9.13}$$

where $\boldsymbol{x}_{n\boldsymbol{g}}$ is the vector from the particle n to the nearest grid point $\boldsymbol{g} \in \Gamma$, the regularisation weight $\beta := 3.5$ steers the influence of this force term on the system, and the contrast weight $\lambda := \frac{1}{\sqrt{10}}$ decides by how much a particle is pulled back depending on the distance to this grid point. This is a nonlinear weighting structurally similar to the Perona-Malik function used in diffusion filtering [201]. However, a higher exponent is used here. Technically, this additional force comes down to adding $\tau \boldsymbol{F}_{\boldsymbol{g},n}$ to Equation (9.8).

The force just introduced pulls particles to grid points, which results in a solution much closer to the rectangular grid. Consequently, the solution looks much better when the particles are mapped to grid locations as last step of the algorithm,

Figure 9.7.: Illustration of dithering results. **Top left**: Initial image. **Top right**: Halftoned image with continuous dot placement. **Bottom left**: Result when mapping the space-continuous dots to discrete pixel locations. **Bottom right**: Dithering result with proposed dithering algorithm.

In some sense, this force negates the effect of the regularisation introduced by computing the attracting image forces only at grid locations, and using bilinear interpolation in between. However, simply removing this regularisation is a poor alternative to this additional term as the attractive forces are unbounded, as already mentioned in Section 9.1.

Still, there are local minima in which particles stay far from the grid positions. Thus, we project all particles back onto the closest horizontal or vertical line between the grid points after each time step. This is the second ingredient used. As a result, particles align in rectangular instead of hexagonal structures.

These two ingredients are sufficient to obtain good-looking dithering results in many cases, see Figure 9.7. However, when there are white regions in the image, a third ingredient must be added. In general, white regions should not contain any particles in the end. This can happen if particles undergo large displacements per time step, e.g. if particles get too close together due to shaking or a too large time step size τ. In the continuous setting, this is no problem as the particle can steadily move back to a good position. In the discrete setting, however, the extensions explained above prevent such a slow movement, as the involved forces are too strong. The easiest way to prevent particles from being stuck is by using the extensions only at non-white image areas, and by limiting the maximal displacement of each particle to one pixel per time step. This allows particles to gradually get back while simultaneously preventing them from moving too far from desirable positions.

9.3.3. Edge Enhancement

It is in the nature of the problem that sampled or dithered images are not as sharp as the original image. Thus, sharp edges are less pronounced after halftoning. However, there are applications in which the visual quality is more important than an optimal approximation of the initial image.

Around 1976, it was thus proposed to emphasise edges by using a preprocessing step before applying the dithering algorithm [131, 132]. This idea was originally proposed for dithering algorithms based on error diffusion, but can also be used for some other halftoning approaches, including the proposed halftoning and dithering approaches. In fact, both the classic unsharp masking [96] and the discrete variant proposed for the Floyd-Steinberg algorithm can be used without modifications. Note that the image brightness can leave the range $[0, 1]$ due to this edge enhancement step. As a consequence, those image parts with a brightness greater than one actually repel particles. This is already included in Equation (9.8), and does not need a special treatment. Further note that those halftoning algorithms which are based on weighted Voronoi cells cannot handle such cases without modifications, while other approaches such as the one in [93] have a built in edge enhancement that cannot be turned off.

9.3.4. Artefact Prevention ('Jittering')

As we will see in Section 9.4, the results of the proposed sampling algorithm are very good with respect to different error measures. In uniform image regions, however, our algorithm often arranges dots in hexagonal structures. Such patterns are easily noted by humans, and are thus undesired in some applications. In [158], it was proposed to prevent these artefacts by

stopping an iterative scheme before convergence. However, it is unclear *when* their iterative schemes should be stopped.

Although this idea also works for the proposed algorithm, we prefer an approach in which it is not necessary to guess when the algorithm should be stopped. Instead, the so-called "jittered" variant of our method converges to a minimum not containing hexagonal structures. Moreover, the user is able to steer between optimal quality and prevention of artefacts such that an optimal tradeoff between visual and analytic quality can be found by adapting a tradeoff parameter.

When using this extension, a high-frequent turbulence field T is constructed during initialisation and added to the attraction field created from the image. This is done by creating a random vector

$$\begin{pmatrix} w_x \cdot d \cdot \sin(\varphi) \\ w_y \cdot d \cdot \cos(\varphi) \end{pmatrix} := \begin{pmatrix} w_x \cdot d(\boldsymbol{g}) \cdot \sin(\varphi(\boldsymbol{g})) \\ w_y \cdot d(\boldsymbol{g}) \cdot \cos(\varphi(\boldsymbol{g})) \end{pmatrix} \tag{9.14}$$

at each grid point $\boldsymbol{g} \in \Gamma$, where $\varphi(\boldsymbol{g})$ is a random direction in the interval $[0°, 360°)$ and $d(\boldsymbol{g})$ is a random number between zero and a parameter j. This parameter, which is 0.1 for all experiments presented here, is the tradeoff parameter described above. The weights $w_i(f, \boldsymbol{g}) := 1 - |\partial_i f(\boldsymbol{g})|$ are due to the fact that jittering is mainly necessary in homogeneous image regions, i.e. where $\partial_i f \approx 0$. From these random vectors, the complete turbulence field T is obtained by performing bilinear interpolation between these vectors. Since T is constant during the particle evolution, and since we use the same bilinear interpolation as for the attractive field computed from the image, these two forces can be combined into a single texture on the GPU. Thus, additional time is only required during initialisation and not during particle evolution. Since we have merely adapted the attractive field, the process stays convergent. Technically, this extension adds the term $\tau T(\boldsymbol{p}_n^k)$ to Equation (9.8).

9.3.5. Colour Images

Halftoning images using linear colour spaces such as CMY and RGB is straightforward, as each channel can be halftoned independently. Printers typically use more complex colour models, though. A common example is the (comparably simple) CMYK colour model which is found in all sorts of printers ranging from inexpensive inkjet printers such as the Canon Pixma i560 via business-scale colour laser printers like the Kyocera FS-C5200DN up to mass-production printing machines. Some recent printers even use more complex, 'asymmetric' colour models: The Canon Pixma iP8500, for example, utilises the CMY-RG-K colour space, while the CMY-RB-K colour space is employed in the Stylus Photo R1800 by Epson. In large scale production printing machines, the 'full' CMY-RGB-K colour model is typically used.

In this section, we introduce an extension to our halftoning approach that allows to use any of the colour models mentioned above. To ease the explanations, we assume that all particles have the same size.

To deal with the interactions between dependent channels, we again use the concept of interaction matrices introduced in Section 9.3.1. However, each matrix entry denotes the interaction between particles with two colours instead of with two sizes.

Since there are no interactions between different channels for linear colour models such as RGB or CMY, the interaction matrix is the identity matrix in this case.

	C	M	Y	R	G	B	K
C	1	α	α	α	α	α	α
M	α	1	α	α	α	α	α
Y	α	α	1	α	α	α	α
R	α	α	α	1	α	α	α
G	α	α	α	α	1	α	α
B	α	α	α	α	α	1	α
K	α	α	α	α	α	α	1

	C	M	Y	R	G	K
C	1	0	α	α	α	α
M	0	1	α	α	α	α
Y	α	α	1	α	α	α
R	α	α	α	1	α	α
G	α	α	α	α	1	α
K	α	α	α	α	α	1

Figure 9.8.: Interaction matrices for the CMY-RGB-K (left) and the CMY-RG-K (right) colour space.

Let us consider the restrictions in the CMYK colour model. First of all, we still want the dots of each colour to approximate the corresponding channels. This is again modelled by an identity matrix. In addition to the requirement that two particles with the same colour should not overlap, we request that no particle at all should overlap with a black particle. This is due to the fact that a black particle can be seen as a union of the remaining three colour. We encode this restriction in a second matrix, which we call *non-overlap matrix*, and obtain the final interaction matrix by a linear combination of these two matrices:

$$(1-\alpha)\underbrace{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}}_{\text{identity}}+\alpha\underbrace{\begin{pmatrix}1&0&0&1\\0&1&0&1\\0&0&1&1\\1&1&1&1\end{pmatrix}}_{\text{non-overlap map}}=\underbrace{\begin{pmatrix}1&0&0&\alpha\\0&1&0&\alpha\\0&0&1&\alpha\\\alpha&\alpha&\alpha&1\end{pmatrix}}_{\text{interaction matrix}} \tag{9.15}$$

Unless noted otherwise, we always use $\alpha = \frac{1}{2}$, i.e. we give the same weight to both constraints.

Extending this approach to other colour models is straightforward: For any two colours whose particles are allowed to overlap, a zero is placed in the non-overlap matrix, and a one otherwise. For the CMY-RGB-K colour model, there should be no overlaps between any particles, resulting in a 7×7 non-overlap matrix containing only 1s. For the CMY-RG-K colour model, a natural choice is to prevent any overlaps except between cyan and magenta particles, as this is the only combination with which a new colour is obtained. The interaction matrices for these two examples are shown in Figure 9.8.

There is one important step not described so far, namely how to create the image attracting the particles for each colour, and therefore also how to choose to amount of particles for each channel. Several suggestions for such a decomposition can be found in the literature, including a chromatic decomposition resulting in nonempty C, M, and Y channels only, or an (a)chromatic decomposition with under colour removal (UCR), see [140]. Any of these method works fine with our approach. In the experiments shown in Section 9.4, we use a

	$\mathbf{S}_a$	$\mathbf{L}_a$	$\mathbf{S}_b$	$\mathbf{L}_b$
$\mathbf{S}_a$	$A_\mathrm{S}A_\mathrm{S}$	$A_\mathrm{S}A_\mathrm{L}$	$\alpha A_\mathrm{S}A_\mathrm{S}$	$\alpha\ A_\mathrm{S}A_\mathrm{L}$
$\mathbf{L}_a$	$A_\mathrm{S}A_\mathrm{L}$	$CA_\mathrm{L}A_\mathrm{L}$	$\alpha A_\mathrm{S}A_\mathrm{L}$	$\alpha CA_\mathrm{L}A_\mathrm{L}$
$\mathbf{S}_b$	$\alpha A_\mathrm{S}A_\mathrm{S}$	$\alpha\ A_\mathrm{S}A_\mathrm{L}$	$A_\mathrm{S}A_\mathrm{S}$	$A_\mathrm{S}A_\mathrm{L}$
$\mathbf{L}_b$	$\alpha A_\mathrm{S}A_\mathrm{L}$	$\alpha CA_\mathrm{L}A_\mathrm{L}$	$A_\mathrm{S}A_\mathrm{L}$	$CA_\mathrm{L}A_\mathrm{L}$
$\mathbf{u}_a$	A_S	$(w_{S,a}+Cw_{L,a})A_\mathrm{L}$	αA_S	$\alpha(w_{S,b}+Cw_{L,b})A_\mathrm{L}$
$\mathbf{u}_b$	αA_S	$\alpha(w_{S,a}+Cw_{L,a})A_\mathrm{L}$	A_S	$(w_{S,b}+Cw_{L,b})A_\mathrm{L}$

Figure 9.9.: Final interaction matrix when using two classes a and b, where each class contains particles with two sizes A_L and A_S.

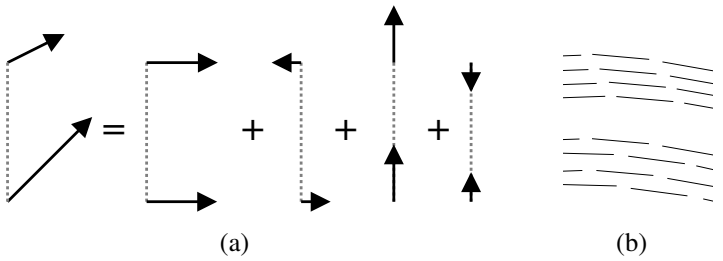


Figure 9.10.: **(a)**: Illustration how we split the forces acting on a line consisting of two particles. The first and third force results in a translation, the second force in a rotation around the middle of the line, and the last force to a length change. **(b)**: Comparison of an inappropriate line placement with visible artefacts (top) and a better placement without such artefacts (down).

simple achromatic composition of the CMY colour space. That is, we set the K channel to the minimum of the C, M, and Y channel, and subtracts this values from the C, M, and Y channel. Afterwards, one of these three channels is zero in every image point. Then, we either subtract the R channel from the M and Y channels, the G channel from the C and Y channels, or the B channel from the C and M channels in each point, depending on which channel is zero.

As we demonstrate with the interaction matrix shown in Figure 9.9, combining colour halftoning with variable dot sizes is straightforward. Thus, our approach is also well suited for general multi-class sampling algorithms, which are needed for object placement or procedural texture creation. Examples will be shown in the experiments section.

9.3.6. Hatching

Finally, our last extension illustrates how to perform hatching, i.e. we explain how to approximate images with lines.

One again, the basic idea behind this extension is very simple: Instead of using charged particles, we use charged rods. However, it is much more complicated to compute the resulting forces in this situation. Thus, we approximate each rod by a few connected particles. In each

iteration, the forces acting on each particle used in the approximation is computed in exactly the same way as before, and the movement of each line is computed from these forces. This is done by splitting the forces into components responsible for translations, rotations, and stretching, as illustrated for an approximation with two particles in Figure 9.10(a). These components can easily be found using projections and elementary operations.

In fact, directly moving each particle instead of using the decomposition into the forces mentioned above would yield the same particle movement. However, the decomposition allows to impose additional constraints necessary to obtain a visually pleasing result. For example, the length and orientation of each line can be arbitrary at the moment, as the two particles basically move independently.

In order to solve this problem, we first introduce a desired line length and orientation at each image position. For the experiments performed in this thesis, the desired line length is always set to a constant, and the desired orientation is set to a direction orthogonal to the eigenvector corresponding to the largest eigenvalue of the smoothed structure tensor [82]

$$K_\rho * (\boldsymbol{\nabla} u_\sigma \boldsymbol{\nabla} u_\sigma^\top), \qquad \text{where} \qquad u_\sigma := K_\sigma * u \tag{9.16}$$

where $K_\star$ is a Gaussian with standard deviation $\sigma = 1.0$ or $\rho = 3.0$ times the pixel size, respectively. However, any other choice is possible, as our GPU implementation simply reads the desired line length and orientation from a texture, which is generated on the CPU at the beginning of the program.

To encourage the lines to obtain the desired orientation, we introduce a new force that rotates the line. The strength of this new force to proportional to the difference between requested and current angle. However, the rotation obtained from the particle movement is not necessarily identical to the new force. That is, as in multi-size and colour halftoning, there are two constraints between which a compromise must be found. Again, we compute the weighted average between these forces to obtain a situation in which both requirements are fulfilled as good as possible. The appropriate weights can (and should) therefore be chosen by the user depending on what is more important: an accurate approximation of the image, or a perfect match between desired and final line direction. For the experiments shown here, both weights are set to 0.4, i.e. their sum is less than one. This favours moving a line instead of rotating it by slowing down the rotations.

In some situations, it can happen that all directions are equally good. In homogeneous regions, for example, the eigenvector corresponding to the largest eigenvalue can be chosen arbitrarily. Thus, the force responsible for aligning the orientation of the line with the desired line direction is additionally scaled by a number from the interval $[0, 1]$. Here, we chose this position dependent number as the largest eigenvalue of the smoothed structure tensor at the current position, divided by the largest eigenvalue occurring anywhere in the image. Consequently, the lines may arrange arbitrarily in homogeneous image regions, but are still forced to align perpendicular to image edges.

Similarly, we introduce a force that shrinks or expands the line, which is as strong as the difference between the current and the requested line length. As before, we weight and average the two forces resulting in a change of the line length. To prevent crossing lines, the line length should only change very slowly, though. Thus, we set the weights to $\frac{1}{5}$ (force acting on

particle) and $\frac{1}{10}$ (force from desired line length), respectively. Additionally, our initialisation uses very short lines distributed with the same approach as if only particles are used to prevent crossing lines in the initialisation. This allows the lines to distribute appropriately before the final line lengths are reached. By this steps, almost all crossing lines can be prevented.

There is one parameter in hatching not discussed so far, namely the width of each line. Here, we use a constant width computed in such a way that the average grey value of output and input image are equal. Again, any other choice is possible, though.

It is obvious that, the more particles are used to approximate each lines, the better but more expensive the approximation will be. According to our experiments, using three particles placed in the middle and at the end of each line already yields a very good approximation at a reasonable running time. As forces acting on the middle particle only influence the translational part of the forces, this situation is very similar to the two particle case.

However, there is still one possible problem left: It can happen that the spaces between lines arrange in regular patterns, as illustrated in the upper part of Figure 9.10(b). When using three particles per line, this problem can be prevented by assigning a higher charge to the particle in the middle of the line, as it encourages the middles of the lines to approximate the image as well. Consequently, the situation shown in the lower part of Figure 9.10(b) is much more likely than the problematic one described before. According to our experiments, the results are very similar for a large range of charge ratios. In the experiment we present in the next section, the middle particle has a charge six times as large as the charge of the particles at the end of the line.

Since the position of dots and lines are determined by the same principles, it is very easy to use both dots and lines to approximate an image, see Figure 9.30. Furthermore, it also straightforward to use lines with different colours or different widths.

9.4. Experiments

"Listen up, everyone! I have one word that I want to say to all of you.
And that word is... FIRST PLACE!"

Yukari
in *Azumanga Daioh, Episode 1.6*

In this section, we evaluate the performance of our halftoning algorithms. This is done by comparing several quality criteria such as the blue-noise property [280]. For all experiments, we compare our continuous results with and without jittering against those obtained with the method of Balzer *et al.* [17]. Our implementation of this method builds upon the source code available at [16]. Unless noted otherwise, we used 1024 points per site, and a Euclidean metric for all images shown here.

Additionally, we also compare our dithering approach against the methods by Bayer [19] (with a matrix of size 8×8), by Floyd and Steinberg [80], by Geist *et al.* [93] (based on neural

networks), by Purgathofer *et al.* [212] (using a matrix at least as large as the image), by Ostromoukhov [188], by Zhou and Fang [306], and by Pang *et al.* [194] (always using the result of the approach by Ostromoukhov as initialisation). For the method of Ostromoukhov, we used freely-available source code [189]. All other methods were implemented from scratch.

Furthermore, we illustrate the effect when using each of the optional extensions introduced in Section 9.3, show particle evolutions for different initialisations, evaluate the dithering approach in the context of image inpainting, and sketch the running time and memory usage of our algorithm.

Note that, since an additional halftoning step is done when printing images, we propose to look at the images in this section in the PDF version of this thesis. Further note that aliasing artefacts can occur at certain zoom ratios, which are not due to the halftoning methods.

9.4.1. Visual Evaluation

First of all, we evaluate the visual quality of our halftoning approaches by halftoning a simple regular ramp covering all grey values from 0 to 255, and a 2-D Gaussian.

Halftoning results for the grey ramp created with the different algorithms are shown in Figure 9.11. The proposed algorithm has a good tone preservation, but suffers from regularity artefacts. When using method of Balzer *et al.* [17], such artefacts are not present. However, tone is also less accurately preserved. When using our jittered variant, the advantages of both approaches are combined: Although the tone is preserved very well, there are no striking artefacts.

Of the eight discrete methods, the proposed method (using the extension from Section 9.3.2) also yields the best results. It does neither create artefacts in the middle of the ramp, such as the methods by Floyd-Steinberg [80], Ostromoukhov [188], Bayer [19], Pang *et al.* [194], or Purgathofer *et al.* [212], nor does it prevent this through a high graininess, which is done in the method of Zhou and Fang [306]. As the method by Geist *et al.* [93] only considers a local neighbourhood, it fails to accurately dither very bright and dark regions.

Figure 9.12 shows results when halftoning a Gaussian with a standard deviation of 40 pixels. Here, all continuous algorithm yield visually very similar results. This is not true for the discrete algorithms, though. The methods by Geist *et al.* once again fail to reproduce low frequencies, while the method by Pang *et al.* preserves only the structure, but not the tone. The method by Purgathofer *et al.* scatters the black pixels too strongly, while the method of Bayer has too many artefacts. Especially for the method of Floyd-Steinberg, the processing order is clearly visible. The same is true for the method by Ostromoukhov, and for the method by Zhou and Fang. In contrast to those approaches, no direction is preferred by our discrete algorithm since it is rotationally invariant. The same is true for our continuous algorithm.

Even though a good halftoning method should be rotationally invariant, halftoning a rotationally invariant image should *not* result in a symmetric output. This is due to the fact that such a result has visually unpleasing artefacts. As can be clearly seen, our algorithm does not create symmetric output images. This is due to (up to) three reasons: First of all, the initialisation step places particles randomly, resulting in an initial particle configuration which is not symmetric. Secondly, the shaking step moves all particles differently. Finally, the turbulence

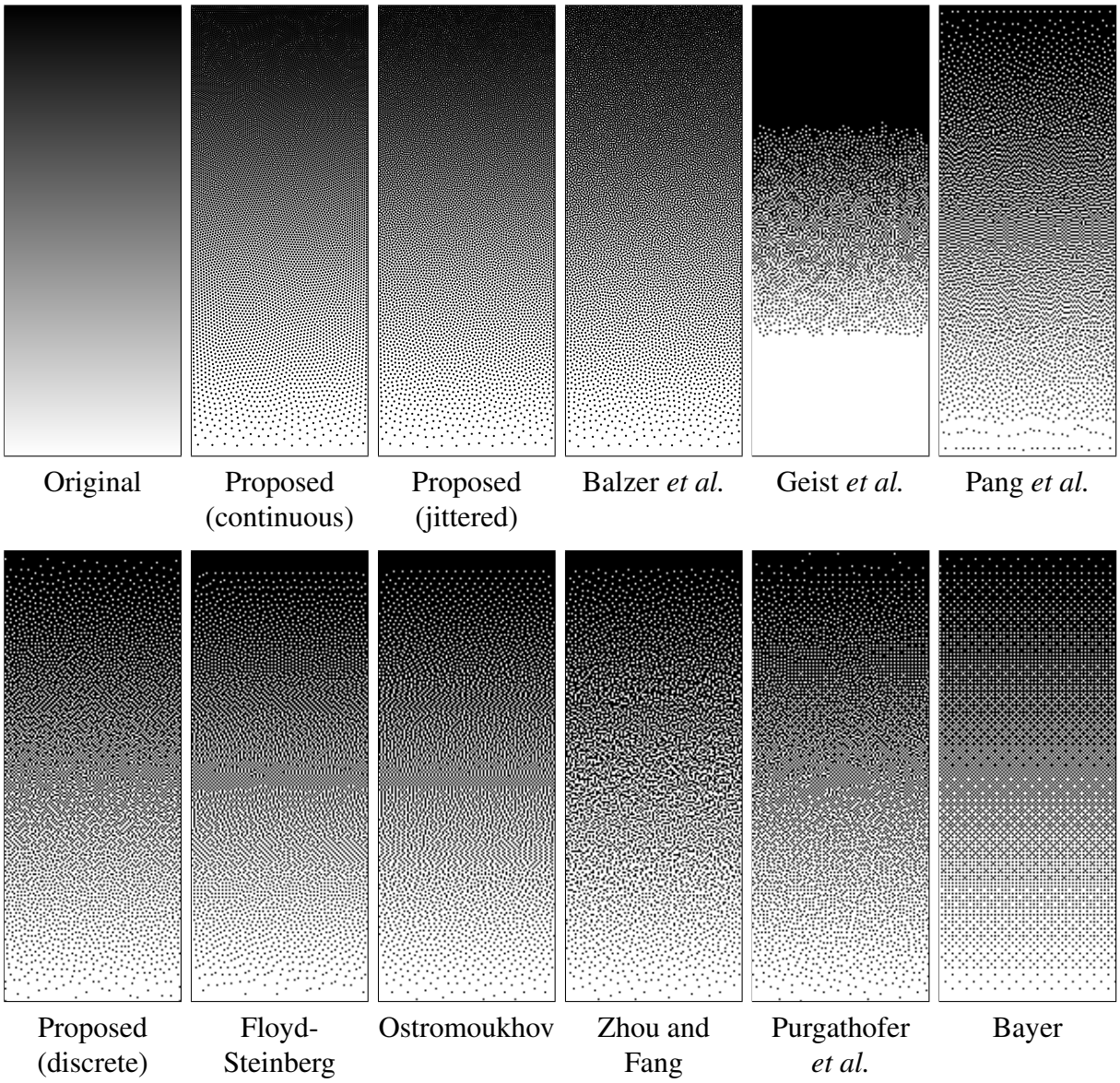


Figure 9.11.: Visual quality of halftoning results for an input image with size 100×256 showing a grey ramp.

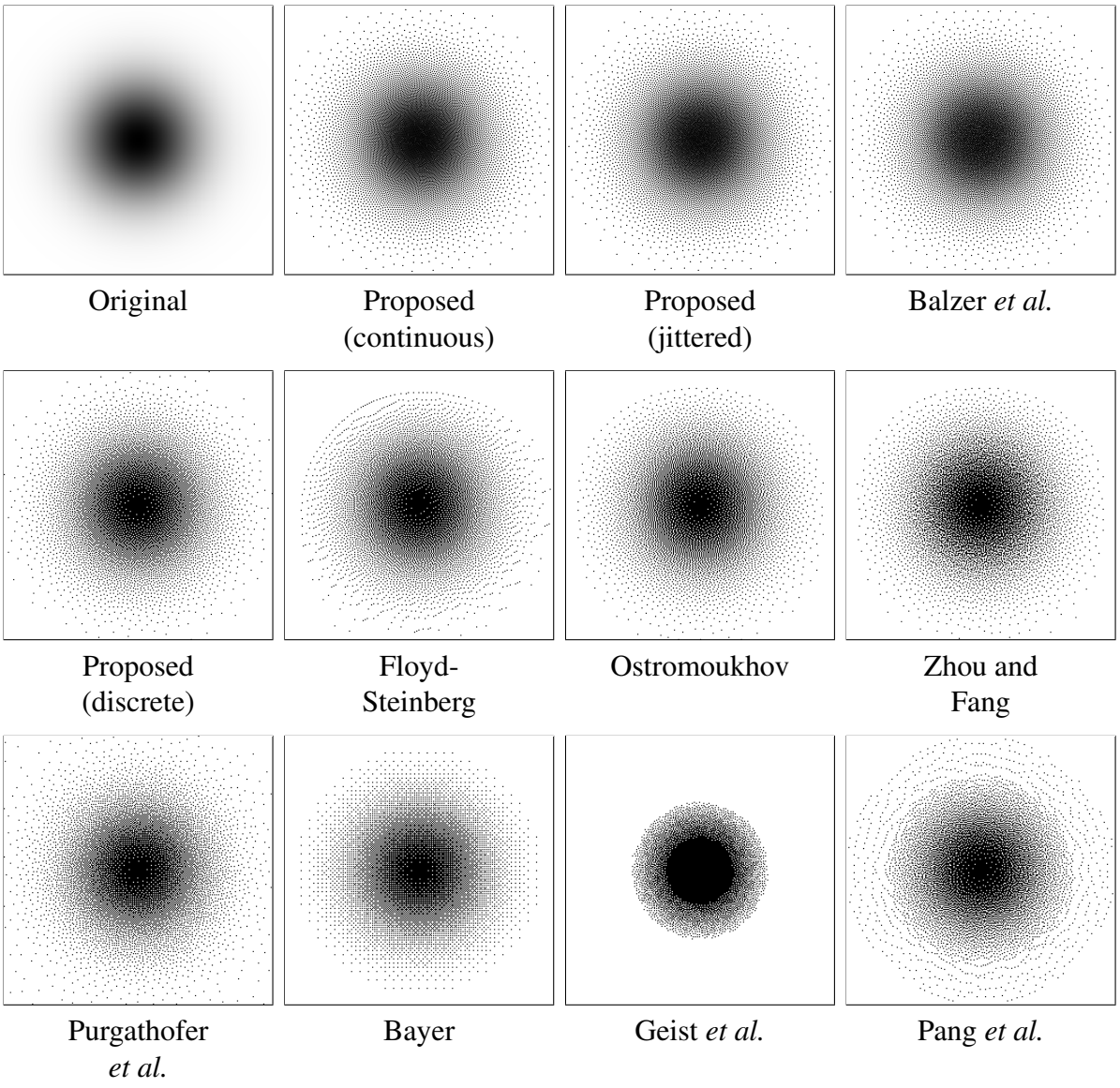


Figure 9.12.: Visual quality of halftoning results for an input image with size 100×256 showing a Gaussian.

field added in the jittered variant of our algorithm is spatially varying and thus prevents symmetric results. Note that each of these steps or extensions does not prefer any directions but still prevents output images containing symmetries. Still, if symmetric images are desired, it is very easy to adapt our algorithm such that it creates such images.

9.4.2. Evaluation in Gaussian Scale Space

In the last section, we showed a visual evaluation of our halftoning algorithm. However, this is only a subjective evaluation. Now, we perform an objective evaluation motivated by the idea that a human looking at a halftoned image from a certain distance should not be able to see a difference to the original image. To this end, we simulate a certain distance by smoothing the images by convolution with a Gaussian, thereby modelling the human visual system.

For this experiment, we use the image "trui" which contains smooth transitions, sharp boundaries, and textured regions at different scales. Figure 9.13 shows halftoned images in the first row, and blurred versions (obtained by convolution with a Gaussian with standard deviation $\sigma = 1.0$) in the second row. Difference images between blurred original and blurred halftoned images are shown in the third row. These error images have been scaled by a factor of five, and shifted by 128. Furthermore, only magnified parts of all images are shown such that details are spotted easier.

If halftoning is perfect, there is no difference between both smoothed images. Thus, for a good halftoning result, the error should be small everywhere, and no image structures should be visible in the error images. However, one should be aware that a Gaussian with a large standard deviation is necessary to prevent artefacts, especially in sparsely sampled regions. This can be seen left of the nose in the error maps of the continuous halftoning methods: As a Gaussian with $\sigma = 1.0$ cannot fill the region properly, artefacts are visible. A similar problem also occurs for all dithering methods, since only few fixed point positions are possible.

Apart from those theoretically justified artefacts, the error of our continuous halftoning algorithm is very small, resulting in an error image that has almost a constant grey colour. The method of Balzer *et al.* [17] creates far more fluctuating error images, indicating that the error is higher.

As seen in the error images, the methods by Geist *et al.* [93] and Pang *et al.* [194] fail to find good approximations due to the fact that only a limited neighbourhood is used, or due to its structure-enhancing property, respectively. For the remaining dithering approaches, no or little structural information can be seen. However, compared to the approaches by Floyd-Steinberg [80] or Ostromoukhov [188], no regular patterns can be seen in our discrete approach. The approach by Purgathofer *et al.* [212] has a lot of inaccurate points manifesting as very dark or very bright dots in the error image. The same is true for the approach by Zhou and Fang [306], while our approach yields much better results in this respect. Thus, our discrete algorithm is the only dithering algorithm without noticeable artefacts in this experiment.

Since the standard deviation of the Gaussian σ directly corresponds to the simulated distance between viewer and (printed) image, a large range of different σ should be considered to evaluate halftoning algorithms. Thus, we evaluate the quality of the halftoned images using

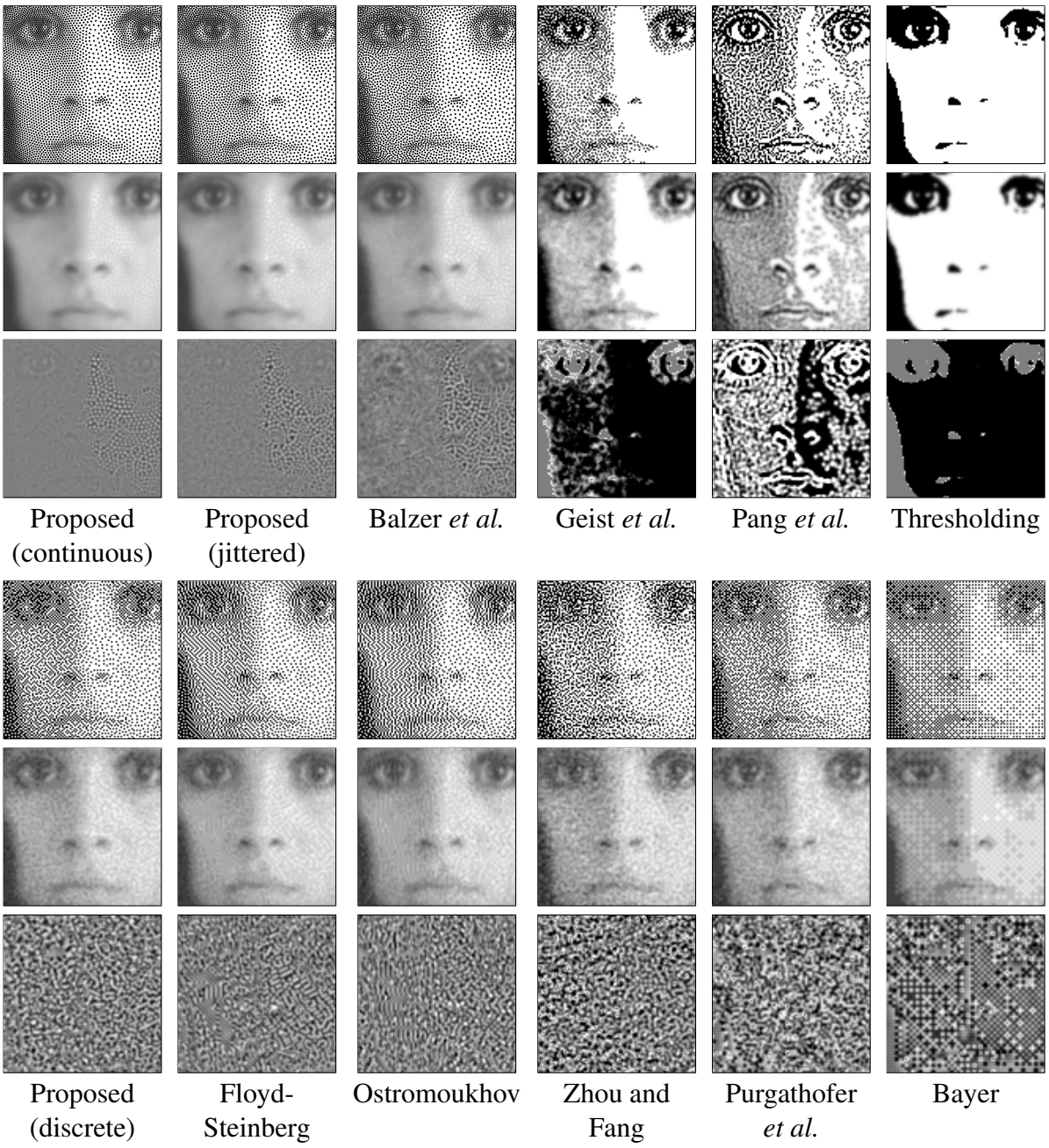


Figure 9.13.: Comparison of blurred results of different halftoning algorithms to the blurred original. As test image, the image "trui" (resolution 256×256) was used. Only a zoom is shown such that the differences are easier to see. The original image is shown in Figure 3.6(a). **From top to bottom**: Dithering/Stippling result, blurred result with $\sigma = 1$, and scaled and shifted difference image. An optimal error image has a grey value of 50% everywhere. Note that, for the continuous results, an optical illusion makes dots appear larger in regions with a small amount of dots for a certain viewing distance. When zooming into the image or out of the image, this illusion disappears.

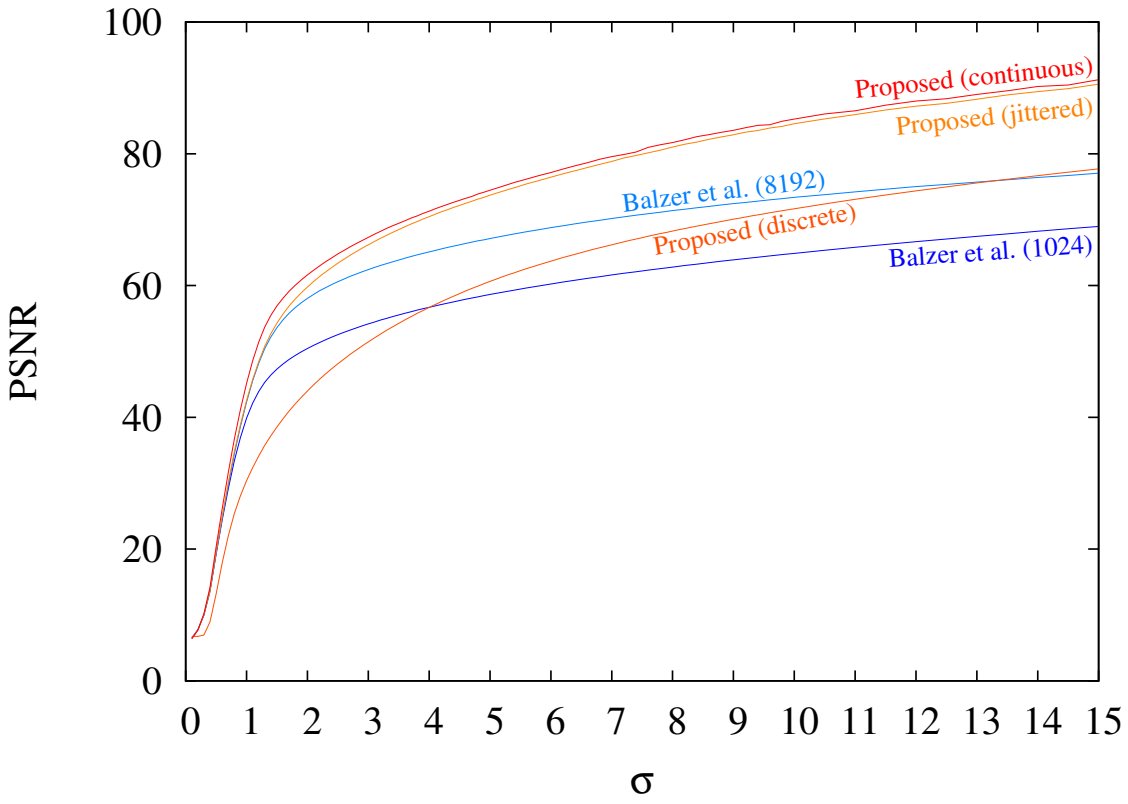


Figure 9.14.: Peak signal to noise ratio (PSNR) for stippling methods under Gaussians of varying standard deviation σ (see also Figure 9.13). For comparison, the plot for our dithering approach is also shown.

different values for σ. That is, for each σ, we compute the *Peak Signal to Noise Ratio* (*PSNR*)

$$\mathrm{PSNR}(f, g) := 10 \log_{10} \left(\frac{255^2}{\mathrm{MSE}(f, g)} \right). \tag{9.17}$$

between blurred original f and blurred halftoned image g, where MSE is the mean squared error between f and g, see Equation (3.2). This results in the graphs shown in Figure 9.14 for the continuous methods, and in Figure 9.15 for the discrete methods.

As can be seen in Figure 9.14, our continuous algorithm yields results that are much closer to the original image than those obtained with the method of Balzer *et al.* with 1024 points per site, especially for large σ. Note that large values of σ correspond to large viewing distances, and thus to fine printing resolution. When using our jittered variant, the results are only slightly worse. The excellent results of our algorithm are due to the global optimisation of the particle locations. Note that even our discrete variant yields more accurate results than the method by Balzer *et al.* for sufficiently large σ, even though the restriction of points positions to grid location makes a good approximation much more difficult. This is even true when increasing the number of points per sites in the algorithm of Balzer *et al.* from 1024 to 8192[1].

[1]The result with 8192 points per site was kindly provided by Thomas Schlömer from the University of Con-

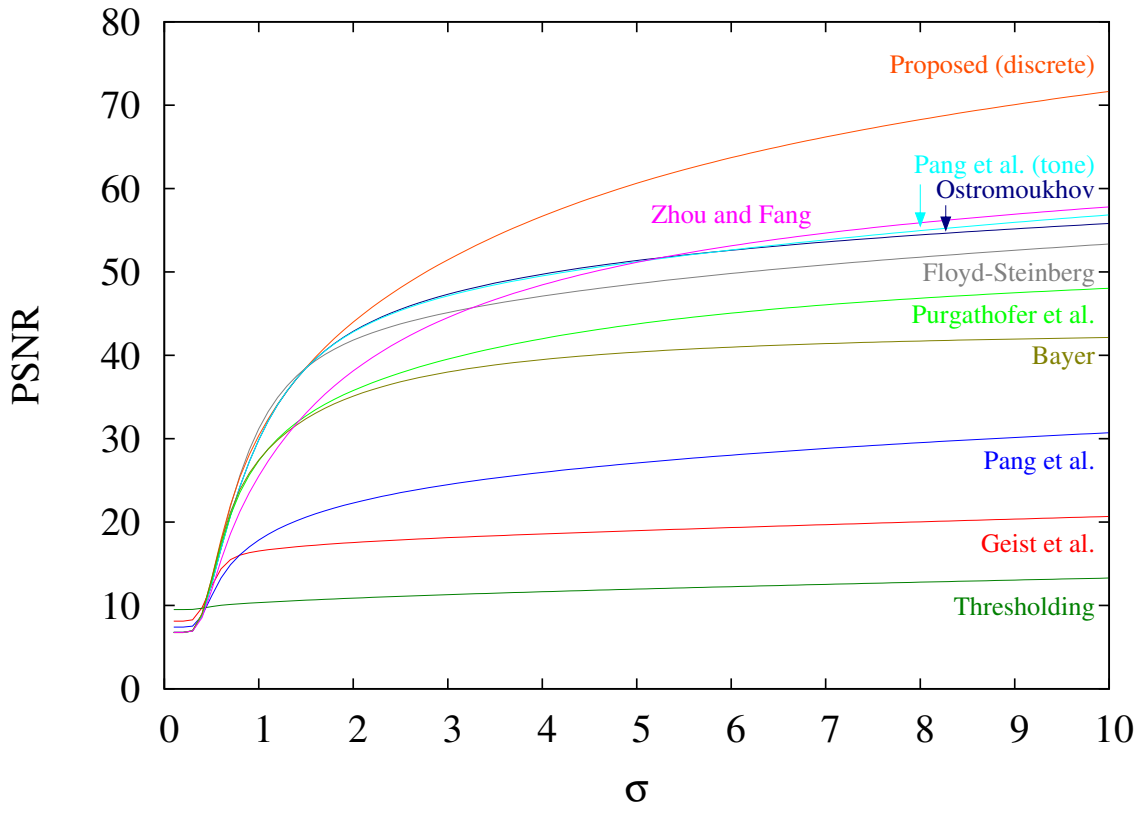


Figure 9.15.: Peak signal to noise ratio (PSNR) for dithering methods under Gaussians of varying standard deviation σ.

Consequently, it is not surprising that our discrete algorithm also clearly outperforms the other dithering algorithms for large σ, see Figure 9.15. The performance for small σ is very similar. It should be noted that a simple thresholding yields the best error for very small σ. Since the visual appearance when using a simple thresholding is very poor, this indicates that larger σ are more interesting.

9.4.3. Evaluation of Blue Noise Behaviour

As an additional quality criterion, we evaluate the spectral properties of our halftoning algorithms and compare it against those of several other algorithms. This is done by halftoning constant images, computing the power spectrum of the difference between original and halftoned image, and radially averaging this power spectrum. This way, the frequencies appearing in the halftoned image can be seen very well. High frequencies cannot be prevented and are thus allowed. Low frequencies, however, should not appear [176, 280]. Images conforming to these properties are said to have a *blue noise behaviour*. The so-called *principal*

stance, as it requires much more memory than can be allocated with the 32-bit operating system we were using.

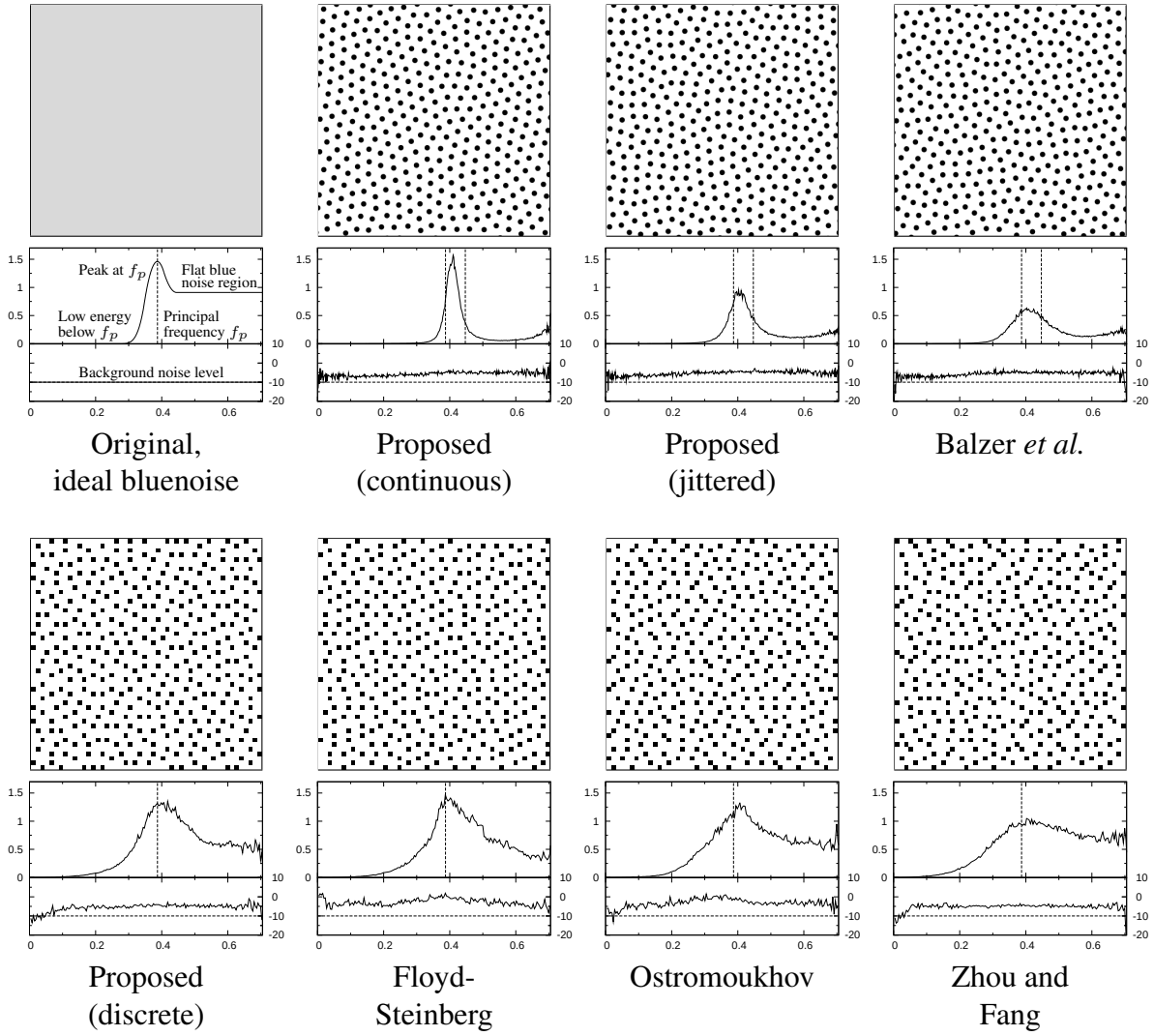


Figure 9.16.: Evaluation of the blue noise behaviour for 85% grey. Every method is represented by a patch of the generated result, by its radially averaged power spectrum (upper graph), and the anisotropy of the power spectrum in decibels (lower graph). The diagrams are obtained from an average of 10 periodograms.

frequency is used to distinguish between low and high frequencies. For rectangular grids, the principal frequency f_g is given by

$$f_g = \sqrt{\frac{1}{2} - \left| g - \frac{1}{2} \right|} \tag{9.18}$$

where $g \in [0, 1]$ is the brightness of the input image. For hexagonal grids, the principal frequency is computed as

$$f_g = \frac{2}{\sqrt{3}} \sqrt{\frac{1}{2} - \left| g - \frac{1}{2} \right|}. \tag{9.19}$$

Since a hexagonal structure is optimal in a continuous setting, the principal frequency in a continuous setting is also given this equation.

According to Ulichney [280], there should be a sharp transition region below f_g and a potential overshoot at f_g. For higher frequencies, a flat blue noise behaviour noticeably lower than the overshoot is allowed. An illustration of a graph with such properties is shown in Figure 9.16, together with our measurements for several algorithms. Our results are obtained by averaging ten randomly selected patches of a larger image, as proposed by Bartlett [18]. Since some methods create artefacts at image boundaries, only patches with a sufficient distance from the image boundaries are considered.

As can be seen in Figure 9.16, our algorithm has a sharper transition region than the method by Balzer *et al.*, especially without using the artefact-preventing extension. Note that the peak for all continuous methods appears between the two principal frequencies for rectangular and hexagonal grids. This is due to the fact that the points still depend on the underlying, rectangular image, while the energetically optimal distribution corresponds to that of a hexagonal grid. Consequently, the peak appears between the corresponding two principal frequencies.

In the discrete case, our approach also works very well. The same is true for the approach by Floyd and Steinberg, and for the method by Ostromoukhov. On the other hand, the method by Zhou and Fang barely has an overshoot at the principal frequency. While this is not a problem itself, it is an indication that the initial image is not approximated very well.

In addition to the frequencies contained in the halftoned image, one must also consider the anisotropy of the averaged power spectra, i.e. the variance on circles of the same frequency. The smaller the anisotropy, the better the halftoning method. Note that the anisotropy x is usually plotted in *decibels*, i.e. using the relation

$$x_{\text{dB}} = 10 \log_{10} x. \tag{9.20}$$

Due to noise in the measurements, there will always be some stochastic background noise. More precisely, this background noise appears at $-10 \log_{10} K$ dB, where K is the number of patches averaged. Thus, the background noise appears at a level of -10dB here.

When looking at the anisotropies shown in Figure 9.16, one can see that there is barely a noticeable difference in the anisotropies. An exception is the method by Floyd and Steinberg, which has a slightly higher anisotropy. The same is true for the method by Ostromoukhov, albeit to a smaller degree. Noticeable peaks, which indicate that certain directions are preferred, are not present for any method.

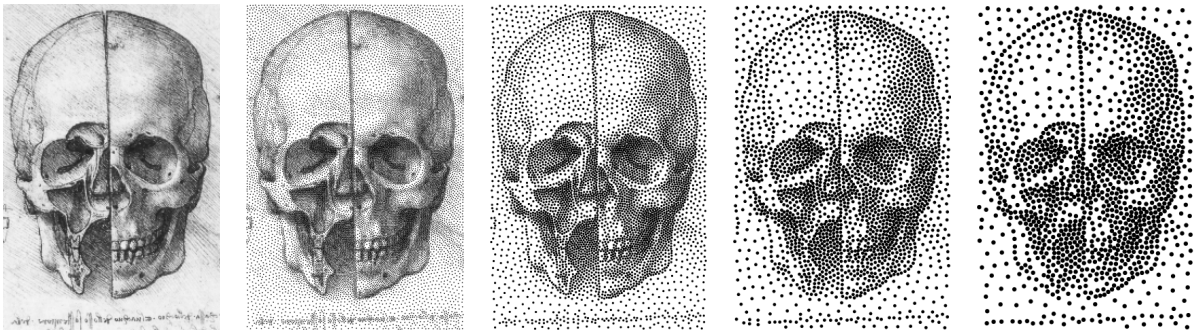


Figure 9.17.: Illustration of halftoning with different dot sizes using the image "The skull bisected and sectioned" by Leonardo da Vinci. The image has a resolution of 200×300. **From left to right:** Original, continuous results with dot sizes of 0.5, 1.0, 1.5, and 2.0.

9.4.4. Illustration of Optional Extensions

So far, we have evaluated the quality of our halftoning algorithm. In this section, we illustrate the effect obtained when using one of the optional extensions introduced in Section 9.3. That is, we show results with dots of different sizes, with and without edge enhancement, with various degrees of artefact prevention, and show a couple of colour images. The extension to dithering is not included here, as it has already been evaluated in detail in the last sections.

Different Dot Sizes

To illustrate the effect of varying dot sizes, we halftoned the image "The skull bisected and sectioned" by Leonardo da Vinci with dots of different sizes. Original and halftoned versions of this image are shown in Figure 9.17. As can be seen, our algorithm creates very good results, independent on the size of the dots. Even in the very coarse representation on the right side of Figure 9.17, lines of the original image are clearly noticeable.

The left column of Figure 9.18 shows a halftoning of a Gaussian with particles with sizes 1 and 3 pixels, and a magnification thereof. The number of particles of each size were chosen such that $w_L = \frac{1}{3}$ and $w_S = \frac{2}{3}$ holds. As desired, the large particles alone additionally approximate the underlying image as well. In the right column of Figure 9.18, a halftoning of the image "skull" with particles of area 1, 2, and 4 pixels with $w_L = \frac{1}{9}$, $w_M = \frac{2}{9}$, and $w_S = \frac{6}{9}$ is shown. As before, the image is not only approximated very well by the set of all particles, but also by the two subsets corresponding to the largest particles, or all particles except the smallest ones.

To create Figure 9.19, we used particles having unique sizes linearly going from 0.5 to 2 pixel. Shown are the complete result, as well as versions in which 25%, 50%, and 75% of the particles with smallest size have been omitted. For the interaction matrix, we set $i(A_j, A_k) = \min(A_j^2, A_k^2)A_jA_k$ for two particles with sizes A_j and A_k. When omitting some of the particles, the image becomes brighter, which is inevitable. Nevertheless, the remaining particles still approximate the underlying image well.

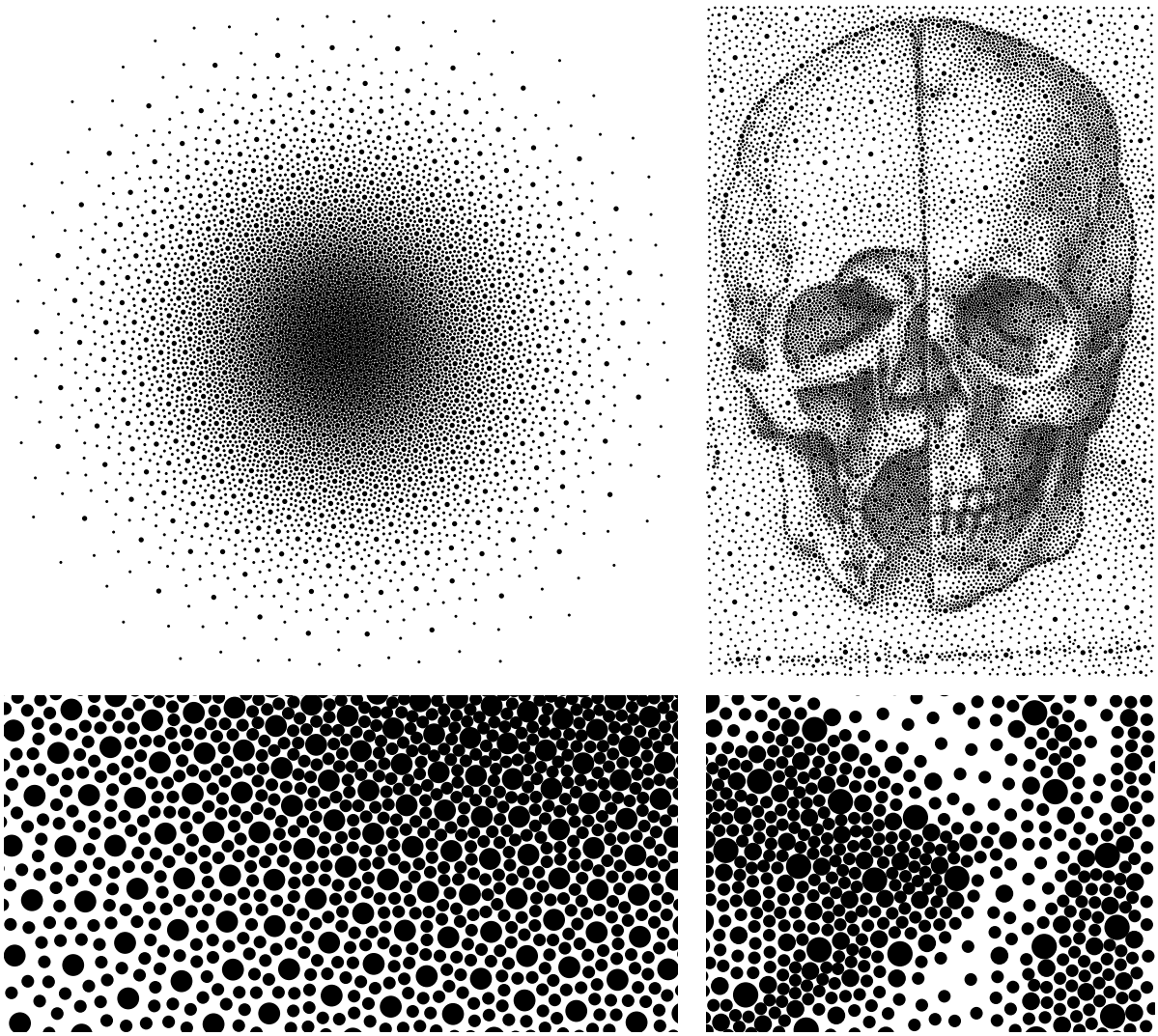


Figure 9.18.: Halftoning results created by our algorithm with two (left) and three (right) different particle sizes. **Left**: Halftoning of "Gaussian" (256×256). **Right**: Halftoning of "skull" (200×300). **Top**: Complete images. **Bottom**: Zoom into the Gaussian and into the tip of the right eye, respectively.

Figure 9.19.: Results when halftoning with dots of continuously varying sizes. Again, the image "trui" (256×256) was used. All images show the same result. However, in all but the first image, some of the dots with smallest radius were omitted. **Top left**: Result with all dots. **Top right**: Result when omitting 25% of the dots. **Bottom left**: Result when omitting 50% of the dots. **Bottom right**: Result when omitting 75% of the dots. Apart from the unavoidable fact that the image gets noticeably brighter, it still approximates the underlying image well.

Figure 9.20.: Illustration of halftoning with edge enhancements using the image "Study for the Head of Leda" by Leonardo da Vinci. The image has a resolution of 256×256. **From left to right**: Input image, continuous result without edge enhancement, and results with little (radius 5, factor 0.5) and more unsharp masking (radius 10, factor 1.0).

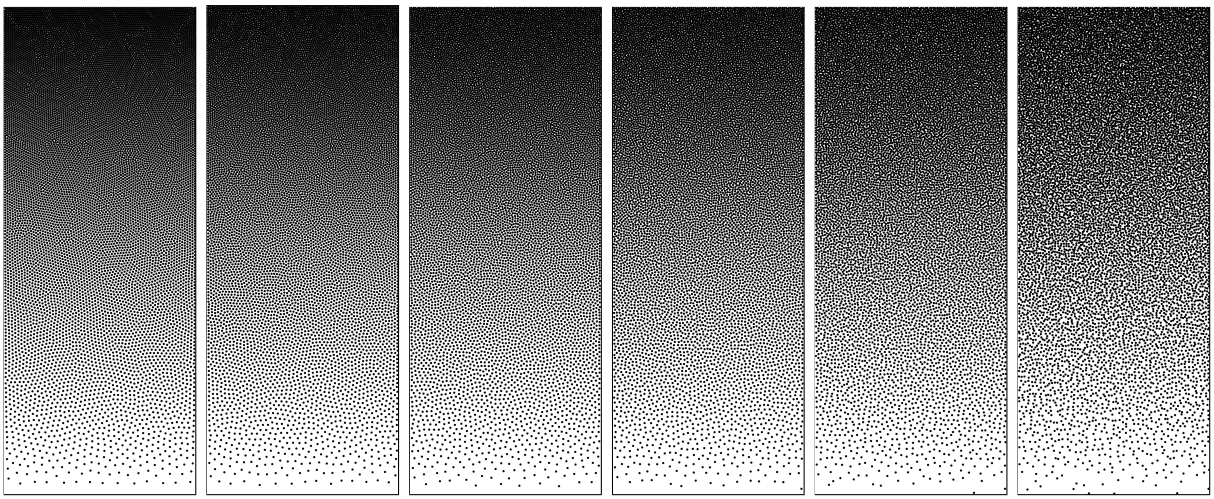


Figure 9.21.: Effect of the parameter j used in our artefact preventing extension. The larger the parameter j, the less likely artefacts are. However, results will also get less accurate. Shown are, from left to right, results for $j = \frac{1}{2}$, $j = 1$, $j = 2$, $j = 4$, $j = 8$, and $j = 16$.

Edge Enhancement

Images illustrating the effect of edge enhancement are shown in Figure 9.20. As edge-enhancing preprocessing step, we have used the unsharp masking implemented in "The GIMP" (GNU Image Manipulation Program), version 2.2, see [199]. Especially for small structures such as braids and curls, the edge-enhancing can be clearly observed, as they are much more emphasised.

Artefact Prevention

Next, we illustrate the influence of the parameter j used in our optional artefact prevention, see Section 9.3.4. A large value of j corresponds to a strong turbulence field, thus resulting in energetically less optimal results with less artefacts. Halftoning results for the image "ramp"

Figure 9.22.: Visual comparison of halftoning results for a photograph of a statue of Charles XIV, in Norrköping. The input image has a resolution of 229×229. Original image licenced as CC-BY-SA 2.0 by User:Thuresson, published on Wikimedia Commons.

with a large range of different j's are presented in Figure 9.21. The original image, as well as the results without artefact prevention, are shown in Figure 9.11.

As can be seen, this parameter allows a smooth transition between energetically optimal and visually pleasing results: For small j, regular structures are visible. As these structures are not aligned identically, there are *cracks* visible between those structures. For large j, there are no such problems, but the halftoned image is not very accurate. For intermediate values of j, a good tradeoff between artefact-freedom and accuracy can be obtained. As mentioned before, we have used $j = 1$ for all other experiments.

Colour Images

Next, we evaluate the visual performance of our algorithms for colour images by comparing the results of our algorithms against those from the eight methods already utilised in Sec-

tions 9.4.1 and 9.4.2. For these first experiments, the CMY colour model was used. Thus, each channel was halftoned independently.

As a first test image, we halftone the photograph shown in Figure 9.22. As can be seen in this figure, the continuous methods yield comparable results of high quality. However, most discrete approaches have some artefacts. The methods by Floyd-Steinberg and Ostromoukhov create regular patterns and worm-like artefacts. The latter can be seen especially well in the upper left corner of the image and at the chest of the statue. With the approach by Bayer, artefacts are visible over the complete image domain. The method by Geist *et al.* fails to reproduce the correct colours due to the fact that two of the channels are nearly zero. This is visible very well in the background. The method by Pang *et al.* once again pronounces edges too strong. The approach by Purgathofer *et al.*, as well as the one by Zhou and Fang, yield good-looking results, but we have already shown that this comes at the cost of a less accurate approximation of the initial image (see Section 9.4.2). In contrast, the result from our approach, which approximate the initial image very well, also has very few artefacts.

The second colour image we use for evaluation is the photograph of two hands holding a Rubik's cube, see Figure 9.23. This image is challenging due to several reasons: First of all, it contains completely white regions. This is challenging since points incorrectly placed in such a region are easily spotted by humans. Similarly, slight variations in the skin colour are also easily recognised. Further note that, in contrast to the first colour image, this image has saturated colours. Thus, the performance for weak and bright colours is evaluated when considering both experiments.

Nearly everything described for the halftoning algorithms with the last image is also true for this image. Thus, we describe only new findings in order not to repeat the problems already explained.

Once again, the overall quality of the continuous halftoning methods is very similar at first glance. However, the approach by Balzer *et al.* places some single dots into white regions, e.g. between the middle and ring finger of the left hand (see first magnification in Figure 9.23). Since this methods represents Voronoi cells by their centres, such stray dots occur whenever colour information is distributed along the boundary of a Voronoi cell. Furthermore, we can see that dark colours are not represented very accurately due to the fact that a this method has a built-in "jittering" (see third magnification). The same is true (up to some degree) for our jittered variant. However, when using the approach of Balzer *et al.*, the jittering cannot be turned off or reduced, which is trivial with our approach.

Our discrete variant also preserves colours nicely, which is not the case for the methods by Geist *et al.*, Pang *et al.*, or Bayer. Furthermore, our method does not put single dots into the white areas, which happens both for the method of Ostromoukhov, as well as for the method by Zhou and Fang. Also note the worm artefacts visible for the methods by Ostromoukhov and Floyd-Steinberg, especially in the second magnification.

So far, we have only considered the linear CMY colour space possible with all halftoning algorithms. In our next experiment, we illustrate that our approach can also use more complex colour models. Figure 9.24 shows results with the common CMYK colour space, and with the asymmetric CMY-RB-K colour space. For comparison, the result with the linear colour space CMY is given. Note that none of these colour spaces contains green as a single colour. Nevertheless, the dark green in the regions of the T-shirt is represented well in all results.

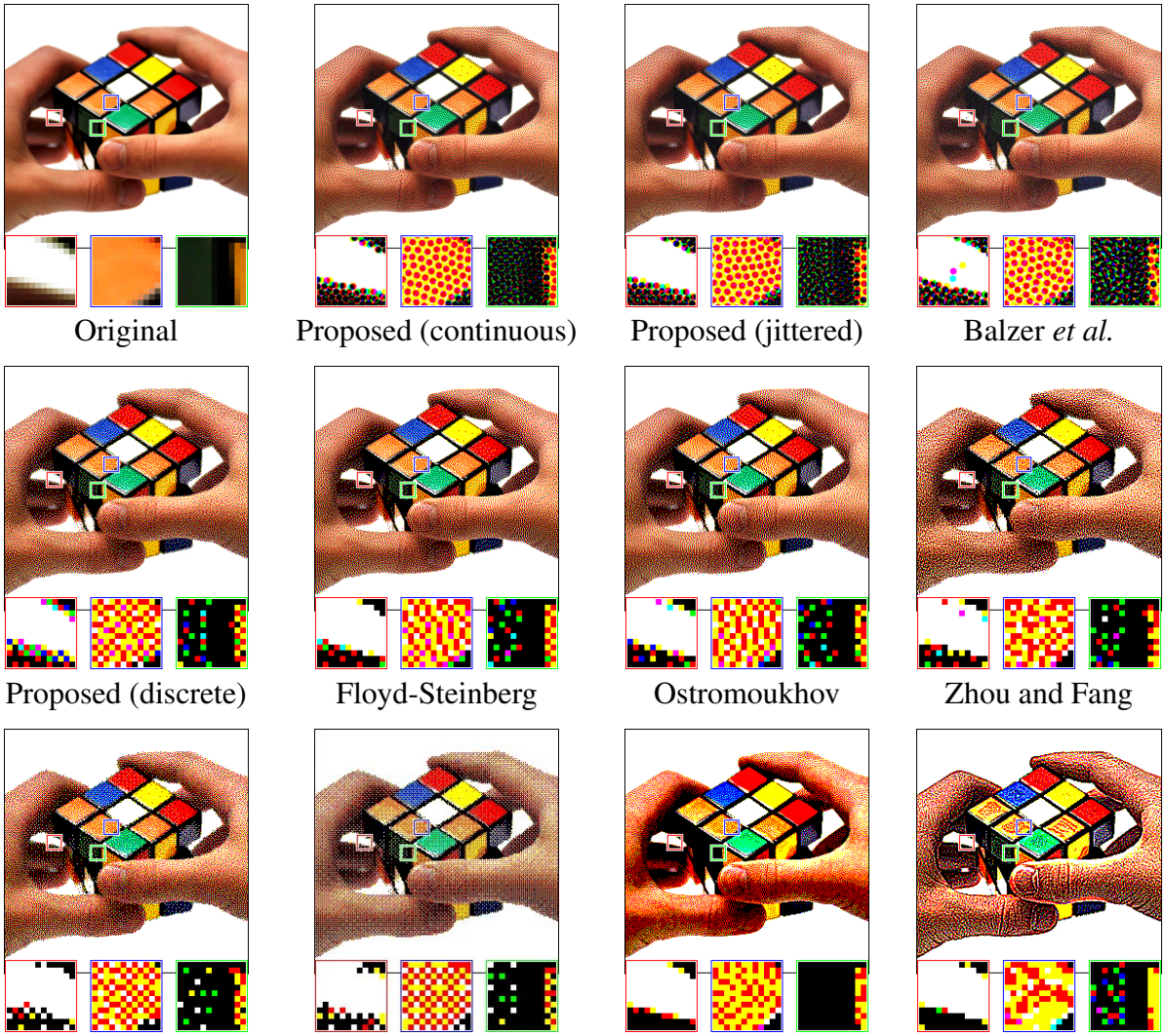


Figure 9.23.: Visual comparison of halftoning results for the “Rubik’s cube” test image. The original has a resolution of 256×256.

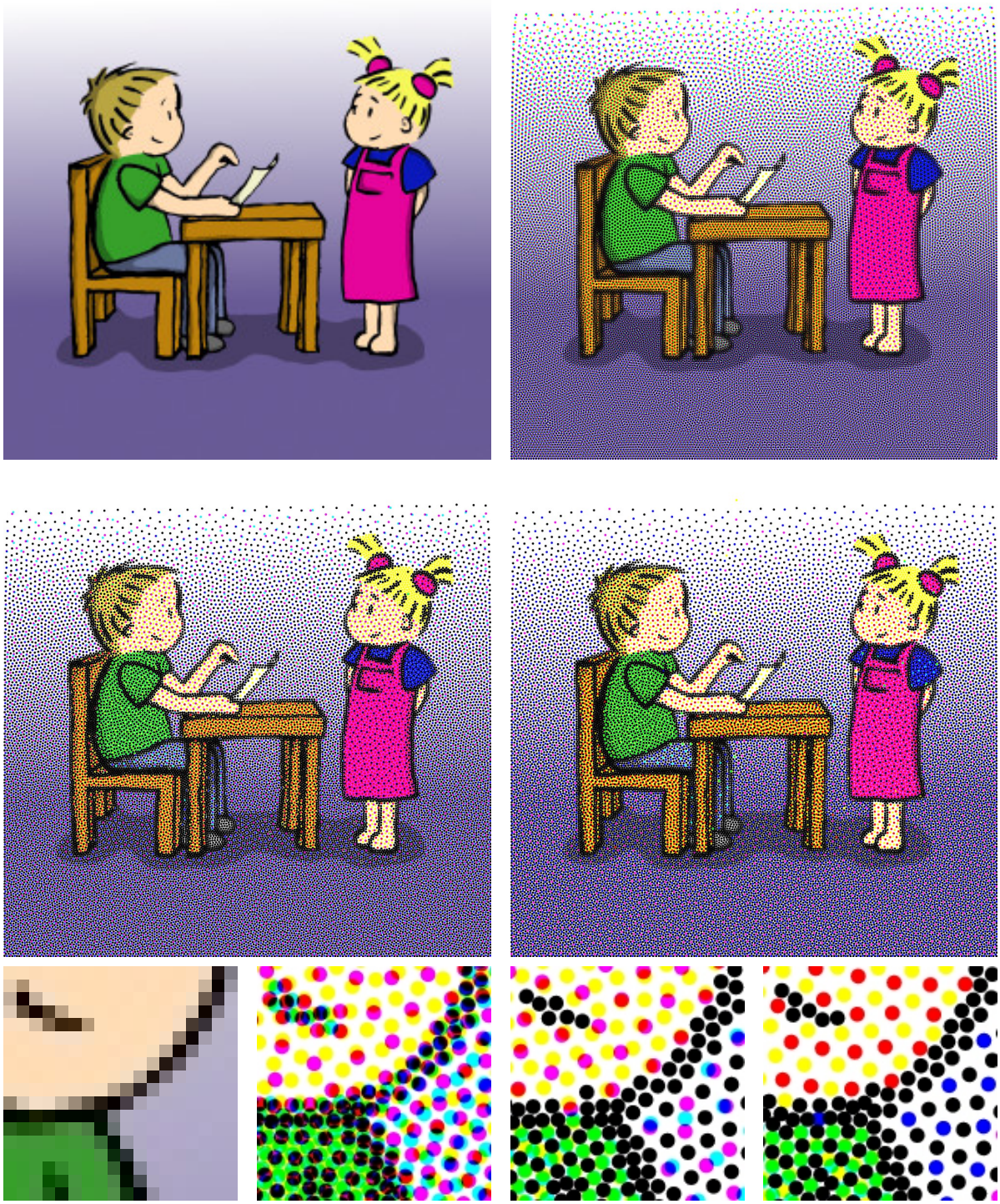


Figure 9.24.: Halftoning of the colour image “comic” in complex colour spaces. **Top left**: Original image (256×256) **Top right**: Using the CMY colour model. **Middle left**: Using the CMYK colour model. **Middle right**: Using the asymmetric CMY-RB-K colour model. **Bottom row**: Zooms into the images.

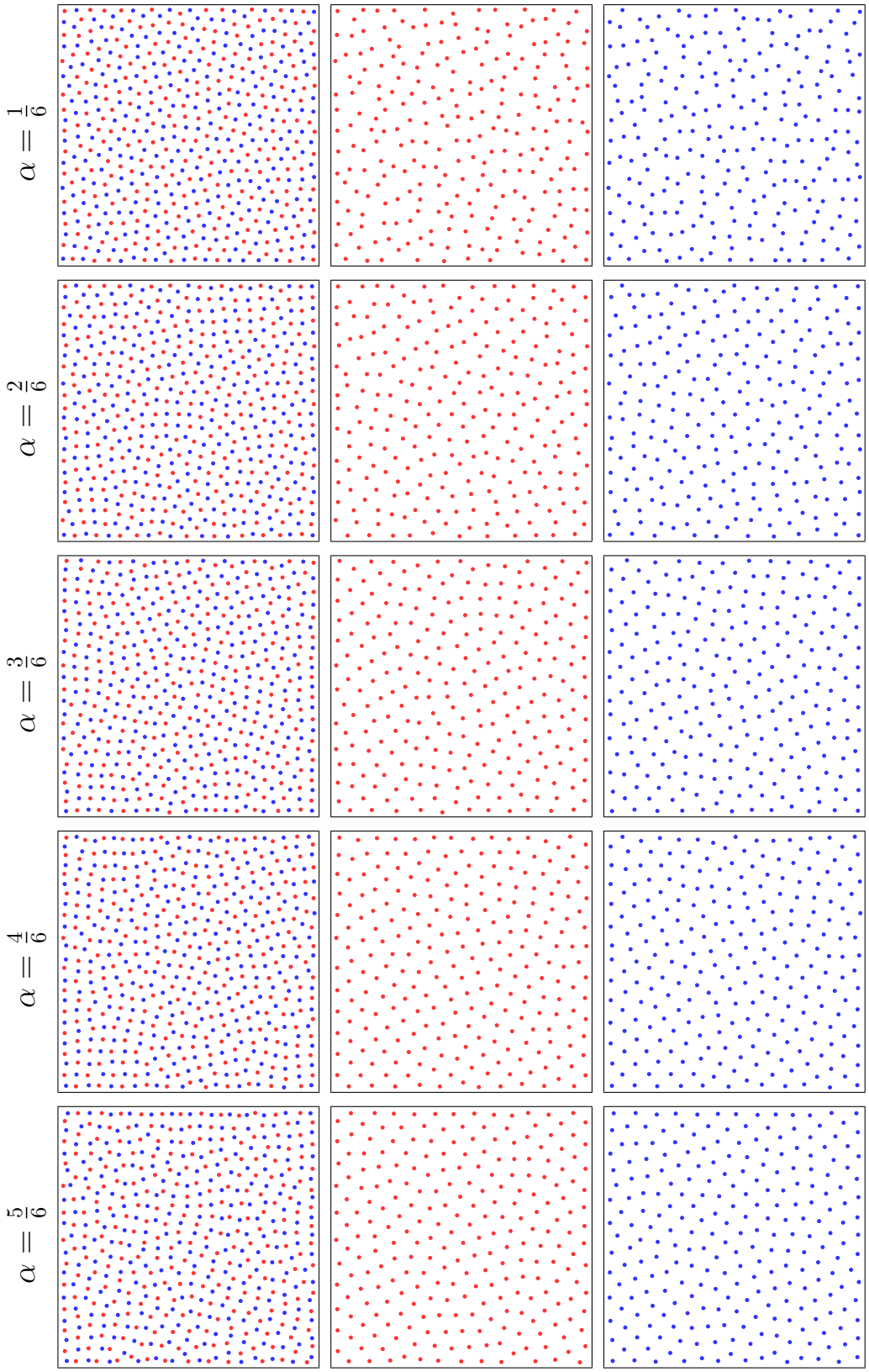


Figure 9.25.: Influence of the weight α on multi-class sampling of a uniform image with particle of equal size. Shown are, from top to bottom, the total set and the individual sets for $\alpha \in \{\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}\}$.

Figure 9.25 illustrates the influence of the parameter α on the tradeoff between good representations of the total set of particles and good representations of subclasses in multi-class sampling applications. Small α cause clusters in the subsets corresponding to single classes, while large α generate clusters in the total set to improve the approximation of the single classes. Depending on which representation is more important, α should therefore be chosen application dependent.

As explained before, a similar tradeoff exists in multi-class sampling, which can be steered using the parameter C. This is illustrated in Figure 9.26. With $C = 1$, only the overall image is approximated well but a bad approximation is obtained when considering only small or large particles. An increasing C forces the larger particles to additionally approximate the image well without the smaller particles, which also yields an appropriate distribution of only the small particles. If C is chosen too large, however, the large particles nearly ignore the positions of the small particles, which yields an excellent approximation with only the large particles, but an inappropriate approximation when considering all particles.

Instead of forcing the large particles to approximate the image well without the small particles, one might also require the same for the small particles. The resulting halftone is also illustrated in Figure 9.26. One can notice at second glance that the small particles cluster around the larger ones in dark image regions. Apart from this issue, halftoning works well, though.

In Figure 9.27, we show results for multi-class sampling with two particle sizes. In this experiment, the ratio between the areas covered by a small and a large particles is $4 : 9$. As indicated in Figure 9.27, there are a total of six different subsets in which the particles should be distributed correctly. Thus, finding a good sampling is difficult in this situation. Nevertheless, it can be seen that our results are rather smooth, and that there are no striking artefacts which hint at the presence of a second class or particles with a different size, respectively. Note that, as explained before, it is possible to adapt the parameters C and α to increase the smoothness of specific classes, if desired.

Next, we show real-world example images created with our multi-class, multi-size halftoning algorithm, see Figure 9.28. To create these images, we sampled the densities shown with particles from three classes, which mutually repelled each other. Additionally, three different particle sizes were used for each class. The images shown are obtained after rendering a (randomly rotated) image of a piece of ham, mushrooms, or salami at the places indicated by the dot locations, which is scaled by the size of the dot.

Hatching

As a last extension, we illustrate the visual performance of our hatching algorithm using the images "skull" (200×300) and "bird" (463×229). The first image was hatched with 20000 lines with a desired length of 2 and 4 pixels, respectively, see Figure 9.29. Note that edge enhancement via unsharp masking has been performed on the input image to improve the visual appearance of the result. Thus, even small details are visible very well.

For the second image, we used 10000 lines and 10000 dots to approximate the input image. Figure 9.30 shows results with and without edge enhancement, as well as a result for with edge enhancement was only used for the dots, but not for the lines. Note that the latter is

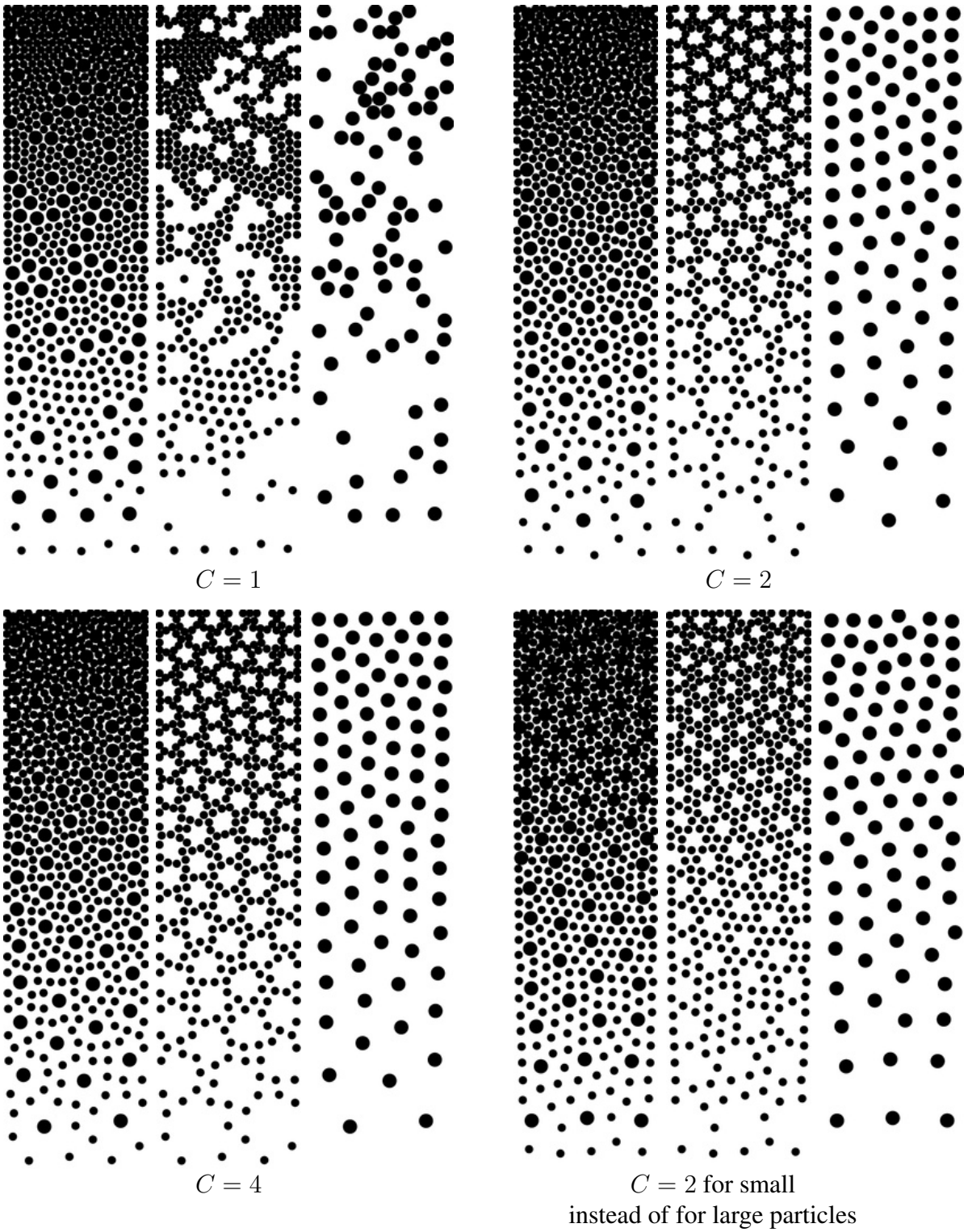


Figure 9.26.: Influence of the tradeoff parameter C on halftoning with multiple dot sizes of the ramp shown in Figure 9.11. Shown are, from top left to bottom right, the results with $C = 1$, $C = 2$, $C = 4$, and the result when setting $C = 2$ for the small instead of for the large particles.

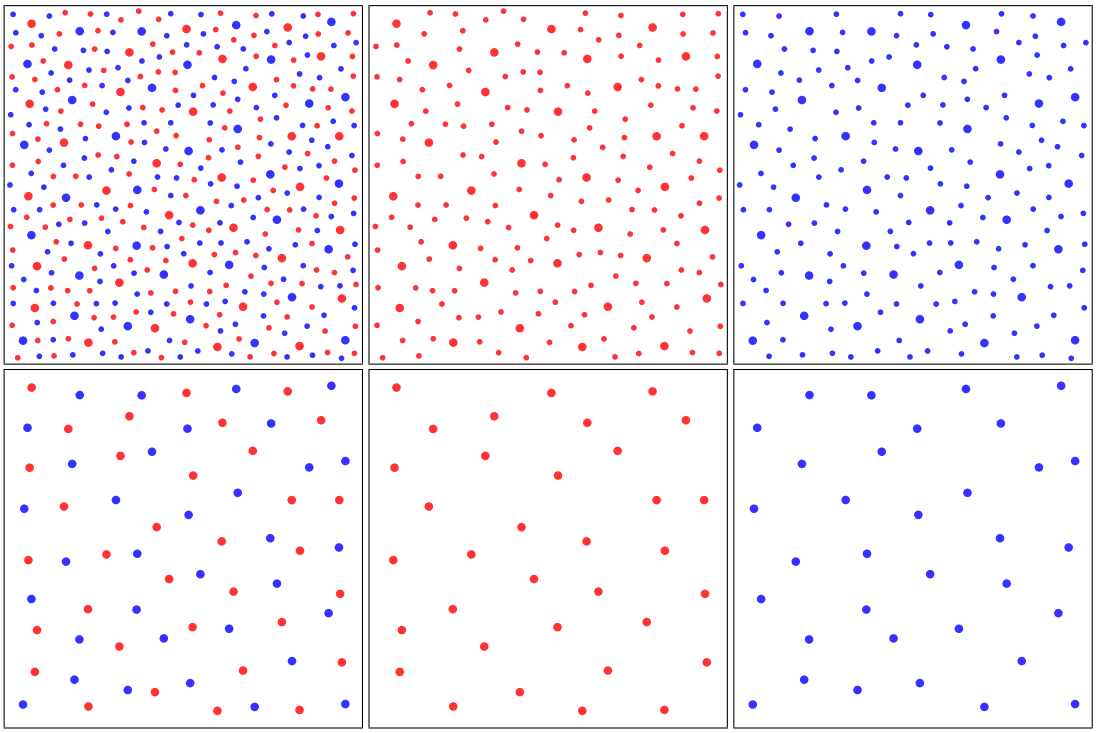


Figure 9.27.: Multi-class sampling of a uniform density with two particles sizes. **Top row**: Results with both dot sizes. **Bottom row**: Results with large particles only. **Left column**: Results with both colours. **Middle and right column**: Red and blue subsets, respectively.

straightforward in our approach: We simply precompute the attractive image forces with and without edge enhancement, and use the appropriate force in the evolution.

9.4.5. Dependence on Initialisation

As explained on Page 152, we are interested in the steady-state of the particle evolution, and not in the evolution itself. Still, it is reasonable to look at the particle evolution to see how fast the algorithm converges.

Two particle evolutions for the image "trui" with different initialisation are shown in Figure 9.31. Even with a random initialisation (upper two rows), the result of our algorithm locally looks very good after only a few iteration. However, too many black particles gather at the upper boundary while there are particles missing at the lower part of the image. This is due to the fact that too many particles are placed in the upper part of the image in the initial particle arrangement. Consequently, these particles are repelled too strongly from the repulsive forces, and try to move out of the image. After some iterations, the "superfluous" particles force other particles to locations in which particles are still missing, and move to the positions of the repelled particles. Thus, the particle positions converge even with a bad initialisation.

The better the initial particle arrangement is, the less probable such problems are. Thus, as explained in Section 9.2.1, the initial particles locations are chosen randomly in such a

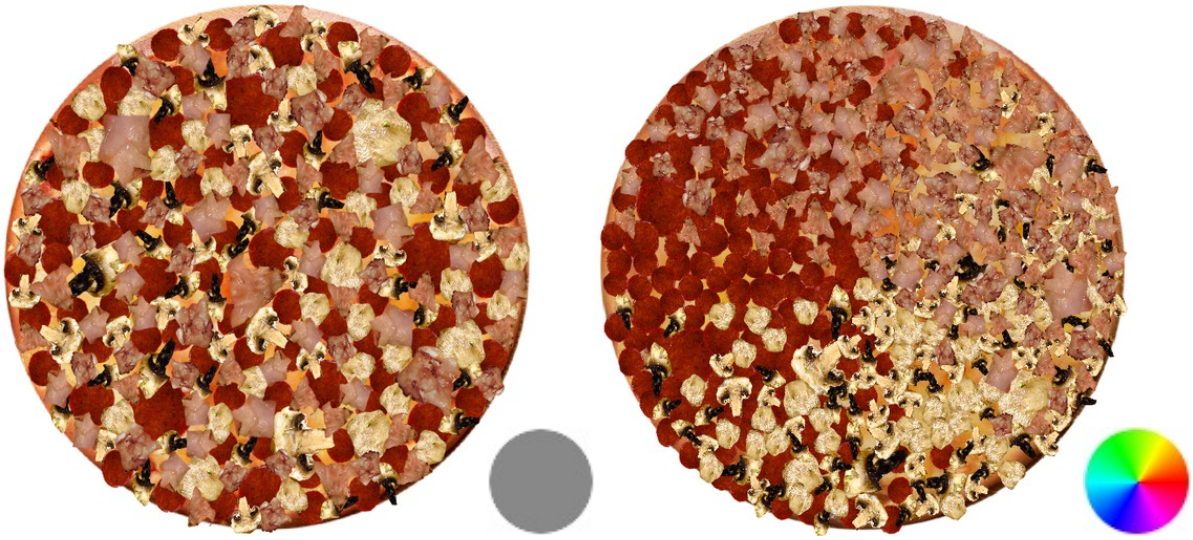


Figure 9.28.: Real-world scenario in which our multi-class multi-size sampling is used to determine the positions of ham (User:fugutabetai_shyashin, flickr.com, CC-BY-NC-SA), mushrooms (User:cyclonebill, flickr.com, CC-BY-SA), and salami (User:vagabondvince310, flickr.com, CC-BY). The circle in the bottom right corners indicate the input images used to place the ingredients. In all images, three different ingredient sizes were used.

way that dark regions probably contain more particles than bright regions for all experiments shown here. Using this initialisation, the particles converge a bit faster than with a random placement. This is shown in Figure 9.31. Note that in this specific example, there are still particles that gather at the upper right boundary. This can happen since the particle placement is still random.

For our dithering algorithm, the situation is very similar, as shown in Figure 9.32. When looking carefully at these images, one can see that a good initialisation is even more important in the discrete setting, as only a rather small force pulls particles at the image boundary back into the image. Due to the attractive forces from the grid locations, particles may stay longer at the boundary than desired. Thus, one can see that there are some artefacts at the upper part of the right boundary even after 256 iterations. Again, we would like to stress that these problems do not appear with a good initialisation, e.g. from another fast dithering approach, but that we have not used such initialisation in order not to bias the results of our algorithm. Furthermore, these problems also disappear when using more iterations.

9.4.6. Evaluation using Image Interpolation

As shown in Section 2.1, it is possible to reconstruct an image from a few selected point; the reconstruction quality depends on the choice of interpolation points. In general, finding the best interpolation points is a difficult task. When linear isotropic diffusion is used for inpainting, however, it was shown in [20] that the interpolation points should be distributed according to the absolute value of the Laplacian. Thus, Belhachmi *et al.* proposed to dither

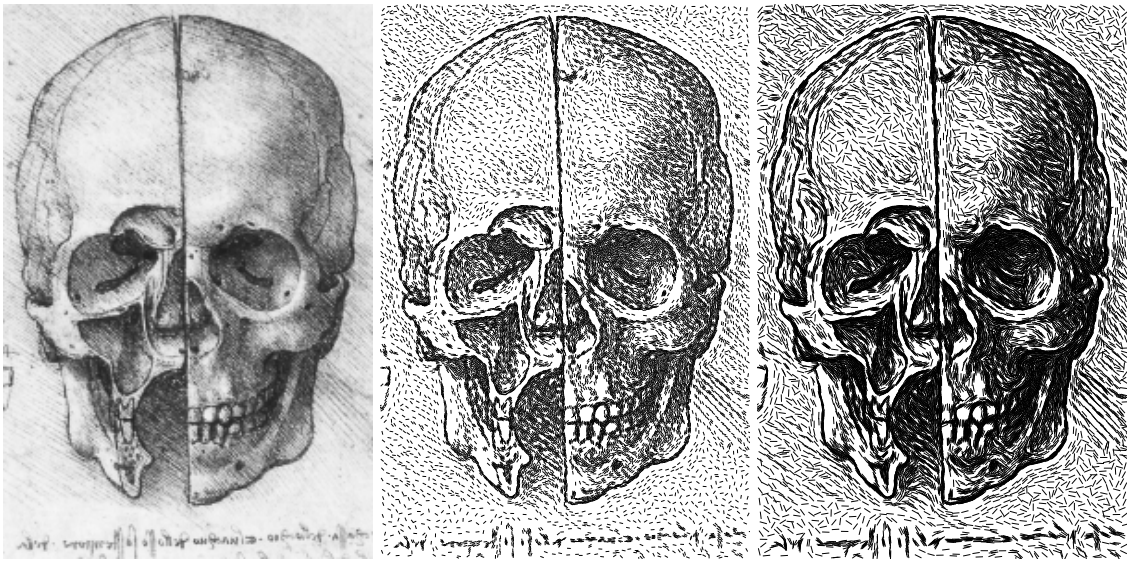


Figure 9.29.: Hatching result of the image "skull" (200×300). **Left**: Original. **Middle and Right**: Result with desired line length of 2 and 4 pixels, respectively.

an image containing the absolute values of the Laplacian of the smoothed input image to find good interpolation points. In order to obtain the desired number of mask points, the image should be scaled correctly before dithering is applied. This algorithm is illustrated in Figure 9.33.

In [20], the Floyd-Steinberg algorithm was used for dithering. Here we will evaluate the inpainting quality with different dithering algorithms to check which algorithm performs best in this context. The reason for these experiments is that we will employ a halftoning algorithm to find a good inpainting mask for inpainting with homogeneous diffusion in the video compression algorithms presented in Chapter 10.

For the first experiment, we used the image "house" shown in Figure 9.33(a). The reconstruction error with four different dithering algorithms with 500, 1000, 1500, and 2000 mask points are shown in Figure 9.34. As can be seen, there is a huge difference in the reconstruction quality when using different dithering algorithms. Independent of both the number of mask points and the amount of presmoothing, our dithering algorithm performs best. The methods by Zhou and Fang ties with the method by Ostromoukhov for second place for most parameters. The method by Floyd-Steinberg is often worst, but reaches the second place for some parameter settings. Furthermore, one can see that using a good dithering algorithm is especially important when a low number of points are used, and when little presmoothing is done.

In Figure 9.35, the same experiment was done using the images "skull" and "trui". In both experiments, 500 mask points were used. As can be seen, similar results are obtained for these additional experiments.

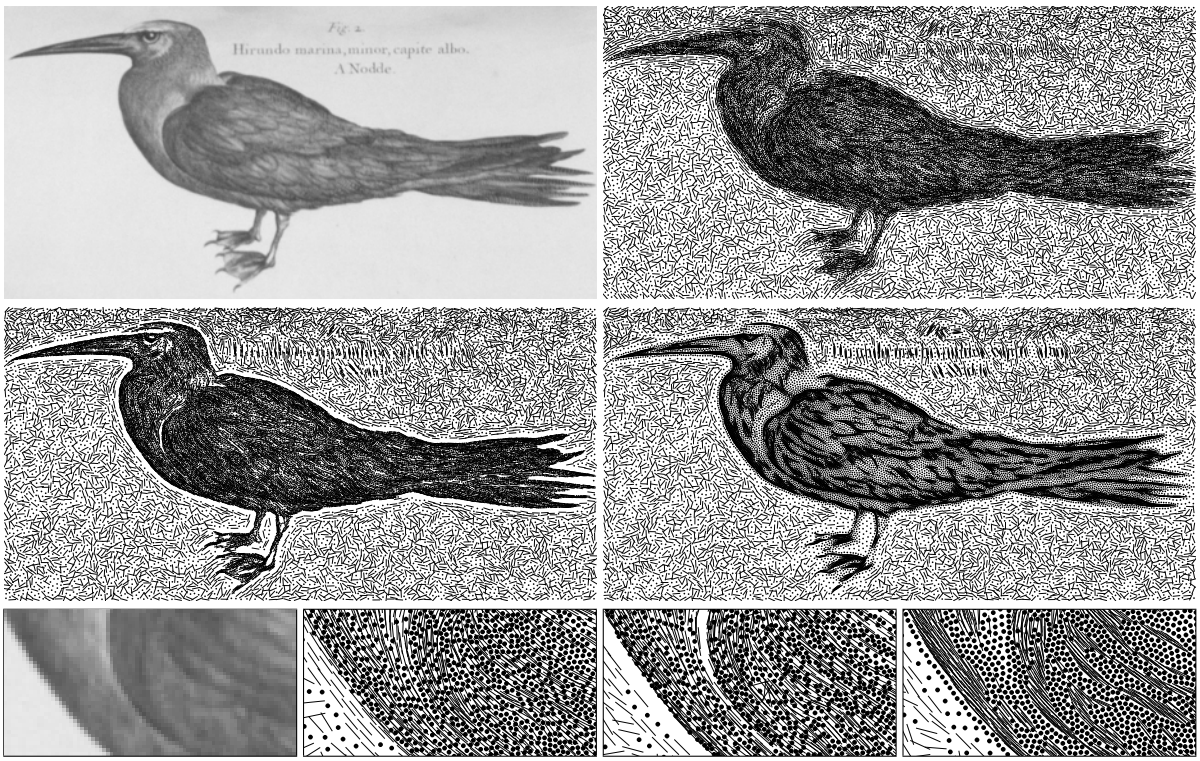


Figure 9.30.: Image "bird" (463×229) and three results with 10000 lines and 10000 dots. **Top left**: Original. **Top right**: Result without edge enhancement. **Middle left**: Result with edge enhancement. **Middle right**: Result with edge enhancement only for the lines. **Bottom**: Magnifications to chest region, respectively.

9.4.7. Runtime and Memory Requirement

Our algorithm consists of the three major parts explained in Section 9.2, namely particle initialisation, force field computation, and particle evolution. The time required for the first step is negligible, whereas the time required for the other two steps is illustrated in Figure 9.36. Note that the force field computation only depends on the image size, and not on the number of particles while it is vice-versa for the particle evolution. Thus, initialisation time and time required per particle evolution is given individually in Figure 9.36. The jumps visible in both graphs are due to CUDA specifics: On our GPU (NVidia GTX 285), we use 128 blocks with 512 threads each; Thus, there is a discontinuity for $128 \cdot 512 = 65536$ particles/pixels. All further discontinuities are due to the number of 30 streaming multiprocessors, which cause jumps at $n \cdot 65536 + m \cdot 30 \cdot 512$ particles/pixels, i.e. whenever the number of blocks exceeds a multiple of 30.

For the simple implementation described in Section 9.2, the initialisation time of our algorithm (in Landau notation) is $O(n^2)$, where n is the number of pixels in the original image. For one particle evolution, the required time is in $O(m^2)$, where m is the number of particles. Furthermore, there is a linear dependence on the number of iterations in the particle evolution step. However, it is possible to design asymptotically faster algorithms, e.g. by using a

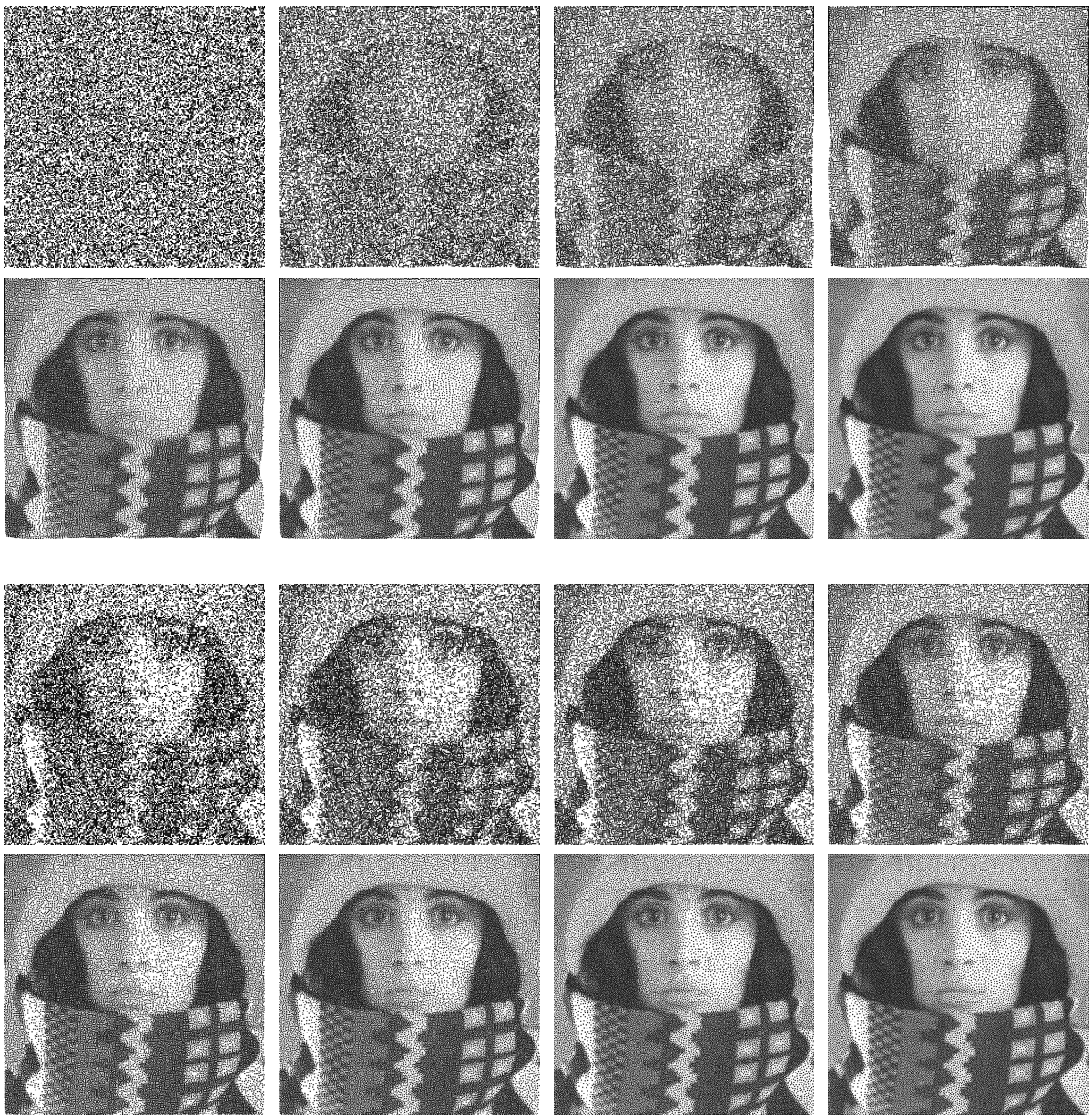


Figure 9.31.: Evolution of our halftoning algorithm with a random initialisation (first two rows), and with our initialisation (see Section 9.2.1, last two rows). For each initialisation, the initial particle placement and the resulting image after 1, 2, 4, 8, 16, 64, and 256 iterations are shown.

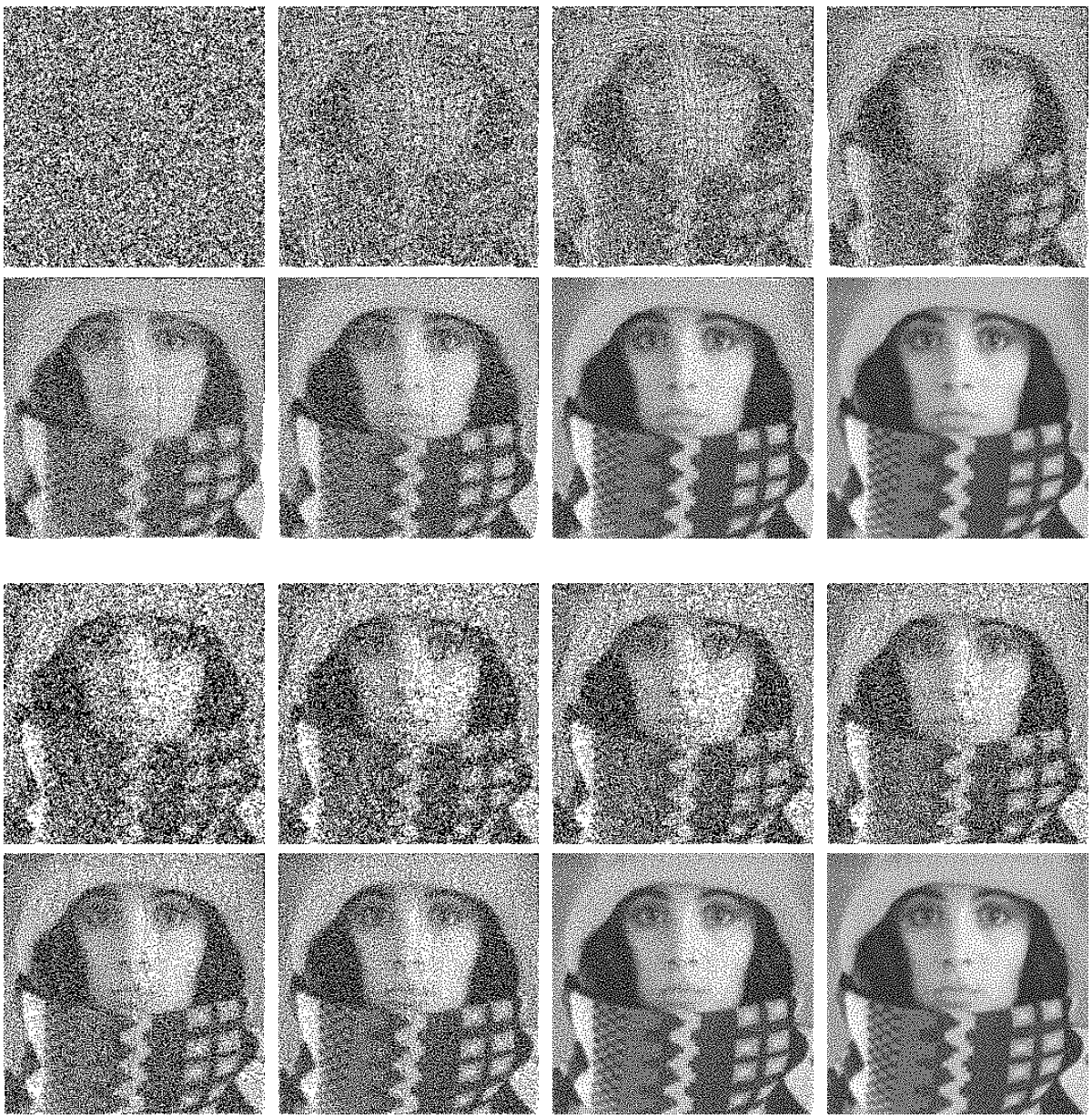


Figure 9.32.: Evolution of our dithering algorithm with a random initialisation (first two rows), and with our initialisation (see Section 9.2.1, last two rows). For each initialisation, the initial particle placement and the resulting image after 1, 2, 4, 8, 16, 64, and 256 iterations are shown.

(a) Initial image (b) Laplace (c) Mask (d) Reconstructed image

Figure 9.33.: Pipeline used for evaluating dithering algorithms using linear inpainting: From an initial image (a), the magnitude of the Laplacian is computed (b) and scaled corresponding to the desired number of mask paints. The mask points are then obtained by dithering this image (c). The last image shows the reconstruction from these points using linear diffusion.

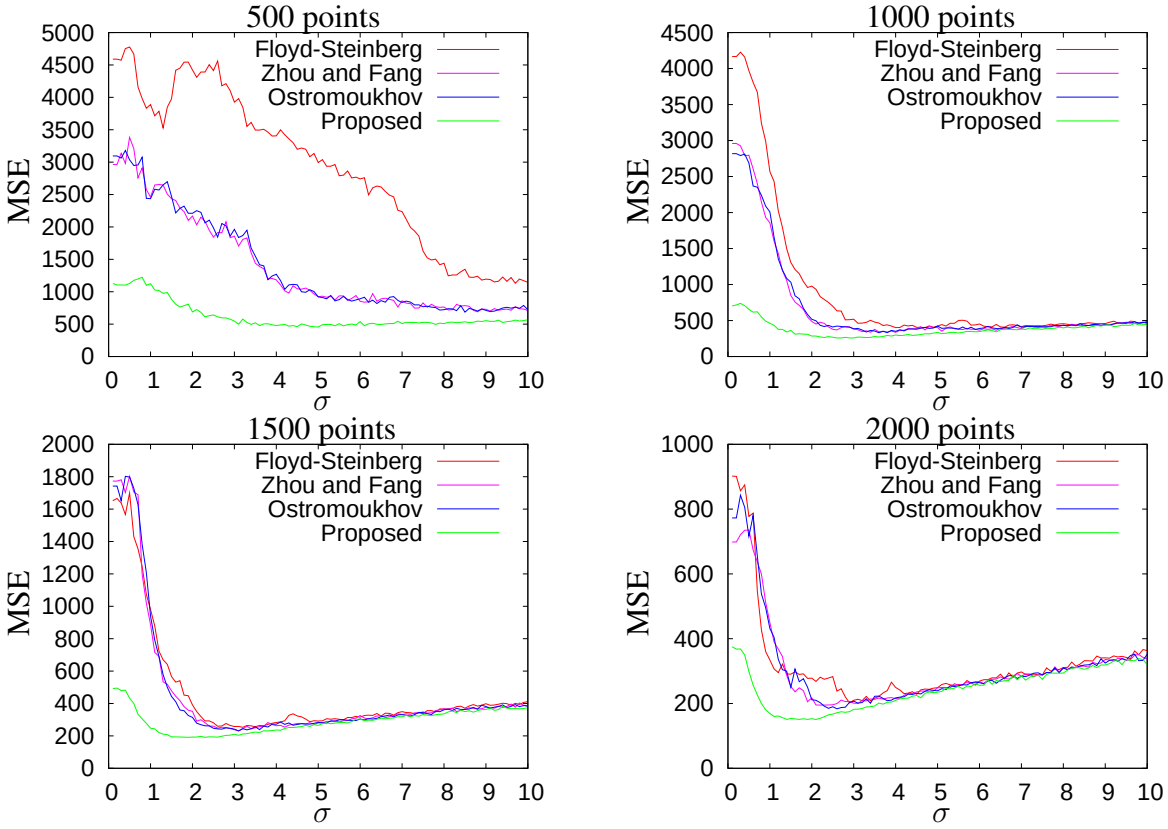


Figure 9.34.: Mean squared errors obtained using linear image interpolation with 500 mask points (top left), 1000 mask points (top right), 1500 mask points (bottom left), and 2000 mask points (bottom right) for the image “house”. The mask points are obtained by dithering the (scaled) absolute value of the Laplacian of the smoothed input image, where a Gaussian with standard deviation σ is used for smoothing. The different graphs in each plot correspond to different dithering methods.

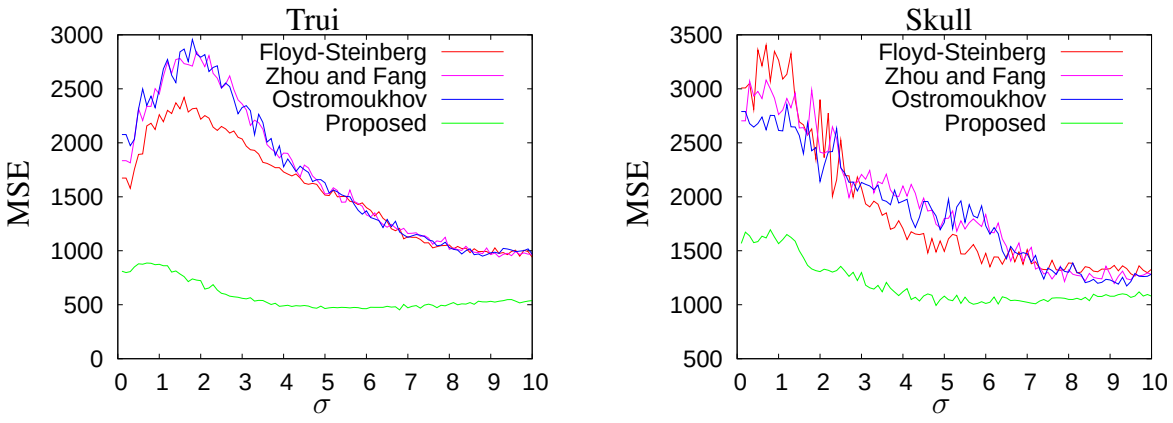


Figure 9.35.: Results for the same experiment as in Figure 9.34 with the images "trui" (left) and "skull" (right). For both experiments, 500 mask points were used.

fast summation procedure based on the fast Fourier transform at non-equidistant knots. This reduces the algorithmic complexity of one particle evolution to $O(m \log m)$. Details can be found in [273], while the GPU-version of this approach is described in [104].

Even with our suboptimal initialisation, we have seen in the last section that reasonable results are obtained quite quickly. As an example, $6.5s$ are required for 300 iterations on a 256×256 image with 16384 particles (25%) using the simple implementation explained in Section 9.2.

The memory complexity of our algorithm in Landau notation is $O(m + n)$, where n and m are the number of pixels and particles, respectively. It uses only very few arrays of size m and n. Although the algorithm by Balzer *et al.* also has a memory requirement of $O(m + n)$, it still requires much more memory in practise, as it first creates a large amount of points for each dot in the final image. In their example implementation and the experiments presented here, 1024 points are used for each dot. Using less points is possible, but will deteriorate the results. Thus, our algorithm requires significantly less memory in practice.

9.5. Conclusion

"It's more fun to arrive at conclusion than to justify it."

Malcolm S. Forbes (1919–1990)

In this chapter, a halftoning approach motivated by electrostatics has been explained, which can be used for applications such as dithering, multi-class sampling, or stippling. By using a large range of different evaluation procedures, we have shown that it yields excellent results which outperform those of all other state-of-the-art halftoning approaches. Our algorithm is very flexible and can easily be adapted to specific applications. It performs edge enhancement and avoids visual artefacts if and only if these features are desired, and is trivial to adapt to

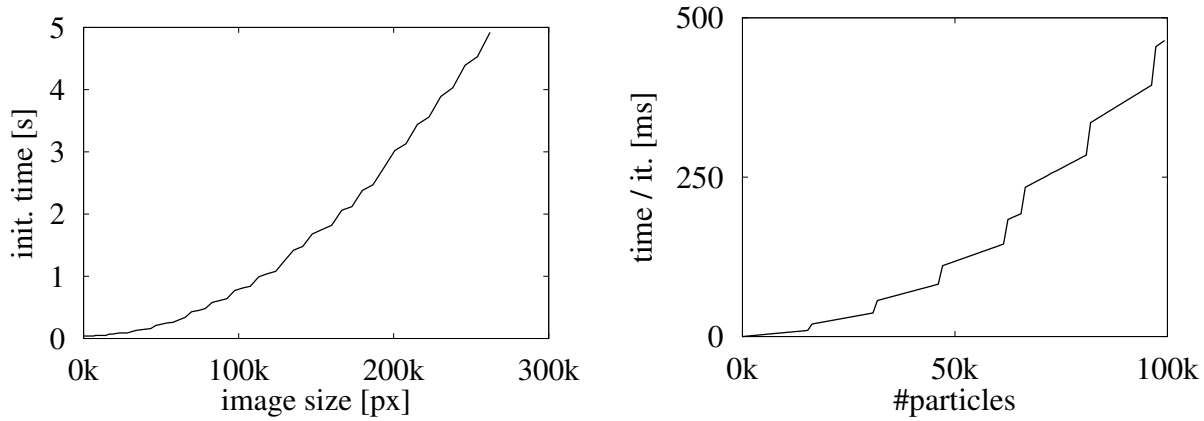


Figure 9.36.: Time required by our algorithm for initialisation (left), and for particle evolution (right). Note that the time required for initialisation only depends on the image size, while the time required for particle evolution only depends on the number of particles. The discontinuities in both graphs are due to CUDA specifics explained in Section 9.4.7.

any printer resolution. Results are excellent independent of the viewing distance or printer resolution.

This concludes the presentation of the third main topic of this thesis. In the next chapter, we show how to combine the developed algorithms into several video compression codecs.

10 Video Compression

"A good video can make all the difference."

Brian May (*1947)

Finally, we come to the last topic we consider in this thesis, which is *video compression*, i.e. the task to reduce the amount of data necessary to represent a sequence of images. Typical video compression codecs consists of two independent parts. The first part is a codec used for compressing the image data. This is the topic of this thesis, while the second part, which is the compression of audio information, will not be discussed here. Most popular video compression algorithms combine pixels within a square to so-called *macroblocks* on which motion estimation is performed to estimate the appearance of these block in nearby frames. In addition to the estimated motion, the appearance change within each block is stored. While decompressing, this information is used to reconstruct the individual frames of the video.

The first official digital video encoding standard was published in 1984 by *CCITT* (*Comité Consultatif International Téléphonique et Télégraphique*, *International Telegraph and Telephone Consultative Committee*), and was called *H.120* [52]. Although its compression quality turned out to be insufficient for practical use, it is the basis of other video codecs such as the *H.261* codec [53], or the more recent *H.263* codec [122]. Both *H.261* and *H.263* [122, 95] are mainly used for video-conferencing.

In 1993, the *Moving Picture Experts Group* (*MPEG*)[1] released *MPEG-1* (part 2) [126], a video codec built upon H.261. Based on these early works, several other well-known and widely-used video codecs have been developed. As result of a joint work of both groups, the standards *H.262*/*MPEG-2* (part 2) [127, 121] and *H.264*/*MPEG-4 AVC* ("Advanced Video

[1]The official designation is ISO/IEC JTC1/SC29 WG11 - Coding of moving pictures and audio (ISO/IEC Joint Technical Committee 1, Subcommittee 29, Working Group 11).

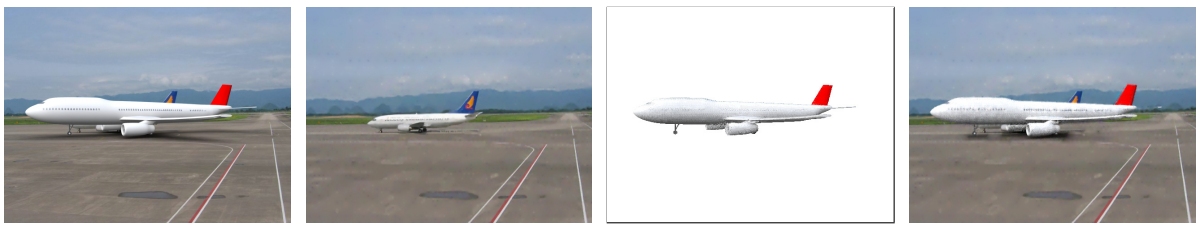


Figure 10.1.: Illustration of the different steps occurring in one of our video compression algorithms. **From left to right**: Initial image of size 640×480, compressed background image (compression ratio: 98 to 1), projected object model in tracked pose, and final result when using information from 5000 additional points whose locations are obtained by dithering.

Coding") [292, 186, 129, 124] have evolved. Currently, these groups are working on a project with the working title *HEVC* (*High Efficiency Video Coding*), which shall either become a new standard or an extension of *H.264/AVC* [66].

There are many more general video compression algorithms most of which are proprietary. Thus, details about those codecs are often not publicly available. Prominent examples for such codecs include *AVS* (*Audio Video Standard*), *Bink*, *RealVideo*, *Smacker video*, *VP8*, and *WMV* (*Windows Media Video*). For a detailed comparison of different general video codecs, we refer to the surveys and introductions by Sullivan and Wiegand [264], Sayood [231], Strutz [259], and Abomhara *et al.* [2].

Apart from general purpose video compression algorithms, there are also video compression codecs using *model-based coding* schemes, often abbreviated as *MBC*. We will give a short summary of the ideas presented in this context, but refer to the survey by Pearson [198], and the thesis by Yao [300] for a detailed overview of the field.

The original idea behind model-based coding schemes, namely to compress moving parts in videos by storing a model and some motion parameters, was introduced in 1983 by Forchheimer and Fahlander [81]. Since this initial work, different approaches to model-based video coding have been pursued.

In [274], for example, Toelg and Poggio propose an approach that uses a small set of example images containing a human face with different facial expressions. With the help of a pose estimation algorithm, a novel view or facial expression is constructed from these example images. Vieux *et al.* use a similar approach for their "Orthonormal Basis Coding" in [284].

While these two approaches are based on 2-D example images, there are also many approaches which use full 3-D models. In [133], a partial description of a model-based coding based on the *MPEG-4* standard is presented. However, this method requires manual interaction, and it is not specified how the texture is stored. Although these questions are answered in the work by Granai *et al.* [100], both methods only explain how to compress the foreground using model-based coding, and ignore the background.

Huang and Tand [114] proposed to save the complete first and last image of the video sequence using JPEG, and to synthesise intermediate frames from estimated motion vectors. In [11], motion compensated temporal interpolation is used to estimate the background onto which the 3-D object model is projected.

Various extensions to model-based coding have been proposed, e.g. for varying illumination conditions [72] or for handling different facial expressions [71, 196]. The latter idea was even included into the *MPEG-4* standard as *facial animation parameters* (or *FAP*s) [193].

However, all papers described only show results for faces seen from the front – sometimes including the shoulder region – and most algorithms are even specialised to this situation. Consequently, they are well suited for compressing videos for video-conferencing, but not for general videos.

In this chapter, we introduce three closely related video compression algorithms, which are based on the ideas of model-based coding, see Figure 10.1. Since they are not specialised to faces or other specific objects, they can handle any kind of moving objects. The next section first introduces our simple approach to estimate the appearance of the object models, followed by a detailed explanation of our three video compression algorithms. We continue with a thorough evaluation of these approaches in Section 10.2 before concluding the chapter.

10.1. Codecs

"We might be the holographic image of a two-dimensional structure"

Brian Greene (*1963)

In this section, we introduce several variants of the video compression algorithm sketched in the introduction. Therefore, we first present the simple algorithm we have used to estimate the colour of each vertex of the model used for tracking, followed by a detailed description of the video compression codecs. Furthermore, we explain the file format used by our algorithms.

10.1.1. Model Colouring

Our first two video compression algorithms require a coloured model of the tracked objects. However, we assumed that only uncoloured object models are available for tracking. Thus, it is necessary to estimate the appearance the object models. This is the topic of this section.

The idea how to find the colour of each vertex in the object model is quite simple and straightforward: Since we have used the uncoloured object model to track the object throughout the image sequence, we know to which image point each vertex of the object is projected in each frame. Thus, to get the colour of a specific vertex $\boldsymbol{v}$, it is theoretically sufficient to set the colour of this vertex to the colour of the image point $u(\boldsymbol{Pv})$, i.e. to the colour of the point to which the vertex $\boldsymbol{v}$ is projected. Thereby, $\boldsymbol{P}$ denotes the corresponding projection matrix and u is one image of the sequence.

In practise, however, this will yield a very poor colouring, as tracking results are inaccurate in general, e.g. due to inaccurate object models. Furthermore, one has to check if the vertex is actually visible in a certain image. Changing lighting conditions and other related problems further decrease the quality of this simple approximation. Thus, we not only consider a single image but utilise the colours over all views and frames of the sequence in which the vertex $\boldsymbol{v}$ is visible. Note that it is easy to check if $\boldsymbol{v}$ is visible when using OpenGL [247].

To handle some of the tracking inaccuracies, we make use of the fact that the background image b is known: If the colour of $u(\boldsymbol{Pv})$ is similar to $b(\boldsymbol{Pv})$, i.e. to the colour of the background image at the same image position, we assume that the estimated position of the vertex is inaccurate. Consequently, we disregard this pixel when estimating the colour of $\boldsymbol{v}$. Although this does not take care of all cases in which estimated vertex positions are inaccurate, this step handles some critical cases in which the colour differs strongly from the real colour of $\boldsymbol{v}$. To further stabilise the estimation, we simply average the colours obtained by different images. An example model coloured with this approach is shown in Figure 10.2.

10.1.2. Basic Model Based Video Codec

Our first video compression algorithm is a simple model-based coding algorithm: As a first step, we track all objects moving in the video with our tracking algorithm explained in Chapter 5 to 8. Thereby, we assume that the necessary data – a projection matrix for each view,

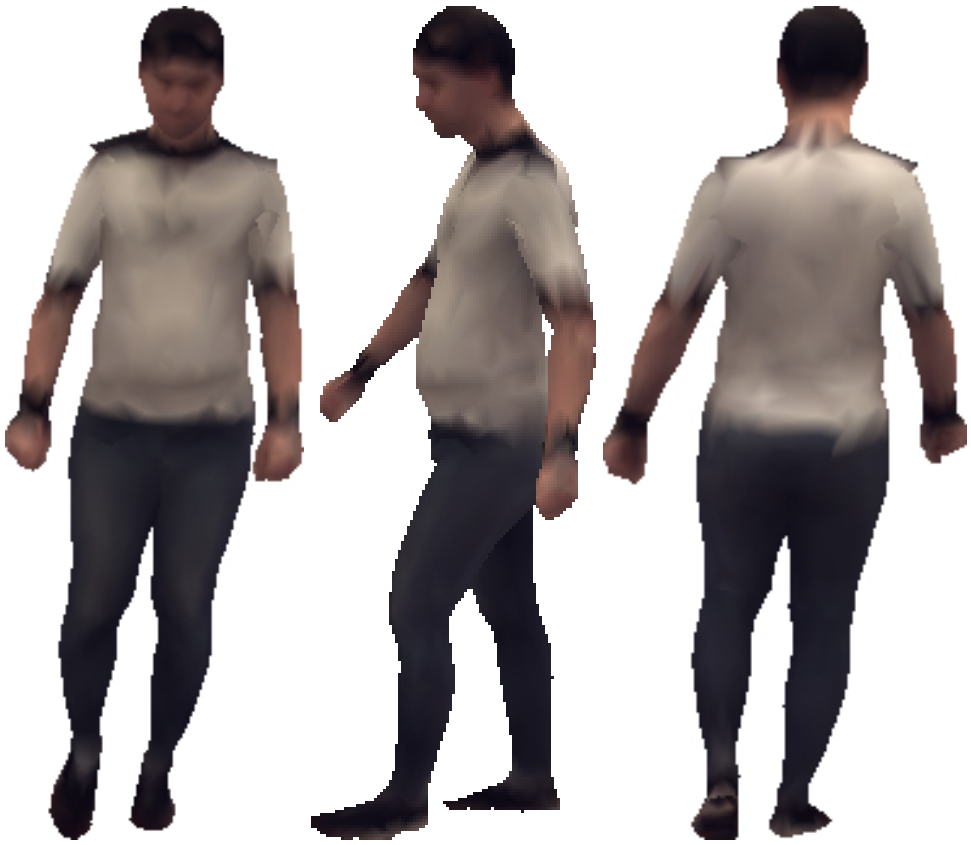


Figure 10.2.: Model coloured with the approach explained in Section 10.1.1, shown from three different views. Here, the HumanEva-II sequence S4 was used to obtain the colours.

an (uncoloured) object model, and a pose initialisation – are given. Additionally, we assume that the background in each view to be saved is static. We will discuss how to overcome these restrictions later.

As a second step, the background image is reconstructed, unless it is known in advance. Although background reconstruction is a difficult topic in general, it is fortunately very easy in our setting: First, each frame of the video sequence is segmented into fore- and background region by using the tracking results. Then, each pixel of the background image is taken from the image in which this pixel is farthest from the projected object model. This is illustrated in Figure 10.3 using two images. Note that points which are always occluded by the object model can be chosen arbitrarily, as they will not be needed later.

Thirdly, the tracking results obtained are used to estimate the colour of each vertex of the given object models, as explained in Section 10.1.1.

Next, the background image is compressed using the PDE-based image compression algorithm introduced in Chapter 3.

Finally, the compressed background image, the coloured object model, the projection matrices, and the estimated pose parameters are saved and compressed using a general purpose entropy coder. Details about this step are given in Section 10.1.5.

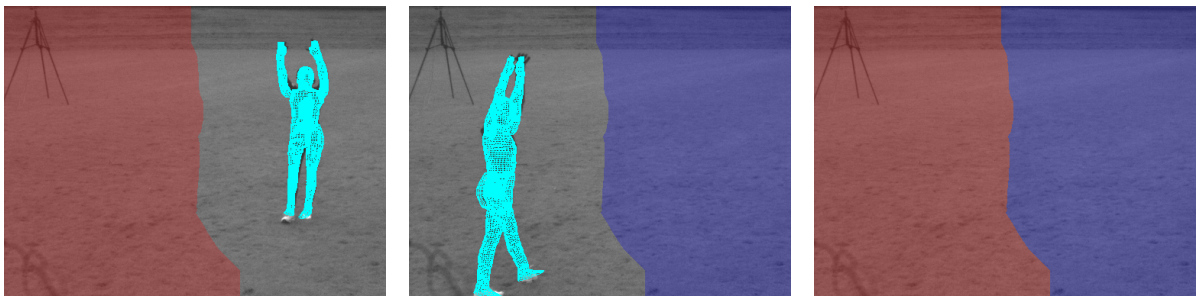


Figure 10.3.: Illustration of a simple way to reconstruct a static background image using 3-D tracking results. The first two images show two frames into which the model in the estimated pose has been projected in cyan. Each pixel is taken from the image for which this pixel is furthest from the projected model. The pixels taken from the first image are shown in red, those from the second image in blue. The reconstructed background is shown in the third image.

This concludes the description of our first video encoding algorithm. We denote this codec by *MB*, which is an abbreviation for *m*odel *b*ased codec.

Reconstructing the video is straightforward: First, the background image is uncompressed. Then, each frame of the video is restored by projecting the object model using the corresponding pose and projection matrix loaded from the compressed video file onto the background image. These two steps are illustrated in the middle images in Figure 10.1.

For some cases, this codec is sufficient to yield a good reconstruction. However, problems such as tracking failures or model inaccuracies can lead to a bad video quality, as the projected object model looks quite different than the original video frame. Thus, we introduce an algorithm that can cope with such problems in the next section.

10.1.3. Video Codec with Residual Coding

The codec introduced in this section is an extension of the *MB* codec from last section. It performs all steps done by *MB* but additionally encodes the residual image, i.e. the difference image between the image compressed by the *MB* codec and the original image.

This residual image is stored as sets of single pixels and reconstructed by inpainting with homogeneous diffusion, as explained in Section 2.1. Note that we will use the additional pixel information only to account for inaccuracies close to the moving foreground, as problems in the background region are easier solved by storing the background image more accurately.

In this situation, we do not use the adaptive grid utilised in our still image compression algorithm in Chapter 3 due to several reasons. First of all the inpainting mask K cannot be chosen arbitrarily if it is restricted to positions on a grid. This restriction limits the possible reconstruction quality, and is not present in our new approach introduced in this section. Furthermore, since we only use points in and close to the foreground, the image region to be saved is not rectangular. Thus, the rectangular grid structure cannot be used without special adaptations.

First, let us consider only the first frame of a grey-valued video. When we use inpainting with homogeneous diffusion (see Section 2.1.1), we know that the interpolation points should be distributed according to the magnitude of the Laplacian of a smoothed version of the image, as discussed in [20]. Thus, we can use a dithering algorithm to obtain the inpainting mask. We use our electrostatic dithering algorithm introduced in Chapter 9, as we have shown in Section 9.4.6 that it is best suited for this situation.

The optimal standard deviation σ of the Gaussian used to smooth the residual image is not known in advance. Thus, we try different standard deviations in our algorithm and use the σ for which the best approximation is obtained. As σ is not needed to decompress the video, this does not increase the final file size.

Another optimisation done at this point is to restrict the domain of the dithering algorithm to a region containing the foreground region and points close to it. This is realised by first computing the Euclidean distance transform of the projected objects, followed by computing the Laplacian only for those pixels whose distance to the projected object models is not larger than a threshold T. For all experiments shown here we have used the fixed value $T = 20$. However, adapting T to the current image is likely to further improve the reconstruction quality. A detailed evaluation is part of our future work.

The image obtained by computing the Laplacian of the initial image is scaled before it is used as input image in the dithering algorithm. This scaling ensures that the total amount of points in the obtained inpainting mask can be chosen depending on the video quality and file size desired. To store the position of the points in the inpainting mask K, we use the *JBIG* file format [128, 123], which is a lossless image compression algorithm for binary images. The grey-value of all points specified by K are quantised uniformly to q values and stored separately. For all experiments done here, we have used $q = 17$. This will be justified by the experiments shown in Figure 10.16. For colour images, each channel is quantised and stored separately.

To summarise, the algorithm for coding the residual image in the first frame performs the following step: First, the residual image is computed. This residual image is converted to a grey-valued image, which is smoothed with a 2-D Gaussian with standard deviation σ. Next, the distance transform of the projected object model is computed, and the Laplacian of all pixels whose distance to the projected object model does not exceed the threshold T are computed. The remainder of the image is set to zero such that only inaccuracies close to the foreground are considered. The resulting image is scaled appropriately such that the electrostatic dithering algorithm used next gives the desired number of particles/interpolation points $|\mathcal{P}|$. The dithered image is stored using *JBIG*, and the corresponding quantised grey values of the residual image are saved to a different file, see Section 10.1.5.

If the video is in colour, we use the same approach, but first compute a grey-valued variant of the difference image. Thereby, different colour models may be used. However, results are quite similar independent of the colour model used, according to first experiments. Thus, we just add the red, green, and blue colour channels to get the grey-valued variant of the difference image.

In the remaining frames, the residual image is coded in a similar way. However, there are some important differences. First of all, we use the dithering result from the previous frame as initialisation of the new dithering process. Furthermore, we perform much fewer iterations

of electrostatic dithering. Since it is possible that two particles have been mapped to the same grid location in one iteration, the particles are deterministically shifted up to a quarter of a pixel before starting the iterations, though. More precisely, the vector $(m \cdot \sin(\phi), m \cdot \cos(\phi))^\top$ is added to each particle location, where ϕ is a random number in $[0, 2\pi)$ and m is a random number in $[0, 0.25)$.

These optimisations do not only speed up the computations, but also allow to store the particle movements relative to the last frame instead of the particle positions. This reduces the amount of data that must be stored if the number of particles is reasonable. While it is trivial to obtain the particle motion when using electrostatic dithering, this is a very hard problem (or not possible at all) using any of the other dithering algorithm discussed in Chapter 9. This is another reason why it is advantageous to use our electrostatic dithering algorithm in this context. The colour values at the new inpainting positions are quantised as in the first frame.

All this data is stored to different files, and compressed using a general purpose entropy coder. Details about the file format will follow in Section 10.1.5. We call this *M*odel-*B*ased codec which stores the *d*ifference image by *H*alftoning *MB+dH*.

To reconstruct the video, we first execute the steps explained for our first codec *MB*. Afterwards, the inpainting mask is loaded (first frame) or reconstructed using the stored particle motion and the particle locations in the preceding frame. As in the case of still image compression, the stored colour information is mapped to the points indicated by the inpainting mask. Furthermore, we add all points whose distance from the projected object model is larger than the threshold T (see above) to the inpainting mask, and set their colour to zero. This ensures that the difference image is zero at the transition between the region adapted by the stored residual image and the unmodified background image.

As can be seen in the last image in Figure 10.1, this can significantly improve the reconstruction quality of the resulting video sequence.

10.1.4. Third Video Compression Algorithm

Although the algorithm *MB+dH* yields significantly better results than the codec *MB*, videos compressed with this approach often appear noisy. This is especially true when only few particles are used to store the residual image.

Let us first illustrate why *MB+dH* can result in noisy images: Even when tracking results are only slightly wrong, there is one (possibly thin) stripe in which the background is visible instead of the object tracked, or vice versa. At the boundary of such stripes, the Laplacian of the difference image is high. Thus, many particles are placed on these boundary to compensate for this problem. The regions next to these boundaries are much smoother, resulting in only a few particles being localised next to these boundaries.

However, when only a few particles are used, there is not enough information to obtain a sharp boundary. As a consequence, the reconstructed residual image might bleed, i.e. the colour values at one side of the border of the stripe are influenced too strongly by the colours from the other side of the border. In contrast to a slight misplacement of the object model, such a problem is easily spotted by humans.

Our third codec prevents this bleeding at the cost of a higher reconstruction error. Instead of projecting a coloured model and coding the residual image as additional information, our

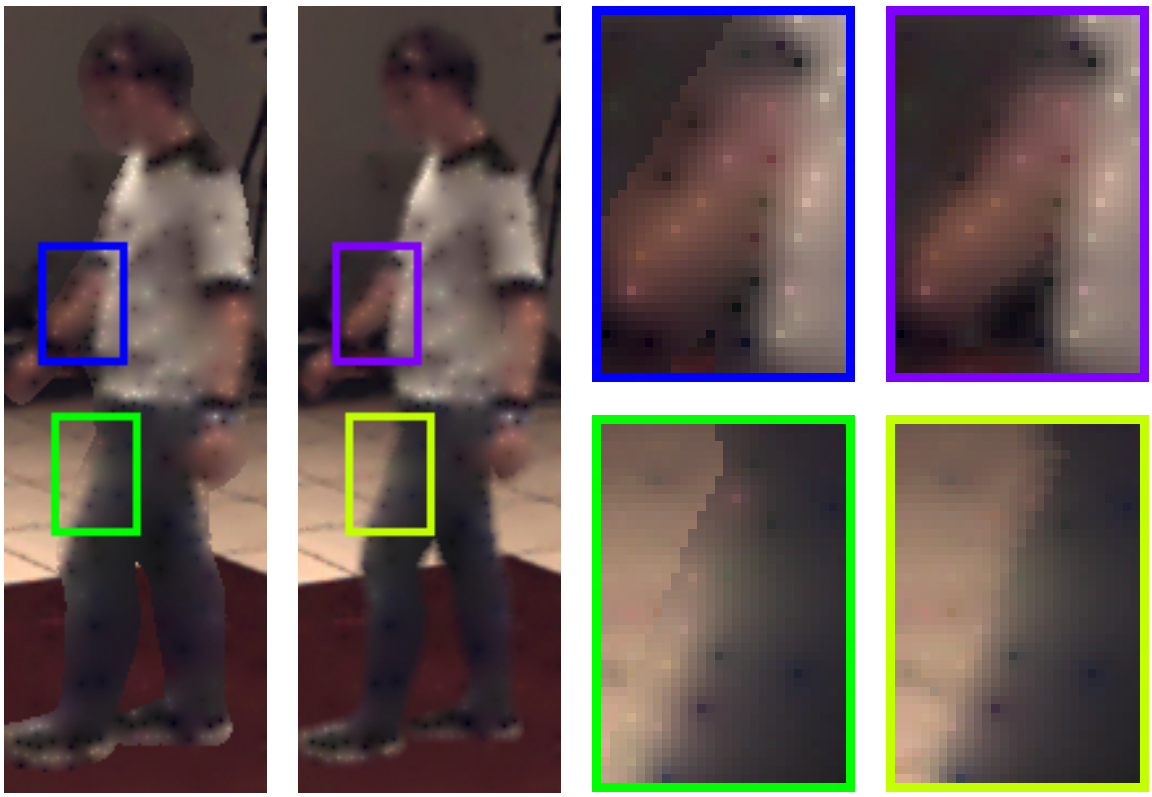


Figure 10.4.: Illustration why blending is used in the codec *MB+mH*. When overwriting the background image with the reconstructed foreground region, visually unpleasant artefacts can appear (first column). With blending (second column), this is not the case. This can be seen especially well in the magnifications of the right arm and the left leg shown in the third column (without blending) and forth column (with blending), respectively. However, blending can also significantly change the appearance, as visible between the legs.

next approach uses the (uncoloured) object model and its pose merely to find the image region F in which the tracked object is visible. As in our codec *MB+dH*, additional grey or colour values are saved at locations obtained by dithering. This allows to reconstruct the foreground region F. The resulting *M*odel-*B*ased codec which stores the *m*odel region by *H*alftoning is called *MB+mH*.

There are some important differences to the codec *MB+dH*: Now, we dither the magnitude of the Laplacian of the image to be saved instead of that of the residual image. Thus, we must set the boundary pixels to the value of the background image at the appropriate positions instead of setting it to zero.

To deal with tracking errors, we extend the foreground region F by adding all points M whose distance to this region is less than T pixels. This increases the likelihood that the whole foreground object is covered by $F \cup M$. As explained in the last section, the threshold T can be adapted to the sequence to obtain optimal results. For our experiments, we use the constant value $T = 5$.

Note that the particles use a colour obtained from the original image while the boundary is initialised to the compressed background image, as only this image is available in the decompression step. Thus, simply overwriting the background image with the reconstructed foreground region can yield a clearly visible boundary, as can be seen in Figure 10.4.

Thus, we propose another approach to combine fore- and background: Let $\boldsymbol{x}$ be a point with distance $d(\boldsymbol{x})$ from the region F, and let $F(\boldsymbol{x})$ and $B(\boldsymbol{x})$ be the colour of the reconstructed foreground and background region, respectively. Then, the point in the final image is computed as

$$(1 - \alpha F(\boldsymbol{x})) + \alpha B(\boldsymbol{x}) \qquad \text{where } \alpha = \min\left(1, \frac{d(\boldsymbol{x})}{T}\right)$$

That is, there are three possible cases:

- Points inside the initial foreground region F are taken from the reconstruction obtained by inpainting.
- Points further away than T, i.e. the points in $\Omega \setminus (F \cup M)$, are taken from the saved background image.
- The remaining points, i.e. the points in M, are reconstructed as a linear combination of fore- and background region.

Through this interpolation, the visible boundaries illustrated in Figure 10.4 will not appear any more. Even though this slightly deteriorates the image quality with respect to quantitative error measures such as the MSE, the result looks more appealing to the human eye. Note that, as can be seen at the legs in Figure 10.4, the visual appearance can be noticeably different with blending.

10.1.5. File Format

The following list summarises the data (possibly) saved by our video compression algorithms. Depending on the variant used, some parts of the data can be omitted.

- For each view:
  - Compressed background image
  - Projection matrix
  - Initial dithering result
- For each view and image:
  - Particle motion
  - Particle colour
- For each moving object:
  - Object model (possibly coloured)
  - Pose parameters for complete video

	HumanEva-II S4			Cart		
method	image size	file size	MSE	image size	file size	MSE
MB	656×490	161223	**102.57**	500×380	68721	52.38
MPEG-1	656×480	2019733	187.25	496×368	202847	48.53
MPEG-4	656×490	537404	210.58	$500/496 \times 380$	112182	**31.52**

Table 10.1.: Overview over the image and file sizes of the videos created using our algorithm *MB*, as well as with *MPEG-1*, and *MPEG-4*. As can be seen, the encoder used for *MPEG-1* cropped all images such that the x and y resolution are a multiple of 16. *MPEG-4* uses the correct resolution, but replaced a stripe of width four at the right hand side of each image with black pixels for one sequence, see Figure 10.11.

For each line in the list, a single file is created. The final file, which contains the complete video, is then created by packing all these files with tar, followed by a compression with PAQ.

As indicated by this list, it is also possible to store several views of the same scene simultaneously. This is especially advantageous with the first codec *MB*, as only one additional background image and a very small projection matrix must be stored for each view.

10.2. Experiments

"The true method of knowledge is experiment."

William Blake (1757–1827)

In this section, we compare the performance of our three video compression codecs against that of the standard codecs *MPEG-1* [126] and *MPEG-4* [129].

The *MPEG-1* videos have been created using the program "mpeg_encode". We used three P-frames between successive I-frames, and three B-frames between other frames, i.e. using the pattern "IBBBPBBBPBBBPBBB". This is a common pattern which usually allows a strong compression. Furthermore, we have set the quantisation levels for all types of frames to the smallest possible value (i.e. 31) to obtain a compression ratio which is as close as possible to the one of our approach.

For *MPEG-4*, we used the Linux program "mencoder" with the codec "msmpeg4v2" and the AVI container format to generate the videos. We used the smallest possible variable bit rate (4000 bits per second) to obtain a small video.

Even though we tried to reduce the file size of the *MPEG-1* and *MPEG-4* videos as much as possible, we could neither create *MPEG-1* nor *MPEG-4* videos which are as small as the ones created with our first approach (see Table 10.1). Note that *MPEG-1* cropped the original images such that width and height are a multiple of 16, see Table 10.1. Moreover, *MPEG-4* replaced a part of the images in one sequence by a black boundary such that the non-black image width is a multiple of 16.

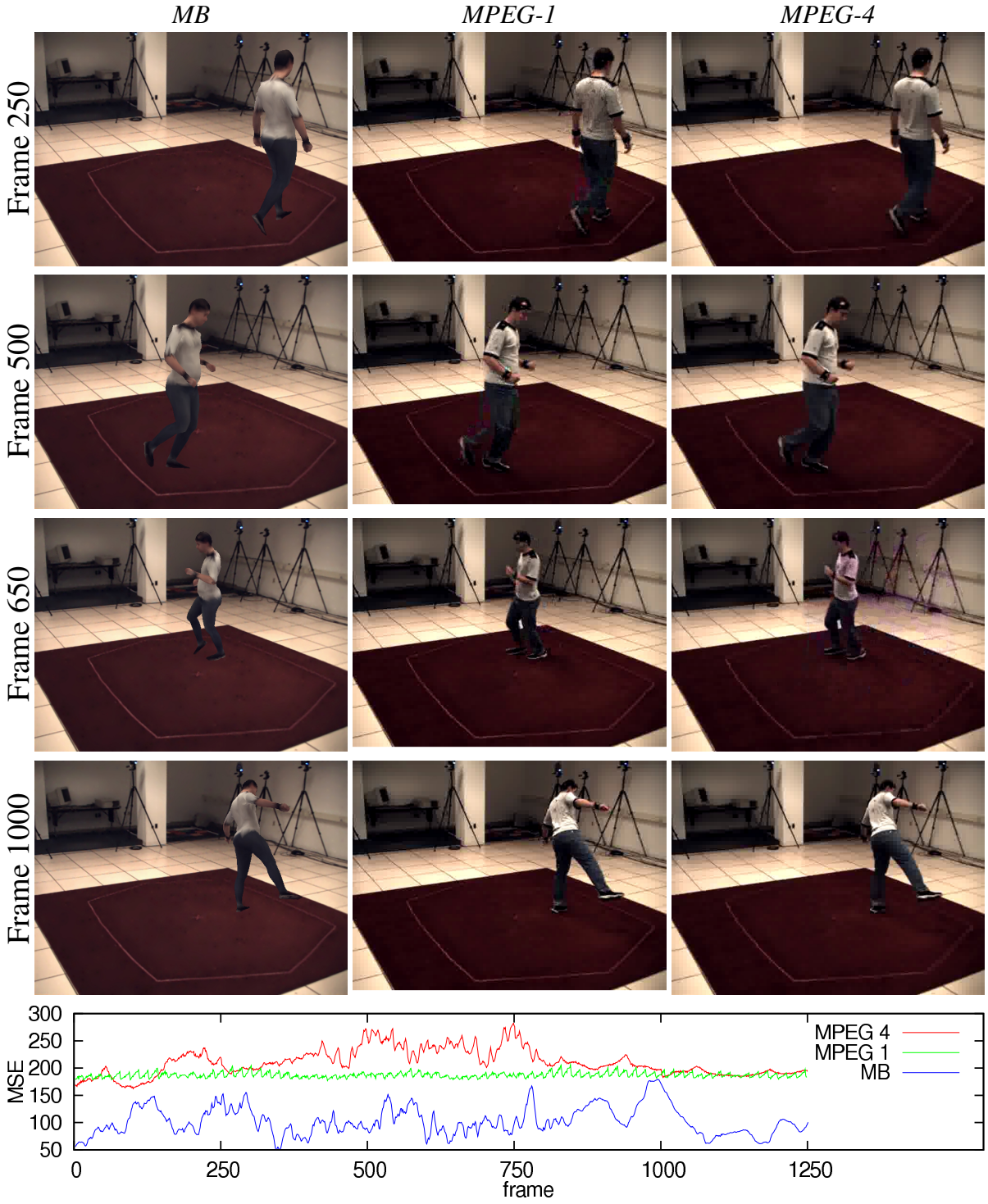


Figure 10.5.: Comparison of our first video compression codec *MB* against *MPEG-1* and *MPEG-4*. Shown are frame 250, 500, 650, and 1000. Note that the files sizes differ strongly, as explained in detail in Section 10.2 and shown in Table 10.1.

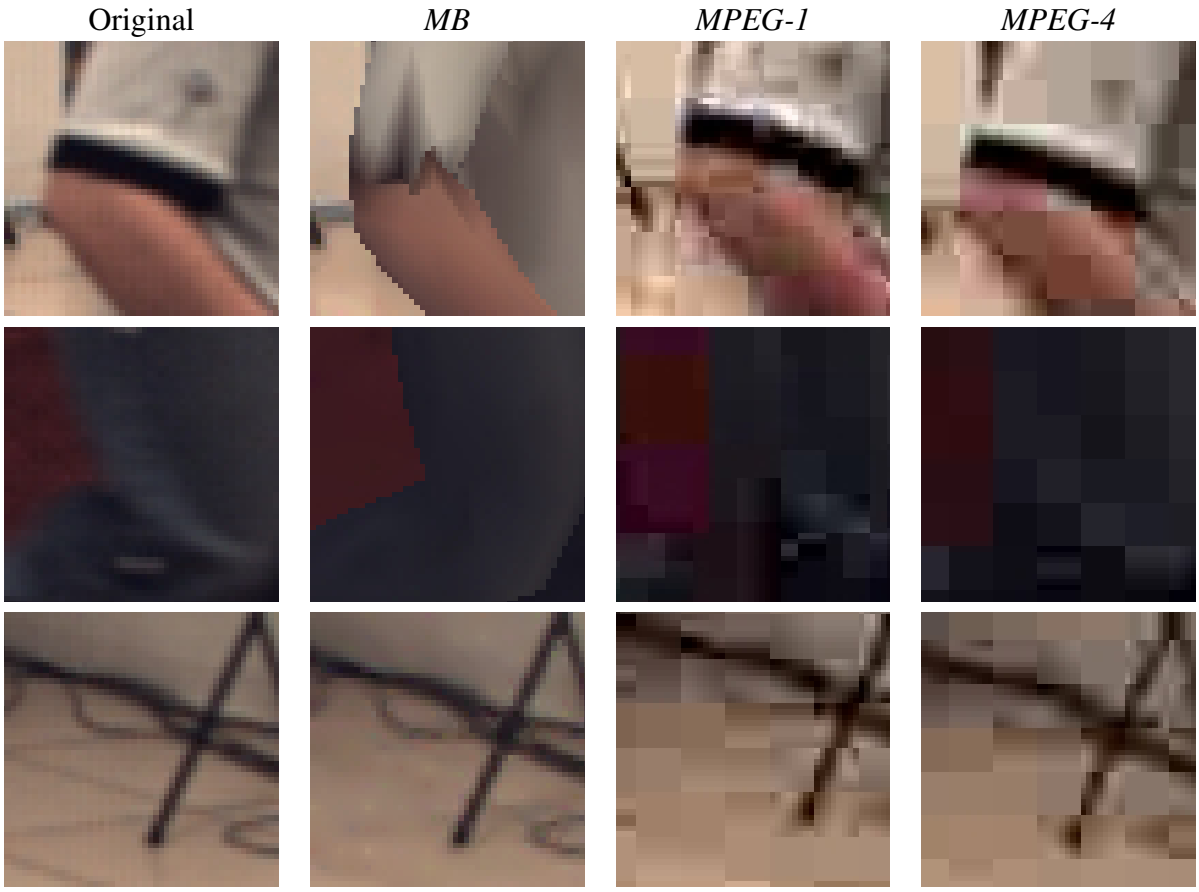


Figure 10.6.: Magnifications from frame 500 of the experiment shown in Figure 10.5. Note the block artifacts when using *MPEG-1* or *MPEG-4*. One can also see that our approach has much sharper boundaries, both in the object and the background region.

In our first experiments, we encode the sequence S4 of the HumanEva-II benchmark. Figure 10.5 shows example frames from the videos created with the codecs *MB*, *MPEG-1*, and *MPEG-4*, as well as a graph showing the MSE obtained per frame with each method. Even though the video created with our approach is considerably smaller than those of the other methods, its error is always below that of *MPEG-1* and *MPEG-4*. This is even true in those frames in which our tracking approach yielded only inaccurate results.

Magnifications for one of the frames shown in Figure 10.5 are given in Figure 10.6. It is easy to observe that our approach creates sharp boundaries, while the approaches from the *MPEG* family generate blocky results. However, our approach yields suboptimal results at the sleeves, due to the rather poor performance of our simple model colouring approach explained in Section 10.1.1.

Results for our codecs *MB+dH* and *MB+mH* with different numbers of particles are shown in Figure 10.7. Again, we show a magnified image part to emphasise the differences, see Figure 10.8. Figure 10.9 illustrates the MSE for each frame, while Table 10.2 contains the file sizes for these experiments before and after applying PAQ.

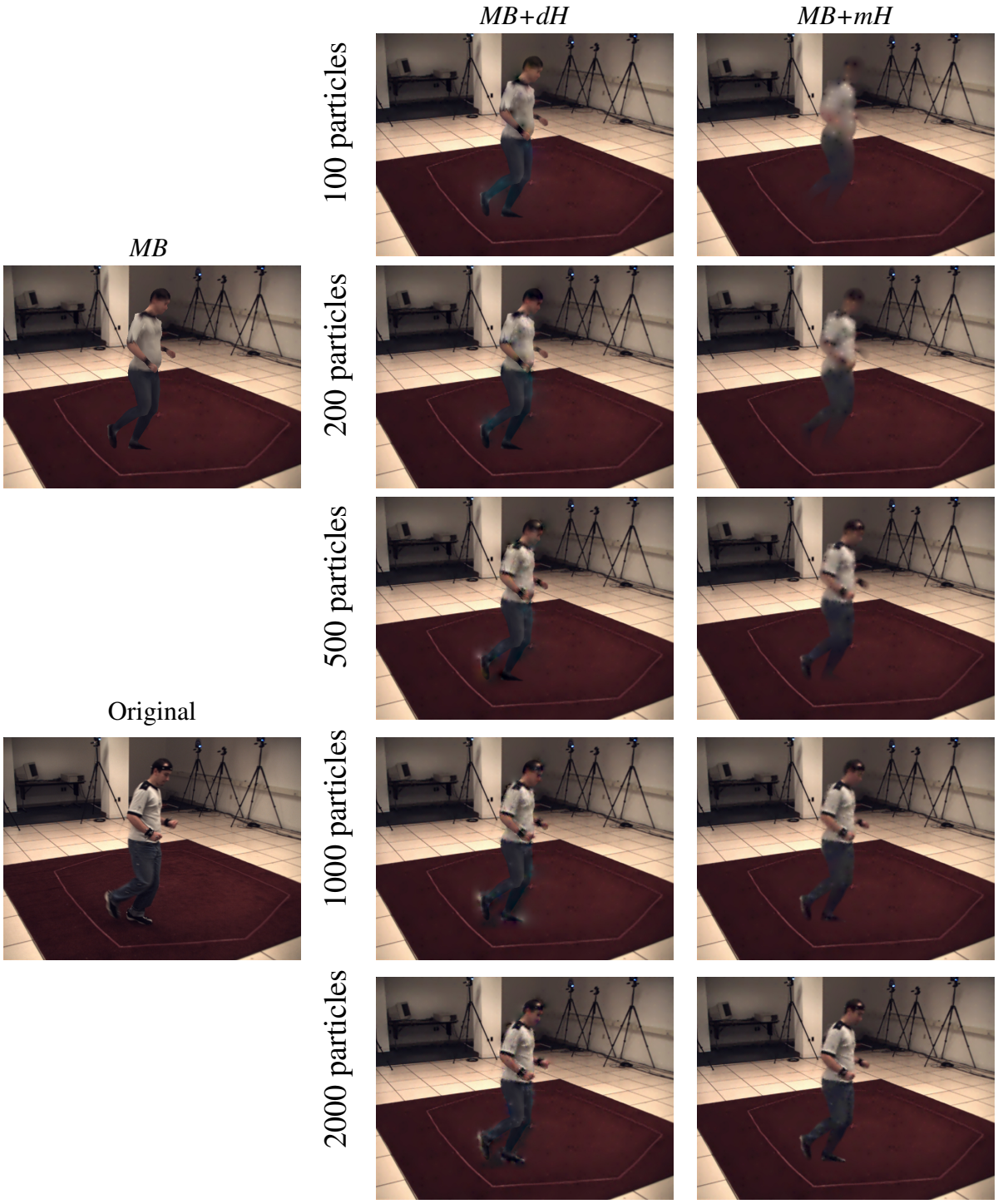


Figure 10.7.: Example results when compressing the HumanEva-II sequence S4 with our codecs *MB+dH* (middle column) and *MB+mH* (right column). Shown are the images in frame 500 of the resulting videos with 100, 200, 500, 1000, and 2000 additional points. The original image and the result with our codec *MB* are shown in the left column. The corresponding images for *MPEG-1* and *MPEG-4* are shown in Figure 10.5.

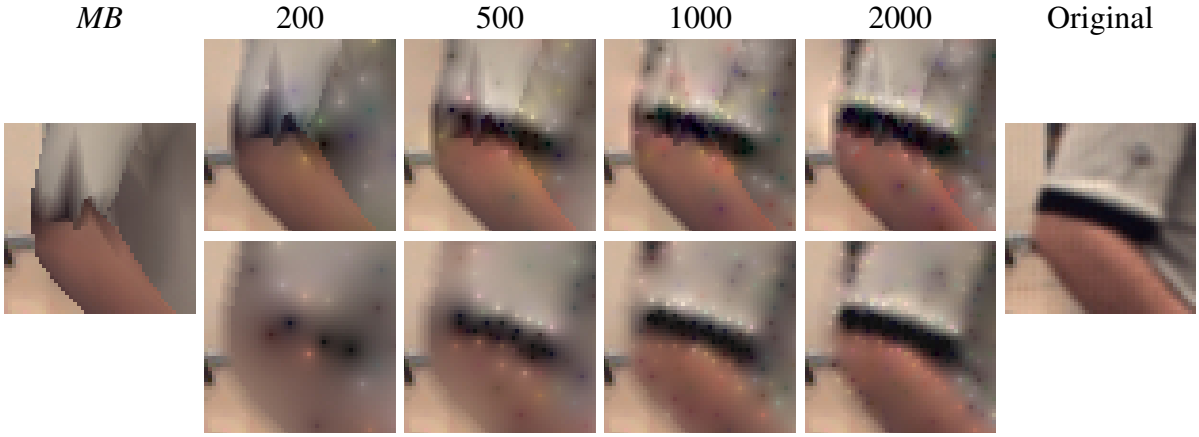


Figure 10.8.: Magnifications from frame 500 of the experiment shown in Figure 10.7. **From left to right**: Result with the codec *MB*, with 200, 500, 1000, and 2000 additional particles, and original image. Results created with the codec *MB+dH* are shown in the upper row, results with the codec *MB+mH* in the lower row.

Codec	Particles	Size before compression	Size after compression	MSE
MB+dH	100	880640	194513	76.53
	200	1505280	267728	63.18
	500	3379200	612452	43.66
	1000	6502400	1256973	30.27
	2000	12748800	2639927	20.50
MB+mH	100	870400	192630	92.97
	200	1495040	264210	67.07
	500	3368960	586778	46.78
	1000	6492160	1208553	37.71
	2000	12738560	2531087	32.43
MB	–	256000	161223	102.57
MPEG-1	–	–	2019733	187.25
MPEG-4	–	–	537404	210.58

Table 10.2.: File sizes when coding the HumanEva-II sequence S4 with the codecs *MB+dH* and *MB+mH* with different numbers of particles. Given are the file sizes before and after applying entropy coding, and the average MSE of the whole sequence. The size and error when using *MB*, *MPEG-1*, and *MPEG-4* are shown as comparison.

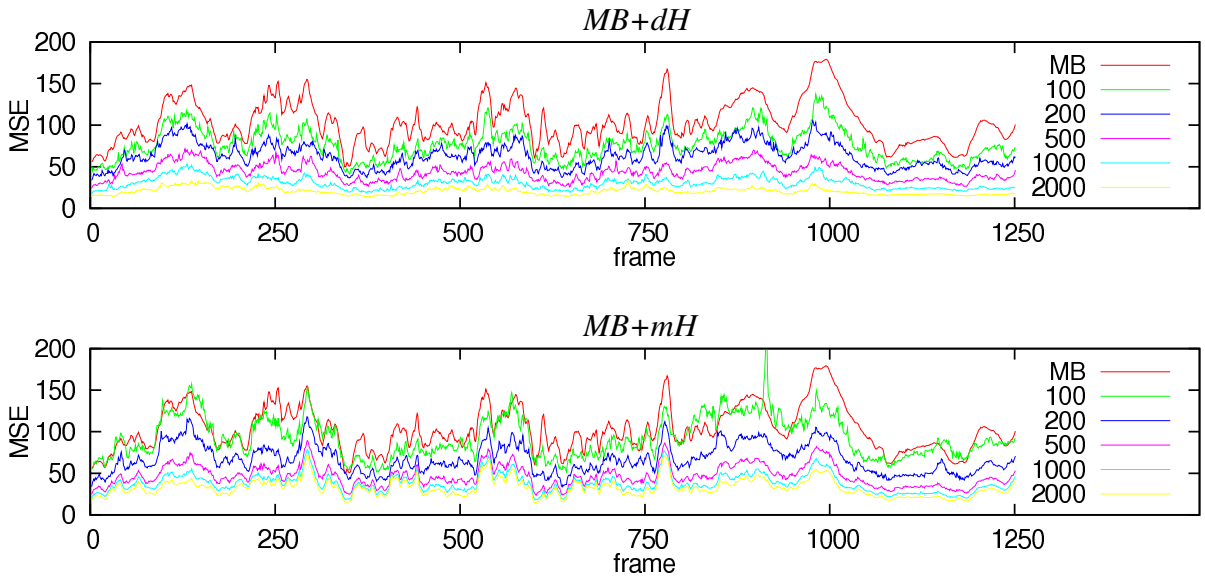


Figure 10.9.: Error per frame with our video codecs *MB+dH* and *MB+mH* with different numbers of additional points when compressing the sequence S4 from the HumanEva-II benchmark. The results with the codec *MB* are shown as comparison.

As expected, the more additional point information is used, the more accurate the reconstruction. This is true both visually and with respect to the MSE, which noticeably decreases even when using only a few additional particles. However, one can also see bleeding artifacts with the codec *MB+dH* when the first estimation obtained by projecting the object model is inaccurate, see the right shoe in Figure 10.7. Furthermore, the foreground is quite noisy with the codec *MB+mH* when only few particles are used.

Additionally, we encoded the sequence "Cart" (see Figure 5.15). A comparison of the performance of our first codec *MB* against *MPEG-1* and *MPEG-4* is shown in Figure 10.10, and magnifications thereof in Figure 10.11. Note that this sequence is much more challenging than the first video for our codec due to several reasons: First of all, the background is very noisy, which deteriorates the results of our diffusion-based image compression approach. Furthermore, the object model used is often not able to represent the complex movement performed by the actor, e.g. due to muscle contractions or missing joint angles. Additionally, the lower side of the feet are visible in many frames. However, this part is not included in the object model. Thus, the object is partly visible from the inside, which results in wrong colours. Finally, this sequence is much shorter than the HumanEva-II sequence S4, which means that the overhead necessary to store the object model is much larger. Especially due to the last reason, the results of our algorithm are, compared to *MPEG-1* and *MPEG-4*, much worse for this sequence than for the sequence S4 of the HumanEva-II benchmark

Due to these reasons, the reconstruction error of our approach is worse than that of *MPEG-4*, see the graph in Figure 10.10. However, note that the file created with our approach is also significantly smaller (see Table 10.1), and that we still beat *MPEG-1* in most frames.

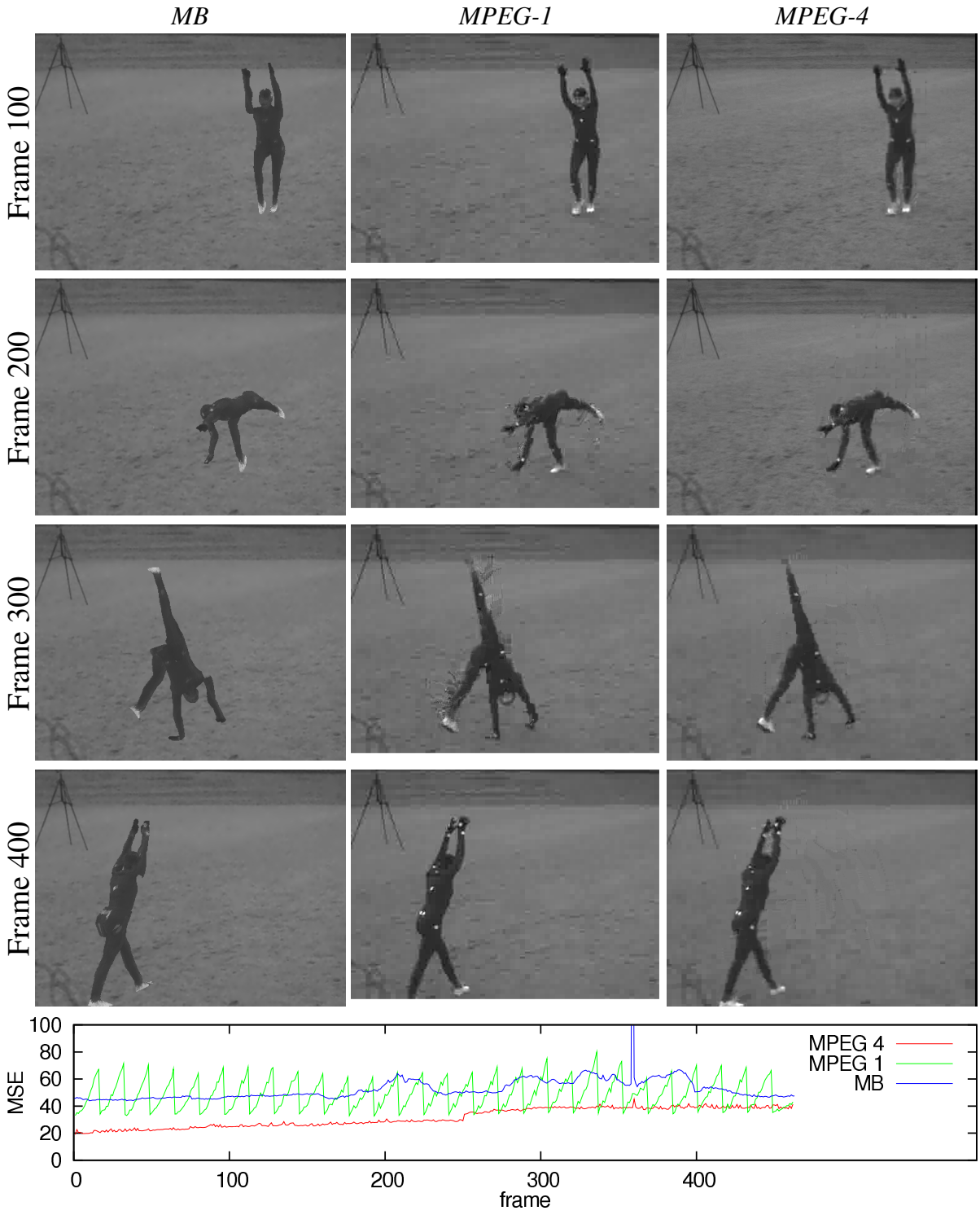


Figure 10.10.: Comparison of our method *MB* against *MPEG-1* and *MPEG-4* using the sequence "Cart" depicted in Figure 5.15. Shown are the frames 100, 200, 300, and 400, and a graph showing the MSE in each frame. The reason for the peak in frame 360 when using the codec *MB* is explained in Figure 10.12, the jump at frame 250 when using *MPEG-4* in Figure 10.13.

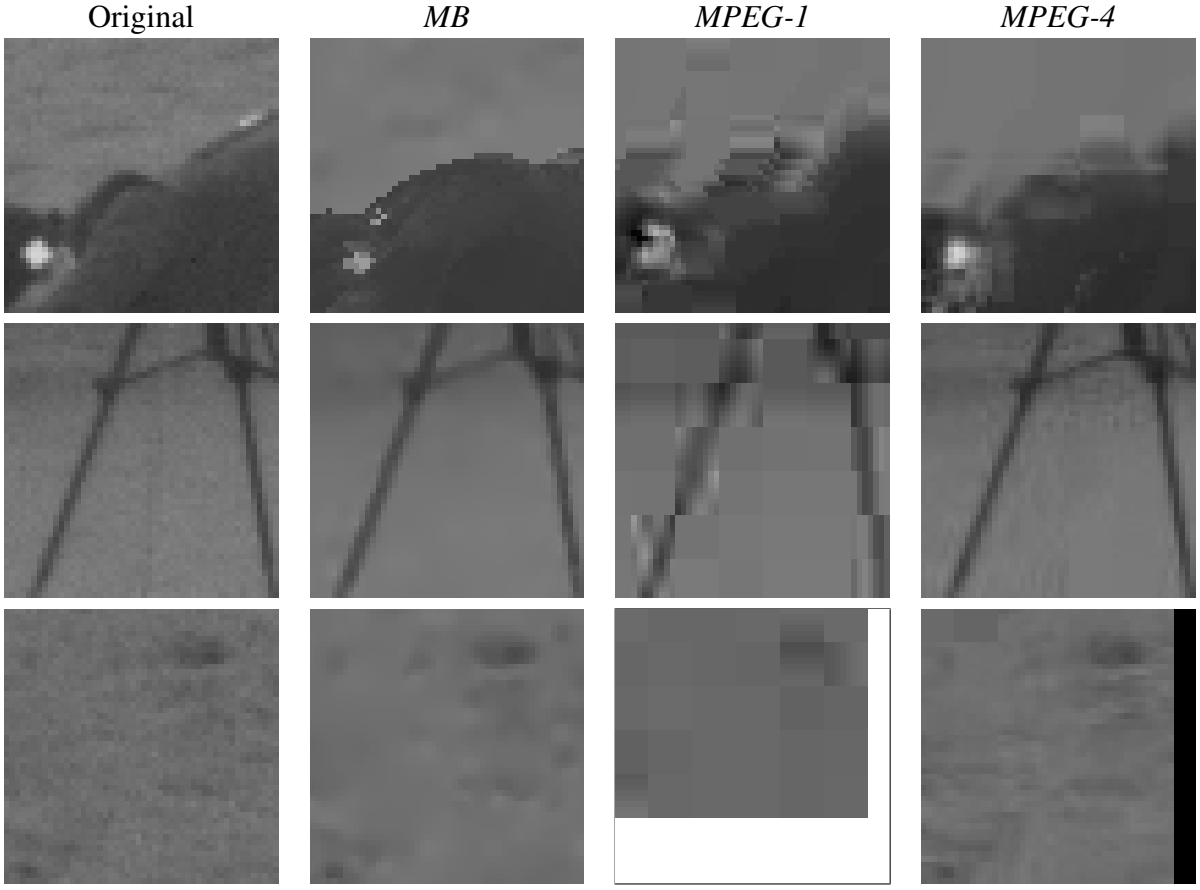


Figure 10.11.: Magnifications from frame 200 of the experiment shown in Figure 10.10. Again, one can easily see the block artifacts when using a codec of the *MPEG* family (third and fourth image in first row). These artifacts are also well visible in the background, as shown in the second row. The last row shows the lower right corner of each image. Since the encoder of *MPEG-1* trimmed the video from 500×380 to 496×368, this video is smaller, as indicated by the white region. The encoder of *MPEG-4* replaced four columns of pixels at the right hand boundary with black pixels.

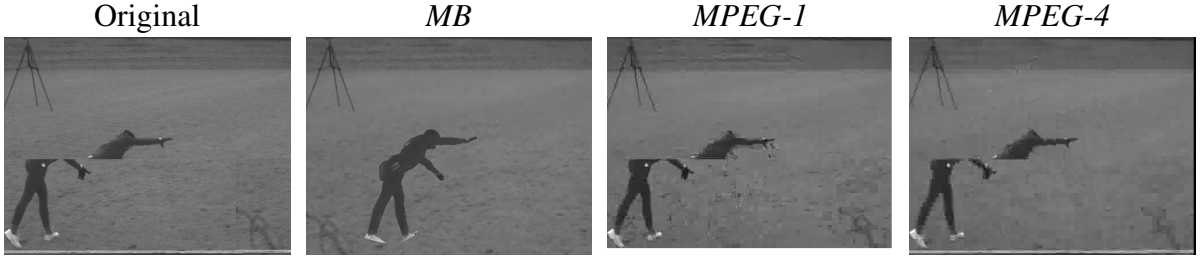


Figure 10.12.: Result in the corrupted frame 360 of the sequence “Cart”. While our codec *MB* ignored the corruption, *MPEG-1* and *MPEG-4* encoded the original frame.

Figure 10.13.: Magnifications of videos created with *MPEG-4*. **Left**: Sequence S4 of the HumanEva-II benchmark. **Right**: Sequence "Cart". For each sequence, one image before frame 250 (left in each block) and one image after frame 250 (right in each block) is shown.

The peak that can be observed with our codec in frame 360 is due to the fact that the input image was corrupted in that frame, see Figure 10.12. While *MPEG-1* and *MPEG-4* encode the corrupted input frame, the codec *MB* stores a "corrected" version. This can be seen as advantage of our algorithm, since it automatically corrected the corrupted frame. However, it is also possible to argue that our algorithm failed to encode this image properly. This is especially true if the background actually changes.

Note that it is well visible that the error with *MPEG-4* suddenly rises in frame 250. This is due to the fact that a new I-frame with lower quality than the first image was inserted by mencoder, as illustrated in Figure 10.13. The low quality is probably due to the fact that a variable bit rate was requested, as this allows the codec to use more bits in complex images. However, as the coder was unable to achieve the requested bit rate, it chose to use less bits for the new I-frame. In the HumanEva-II sequence, this also occurred as illustrated in the same figure. When using *MPEG-1*, it is apparent from the sawtooth pattern that the quality of the I-frames is better than those of the P- and B-frames.

Results for our other codecs are shown in Figure 10.14 and 10.15, respectively. As expected, more particles lead to more precise reconstructions. Furthermore, one can clearly observe that using too many particles should be avoided: The total reconstruction quality can never be much better than the quality of the background image, as only the foreground region is enhanced through the particles. Consequently, the appropriate fraction of the total space available should be used to store the background.

We also evaluated the effects of using different quantisation parameters than the standard setting $q = 17$ used in all experiments so far. Therefore, we again consider the sequence "Cart". Figure 10.16 illustrates the resulting MSEs obtained with different number of particles for our codecs *MB+dH* (left) and *MB+mH* (right). For both codecs, the same behaviour can be seen: While allowing too few quantised values noticeable degrades the result, the performance of our algorithms is very similar for a large range of different quantisation parameters. This justifies using the constant value $q = 17$ even though the results with the codec *MB+dH* are slightly better for this sequence when using $q = 33$.

Since the experiments in Section 3.2.1 showed that EED is better suited for 2-D image inpainting than homogeneous diffusion, one might wonder why we have not used EED for our video compression algorithms. In fact, there are several reasons for this decision. First of all,

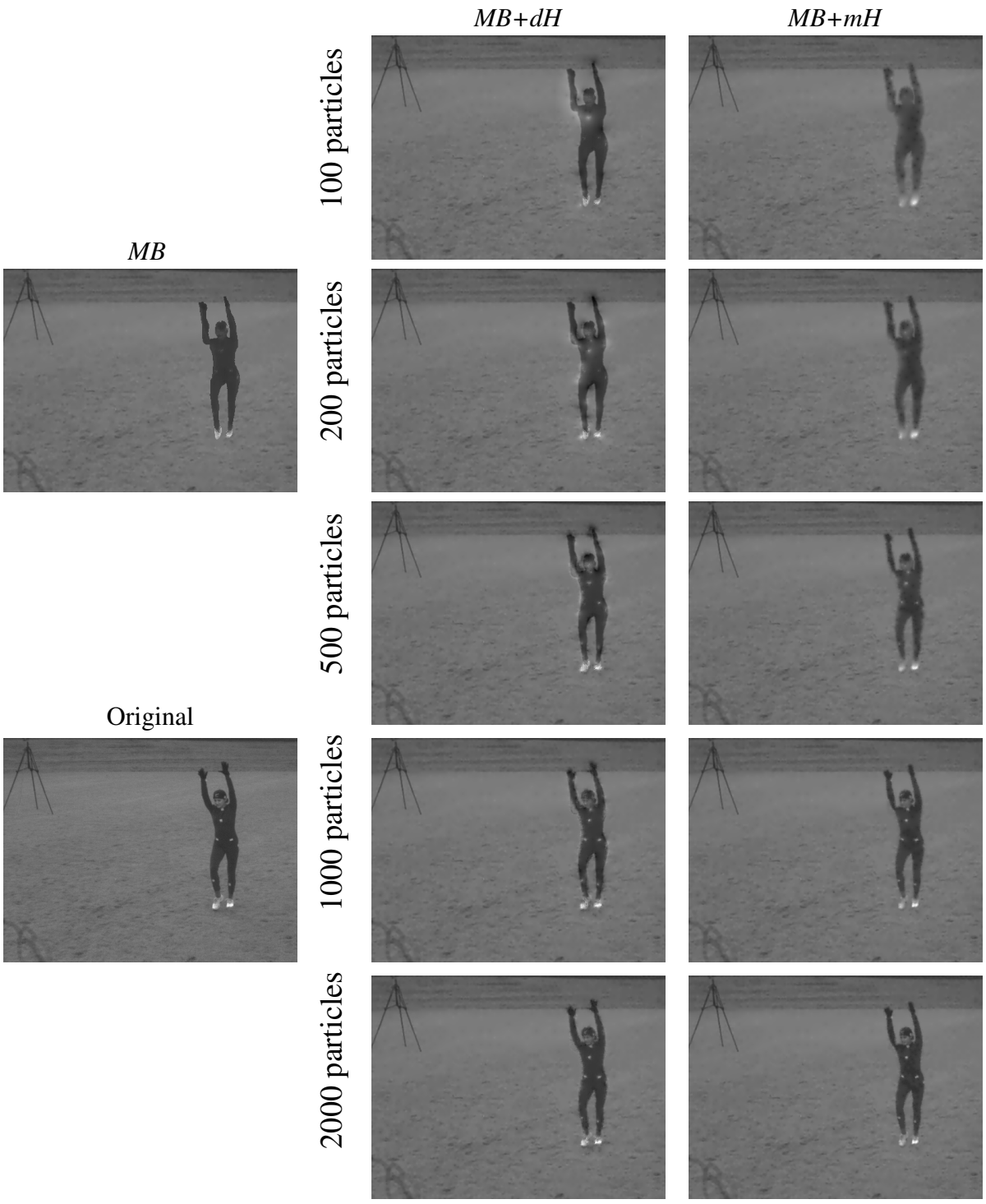


Figure 10.14.: Example results when compressing the sequence “Cart” with our codecs *MB+dH* and *MB+mH*. Shown are the images in frame 500 of the resulting videos with 100, 200, 500, 1000, and 2000 additional points. The original image and the result with our codec *MB* are shown in the left column.

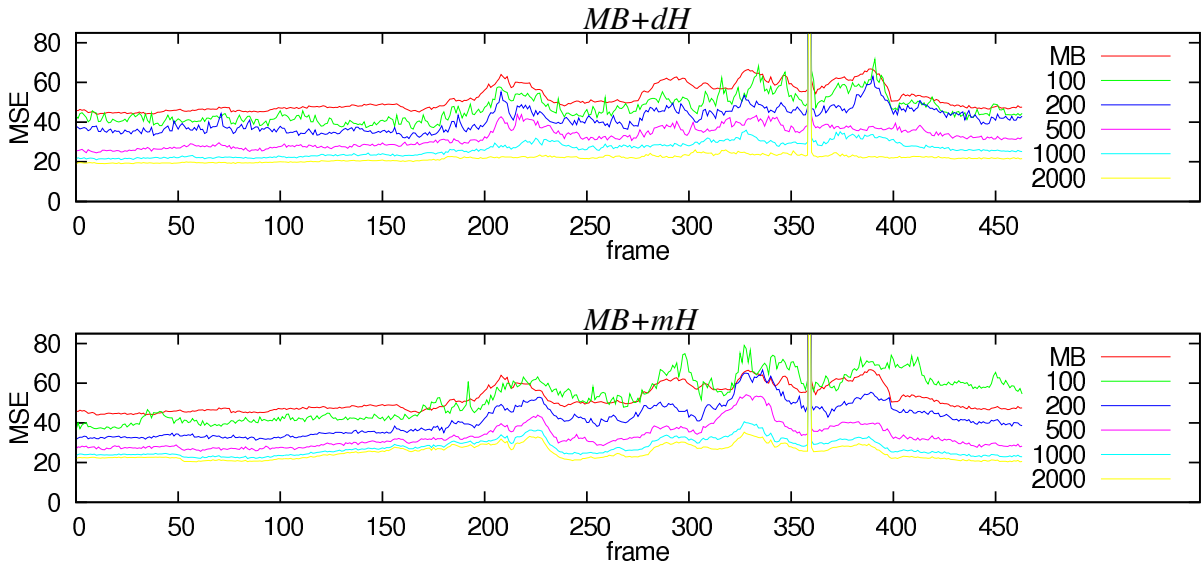


Figure 10.15.: Error per frame with our video codecs *MB+dH* and *MB+mH* with different numbers of additional points when saving the sequence “Cart”. The results with the codec *MB* are shown as comparison.

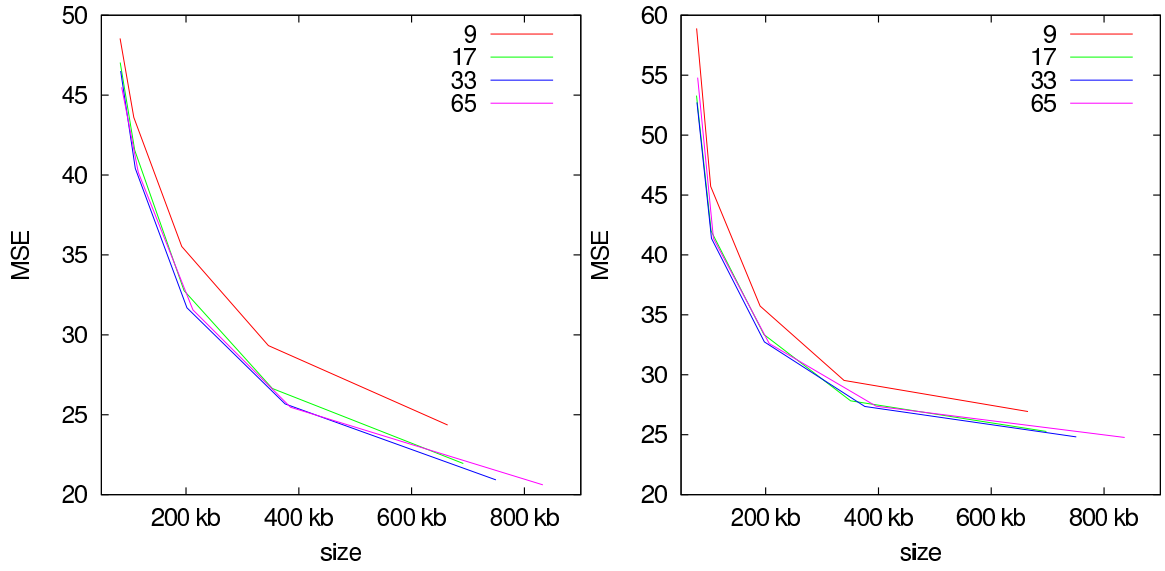


Figure 10.16.: MSE obtained when using different quantisation parameters q while compressing the sequence “Cart” with different number of particles. Results for the codec *MB+dH* are shown on the left, results for the codec *MB+mH* on the right.

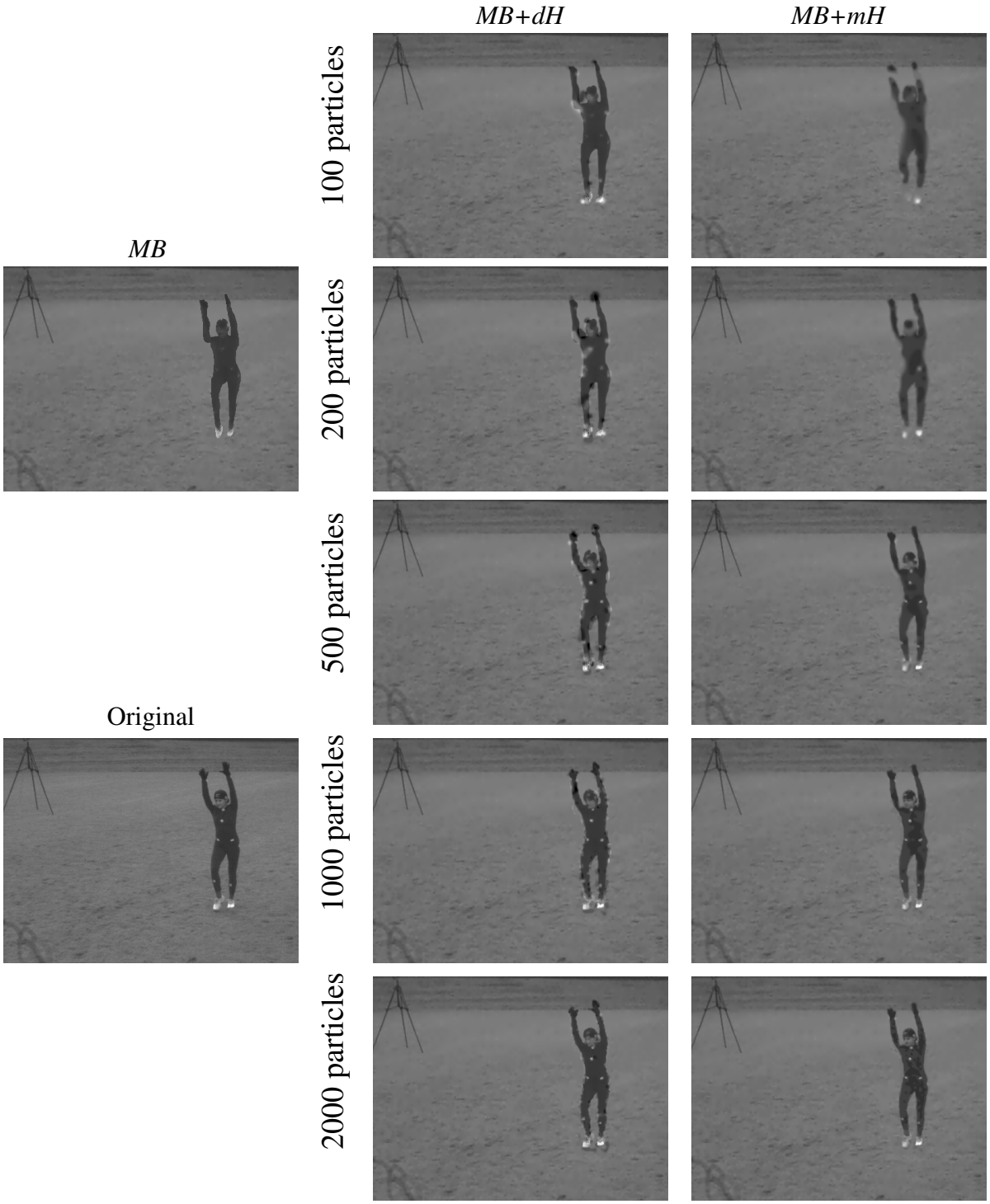


Figure 10.17.: Example results when compressing the sequence “Cart” with our codecs *MB+dH* (middle column) and *MB+mH* (right column) with EED instead of homogeneous diffusion. Shown are the images in frame 100 of the resulting videos with 100, 200, 500, 1000, and 2000 additional points. The original image and the result with our codec *MB* are shown in the left column, results for *MPEG-1* and *MPEG-4* in Figure 10.10.

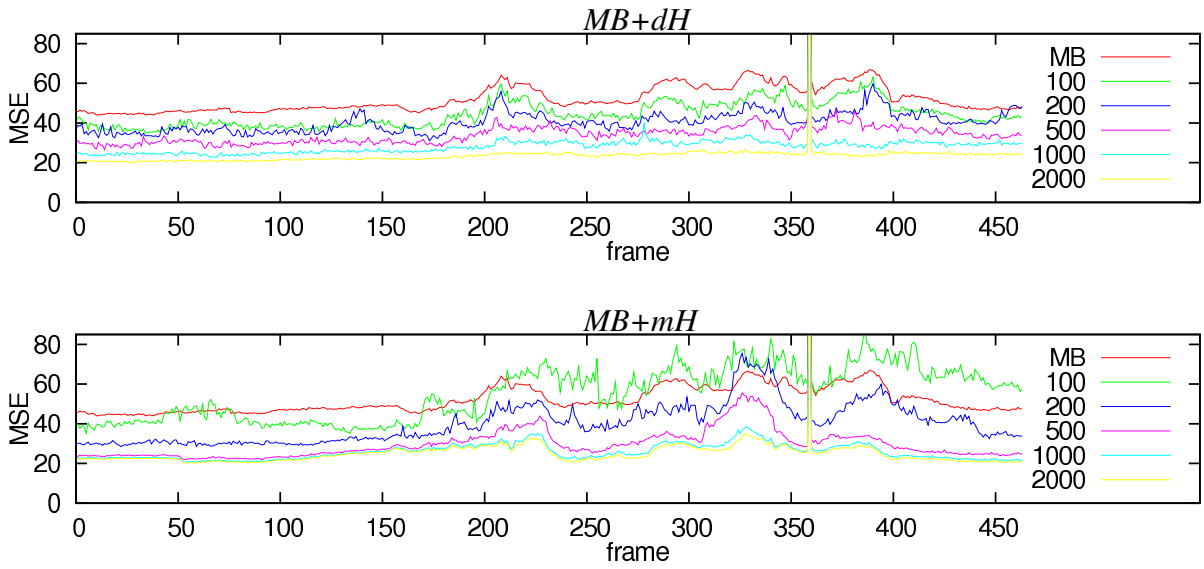


Figure 10.18.: Error per frame with our video codecs *MB+dH* and *MB+mH* when using EED instead of homogeneous diffusion. Shown are results with different numbers of additional points when saving the sequence "Cart". The results with the codec *MB* are shown as comparison.

it is known how to chose the inpainting points for homogeneous diffusion, namely according to the magnitude of the Laplacian of the smoothed image [20]. For EED, however, there is no such result. Additionally, the resulting files with EED were 1.6% larger on average, as the final entropy coding scheme was less efficient. Finally, inpainting with EED is slower than inpainting with homogeneous diffusion. Thus, we decided to use the computationally more favourable homogeneous diffusion.

To justify this decision, we also evaluate variants of our codecs which use EED as inpainting operator in the "Cart" sequence, see Figures 10.17 and 10.18. Thereby, we still use same approach to find the inpainting mask as before, since this is also the best known approximation when using EED. According to our tests, these variants were not consistently better than those with homogeneous diffusion, see Table 10.3. However, this might also be due to the fact that we used the constant parameter set $\sigma = 0.8$, $\lambda = 0.1$ for our experiments. It is possible that better results are obtained when optimising them. This is part of our future research.

Table 10.4 shows a final numerical comparison of our algorithms against *MPEG-4*. Here, we have chosen the number of particles in such a way that the file size with our algorithms is only slightly smaller than the one when using *MPEG-4*. For the HumanEva-II sequence S4, this results in 400 additional particles, while 200 additional particles are used for the "Cart" sequence. One can see that we clearly outperform *MPEG-4* in the HumanEva-II sequence: Even though the video created with *MPEG-4* is 8% larger, its MSE about 4.4 times as large as those of the videos created with our approaches. For the cart sequence, however, we are slightly worse than *MPEG-4* (due to the difficulties explained above).

Codec	Inpainting method	Particles	Size	MSE
MB+dH	homogeneous diffusion	100	83355	47.04
		200	109137	41.55
		500	196465	32.77
		1000	353026	26.64
		2000	691823	21.95
MB+dH	EED	100	84060	44.62
		200	108359	40.70
		500	199590	34.20
		1000	360625	28.23
		2000	713183	23.53
MB+mH	homogeneous diffusion	100	77471	53.30
		200	103362	42.00
		500	197006	33.36
		1000	350542	27.83
		2000	697960	25.27
MB+mH	EED	100	77257	55.38
		200	103620	40.45
		500	196707	29.88
		1000	362601	26.10
		2000	737561	25.08

Table 10.3.: File sizes and average MSE when compressing the sequence "Cart" with the codecs *MB+dH* and *MB+mH* with different number of particles. Both results with homogeneous diffusion as well as with EED are given.

	HumanEva-II S4		Cart	
method	file size	MSE	file size	MSE
Proposed	494 kb	48.09	110 kb	40.41
Proposed (with EED)	496 kb	47.66	104 kb	40.45
MPEG-4	537 kb	210.58	112 kb	31.52

Table 10.4.: File size and MSE of our best codecs and *MPEG-4* with comparable file sizes.

10.3. Summary

"People mistakenly assume that their thinking is done by their head;
it is actually done by the heart which first dictates the conclusion,
then commands the head to provide the reasoning that will defend it."

Anthony de Mello (1931 – 1987)

The video compression algorithms presented in this chapter show promising results that can beat those of *MPEG-1*, and even of *MPEG-4*. The ideas used are fundamentally different from standard video compression algorithms, and thus have different advantages and drawbacks. For example, our approaches are currently only applicable if one or several object models and a projection matrix are known. Furthermore, the sequence must have a fairly static background. Examples for such situations include traffic or security surveillance, or video conferencing. Note that generating intermediate frames not saved is straightforward with our algorithm: All that has to be done is to interpolate the object pose (and the particle locations and colours, if present) between two frames. If the same object is used in several sequences, or if several views from the same sequence should be compressed, the space required by our algorithm can be further reduced. Also note that the video quality should be significantly improved if coloured models of the foreground objects are already given.

11 Summary and Conclusion

"Science may set limits to knowledge, but should not set limits to imagination."

Bertrand Russell (1872 - 1970)

This thesis discussed the challenging topics of image compression, 3-D pose tracking, and halftoning, and illustrated that this apparently unrelated areas can be combined to video compression algorithms.

First, we developed a compression algorithm for still images. Our novel image compression codec is explained in Chapter 3. It is based on the idea to store only few single points, while the remainder of the image is reconstructed by inpainting using partial differential equations. The points are located on an adaptive rectangular grid to reduce the overhead necessary to store the point locations. As we have shown, our algorithm can outperform standard compression algorithms such as JPEG 2000 in images with a low or moderate amount of texture, especially for high compression ratios.

In Chapters 4 to 8, we have developed our 3-D pose tracker. We started by introducing related work in Chapter 4. In particular, we discussed the tracking approach of Brox, Rosenhahn, and Weickert, as our basic tracking approach is built upon this work.

The first version of our tracker is introduced in Chapter 5. In contrast to the approach from Chapter 4, our tracking algorithm directly optimises the pose without using segmentation as an intermediate step. Furthermore, Chapter 5 also introduces some optional extensions such as handling of kinematic chains, reusing image information from the previous frames, or using prior knowledge about the scene or motion patterns.

In Chapter 6, we have extended our tracker such that it can perform a joint tracking of multiple objects, even in case of multiple mutual occlusions. We have shown that, even if only one object is of interest, tracking additional objects passing before the object of interest can stabilise the tracking.

Chapter 7 introduces the necessary improvements to deal with self-occlusions, which are very common when tracking articulated objects such as humans. The idea to use additional internal silhouettes obtained by splitting the object model is a consequent refinement of the ideas presented for tracking multiple objects.

As further enhancement, we proposed to split not only the object, but also the background region. The resulting localised mixture model is explained in detail in Chapter 8. The resulting region model works both for known and unknown background images, and for static or variable background images. In all cases, tracking was further improved by this new model. To the best of our knowledge, our tracking algorithm shows the second best results in the literature for the HumanEVA-II benchmark. The only algorithm that yields better results requires five times as much time and additional background images.

As a final step, we have introduced our novel dithering and stippling algorithms in Chapter 9. These models are based on electrostatic interactions, and are thus intuitive and flexible. Using a detailed evaluation with different quality criteria, we have shown that our algorithms approximate images or PDFs with a much higher quality than any other state-of-the-art dithering or stippling algorithm. We further proposed several extensions that allow to adapt our halftoning algorithms to specific tasks.

Finally, we introduced several video compression algorithms optimised for scenes with static background and few moving foreground objects. They are well suited for compressing video conferencing or surveillance videos. We have shown that the proposed codec can yield better results than the standard video compression algorithms *MPEG-1* and *MPEG-4* in some situations. This was possible by combining all the approaches introduced in the previous chapters.

To summarise, this thesis has introduced novel algorithms for different topics. These topics are halftoning, tracking, as well as image and video compression. In each of these topics, our approaches yield excellent results that can outperform other state-of-the-art algorithms.

11.1. Future Research Directions

"Paradise is exactly like where you are right now...
only much, much better."

Laura Phillips "Laurie" Anderson (*1947)
in *Language Is A Virus*

Although the presented algorithms yield excellent results, there are many open questions that should be addressed in the future. In the remainder of this chapter, we will discuss some possible future research directions.

First of all, the tracking algorithm is not perfect and may fail in some situations. Thus, one possible future work is to find a way to detect tracking failures and, if possible, to recover from them. For example, it might be possible to deduce that tracking has failed if the appearance of the object drastically changes. In such cases, searching for a new initial pose can stabilise the tracking.

Another issue is that we currently assume that an (approximate) model for each moving object is known. Even if only persons are moving, the models necessary to track them can look quite different. Thus, another possible improvement is to use only a simple template model, which is adapted during tracking. If different objects shall be tracked, it is beneficial to automatically chose the right model for each moving object. Alternatively, one can also try to create the object models directly from the image data.

In Section 5.4, we have explained several methods how to include prior knowledge about the scene and the objects. A promising idea is to further follow this idea and include more knowledge into our tracking algorithm. For example, it can happen that the estimated positions of two or more objects intersect each other. The same is true for two parts of a single kinematic

chain. In reality, however, this is impossible. Including this knowledge into our tracking algorithm should therefore yield improved results.

Furthermore, our tracking approach has some free parameters which must be searched for each new sequence. Mainly, this is the length l of the force vectors, and the threshold T (see Section 5.2). One direction of future research is therefore to find a good heuristic for these parameters, e.g. by using the distance of the objects from each camera. This would eliminate the need to find these parameters by hand.

Similarly, there are a number of parameters in our image compression algorithm, such as the number of possible quantisation levels, for which good values must be found for each new image and each new compression ratio. Currently, this is done using exhaustive search, binary search, or Fibonacci search. However, a good heuristic could significantly reduce the compression time. This is especially true for the optimisation of the brightness values explained in Section 3.1.5.

In some hatching applications, one might allow lines to bend. While this is not possible with our current implementation, it should not be very difficult to extend our approach to this setting. For example, one might replace the constraint that all points are on one line by a force that only favours this.

It is well known that the simple RGB colour model is not optimal for compression, since there are far more advanced possibilities to treat colour images than the simple methods used in this thesis. For example, it is often advantageous to convert images such that they use a different colour model, e.g. the *YCbCr* colour space used by the lossy versions of JPEG and JPEG 2000 [259]. This reduces the amount of correlation between the channels, and thus increases the compression ratio of the entropy coder used. Additionally, it is often proposed to subsample the colour channels, or to quantise them coarser, since the human visual system is less sensitive to colour differences than to brightness differences. In our image compression framework, it may also be pay off to use different masks for different channels such that less information about the colour channels is given. All this is part of our ongoing research.

One important part upon which our video compression algorithm is built is the ability to track objects moving in the video to be compressed. However, to make this approach applicable for compressing general video sequences, it is necessary to detect new objects entering the field of vision, and to find initial poses for all objects. In the literature, there are several algorithms for these tasks which can be included into our framework. However, we have not implemented them, since no new scientific insights are to be expected.

Obviously, the quality of our video compression algorithms crucially depends on the model used for tracking. If the model is sampled too fine, saving the model requires too much space. If it is sampled too coarse, our algorithm *MB* approximates the foreground region poorly. Thus, our approach either yields a high error, or many additional particles are needed to improve the first estimation of the foreground region. Thus, it might pay off to investigate the effect of mesh refinement or mesh simplification. It is also possible to compress the object models with diffusion approaches, as done in [15], or to use a (compressed) texture instead of saving the colour of each vertex.

As we have elaborated, there are many possibilities to improve and adapt our flexible approaches to future requirements.

A Axiomatic Introduction to Statistics

"In these matters the only certainty is that nothing is certain."

Pliny the Elder (23 – 79)

In this section, we introduce the necessary statistics to axiomatically introduce probability density functions, in contrast to the intuitive introduction given in Section 2.7. We will skip those definitions and theorems that are not required in this thesis. More complete introductions can be found in [130, 145, 257].

We start this section with two basic definitions:

Definition A.A: (σ-algebra) A *σ-algebra* $\mathcal{A}$ is a family of subsets of a non-empty set Ω which fulfils the following properties:

- $\Omega \in \mathcal{A}$
- $A \in \mathcal{A} \Rightarrow \Omega \setminus A \in \mathcal{A}$
- $(\forall i \in \mathbb{N} : A_i \in \mathcal{A}) \Rightarrow \bigcup_{i=1}^{\infty} A_i \in \mathcal{A}$

Definition A.B: (Measurable space) The pair $(\Omega, \mathcal{A})$, with a set Ω and a σ-algebra $\mathcal{A}$, as defined in Definition A.A, is called *measurable space*.

The most prominent example for a σ-algebra over $\mathbb{R}$, which will also be used in this thesis, is the *Borel σ-algebra* $\mathcal{B}$. By definition, this algebra is the smallest σ-algebra containing all open sets. Is it easy to show that the Borel σ-algebra also contains all half-open and closed sets, and the union of all such sets. In fact, it is not trivial[1] to write down a set that is not contained in the Borel σ-algebra.

Next, we introduce the notion of *probability distributions*, which is the core definition of probability theory:

Definition A.C: (Probability distribution) A function $P : \mathcal{A} \to [0, 1]$ defined over a σ-algebra $\mathcal{A}$ is called *probability distribution* (or *probability measure*) if the following properties hold:

- $P(\emptyset) = 0$
- $P(\Omega) = 1$

[1] The axiom of choice is required

- P is *σ-additive*. That is, if the sets A_i are disjoint, the property

$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i) \tag{A.1}$$

holds.

Probability distributions are rarely used alone, but are used as a part of a larger entity. This larger entity is given by the following definition:

Definition A.D: (Probability space) The triple $(\Omega, \mathcal{A}, P)$ is called *probability space* if the following properties hold:

- $\mathcal{A}$ is a σ-algebra of subsets of Ω
- P is a probability distribution over $\mathcal{A}$

The elements of $\mathcal{A}$ are called *events*. As a simple example, consider the possible outcomes of rolling a fair die. In this case $\mathcal{A} = \{1, 2, 3, 4, 5, 6\}$, and every of these numbers is one possible event.

Intuitively, the probability distribution P assigns a probability to each event. In case of the fair die roll, the corresponding probability distribution is $P(\{i\}) = \frac{1}{6}$, with $i \in \{1, 2, 3, 4, 5, 6\}$.

Often, one is not interested in the random event itself, but on some quantity derived from it. If two dice are thrown, for example, one might only be interested in the sum of both dice instead of the numbers on each die. In such a situation, *random variables* are used. To introduce random variables, we have to first define *measurable functions*, though:

Definition A.E: (Measurable function) A *measurable function* is a function $f : (\Omega, \mathcal{A}) \to (\Omega', \mathcal{A}')$ with

- measurable spaces $(\Omega, \mathcal{A})$ and $(\Omega', \mathcal{A}')$
- $f^{-1}(A') \in \mathcal{A} \quad$ for all $A' \in \mathcal{A}'$.

Definition A.F: (Random variable) Let $(\Omega, \mathcal{A}, P)$ be a probability space and $(\Omega', \mathcal{A}')$ a measurable space. Then, a *random variable* is a measurable function $X : (\Omega, \mathcal{A}) \to (\Omega', \mathcal{A}')$.

Here, we will only consider random variables with domain $\mathbb{R}^n$, where n is an integer larger than zero.

As explained in Section 2.7, a *random variable* can be visualised as numeric result of a non-deterministic process. To compute how likely the different outcomes of the random variable are, one uses either its *distribution* or its *cumulative distribution function* (or *CDF*). These are defined in the following way:

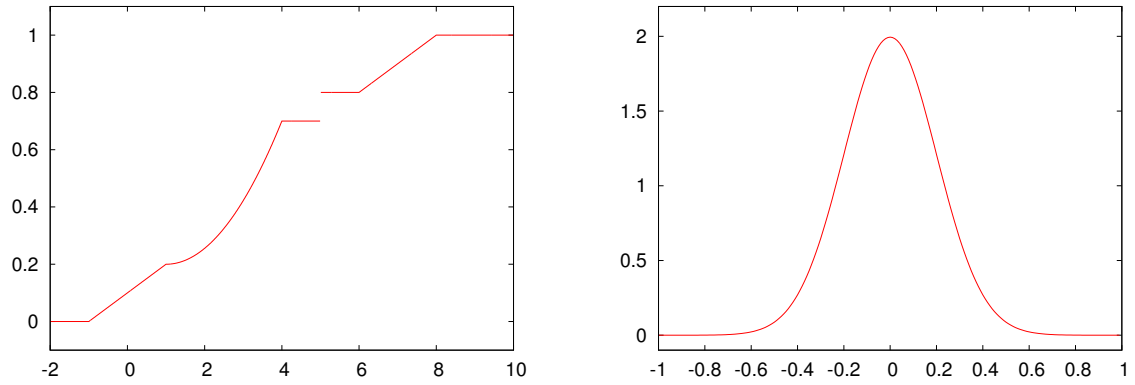


Figure A.1.: Examples of a CDF (left) and a PDF (right). As can be seen, a CDF is monotonically increasing and has the limits zero and one. A PDF, however, must only be non-negative, but can be arbitrarily high as long as the integral over it is one. There can be jumps and constant sections in a CDF and in a PDF.

Definition A.G: (Distribution) The *distribution* of a random variable X on the probability space $(\mathbb{R}, \mathcal{B}, P)$ is given by

$$\begin{aligned} P_X(B) &:= P(X^{-1}(B)) && \text{(A.2)} \\ &= P(\{\omega : X(\omega) \in B\}) && \text{(A.3)} \\ &= P(X \in B) \qquad \text{for all } B \in X(\Omega) \cap \mathcal{B}. && \text{(A.4)} \end{aligned}$$

Definition A.H: (Cumulative distribution function) Let X be a random variable on the probability space $(\mathbb{R}, \mathcal{B}, P)$. The *cumulative distribution function* (or CDF) F of X is given by

$$F(x) := P(X \leq x) = P(X^{-1}(]-\infty, x])) = P_X(]-\infty, x]). \tag{A.5}$$

In order to understand how a cumulative distribution function looks like, we consider the following properties:

Theorem A.I: (Properties of cumulative distribution functions) Let F be the CDF of a random variable on $(\mathbb{R}, \mathcal{B}, P)$. Then, we have:

- F is non-decreasing, i.e. $x > y \Rightarrow F(x) \geq F(y)$ for all $x, y \in \mathbb{R}$
- F is right continuous, i.e. $\lim_{x \to y+} F(x) = \lim_{x \downarrow y} F(x) = F(y)$ for all $y \in \mathbb{R}$
- $\lim\limits_{x \to -\infty} F(x) = 0, \qquad \lim\limits_{x \to \infty} F(x) = 1$

Furthermore, every function which fulfils these three properties is a distribution for some random variable.

For some distributions, it is more convenient to define the distribution using a *probability density function*, or short *PDF*:

Definition A.J: (Probability density function) A function $f : \mathbb{R} \to \mathbb{R}$ is called a *probability density function* (or *PDF*) for the random variable X with distribution F if

- $f(x) \geq 0 \quad$ for all $x \in \mathbb{R}$
- $F(x) = \int_{-\infty}^{x} f(t)\mathrm{d}t \quad$ for all $x \in \mathbb{R}$
- F is continuous on $\mathbb{R}$

One of the advantages of defining a distribution by giving a PDF is that one can immediately see which outcomes are more likely, namely those regions where the PDF assumes large values. Moreover, the PDF is often much simpler than the distribution of a random variable. Figure A.1 shows examples for a CDF and a PDF.

Note that not every random variable has a probability density function, though. Specifically, all random variables with a non-continuous CDF, i.e. all random variables for which there exists a point with a non-zero probability, cannot be given as PDF. In fact, it is quite easy to show that the following properties hold:

Lemma A.K: (Properties of PDFs) For a probability distribution P given by the PDF f, the following properties hold:

- $\forall b \in \mathbb{R}: \quad P(\{b\}) = 0$
- $\forall a < b \in \mathbb{R} : P(]a, b]) = P([a, b]) = P(]a, b[) = P([a, b[) = \int_{a}^{b} f(t)\mathrm{d}t.$

When considering more than one random variable, e.g. the individual components of a multi-dimensional random variable, it is possible that some variables depend on each other. As a simple example, consider rolling a die, with the random variables A and B which consider the number rolled (A), and whether the die shows an even number (B), respectively. Thus, knowing the result of the random variable A automatically determines the outcome of the random variable B.

It is also possible that all outcomes are still possible, but the likelihood of some outcomes has changed, e.g. consider the random variables B and C, where C indicates whether the die shows a number smaller than four. In this case, the random variables are *dependent* on each other. If this is not the case, they are called *independent*. If random variables are independent, this means that knowing the outcome of one of the experiments does not yield any information about the outcome of other experiments. For example, when a fair die is rolled twice, the

result of the second roll is independent from the first. However, the sum of the two rolls is not independent from the result of the first roll.

Mathematically, independence is defined as follows:

Definition A.L: (Independence) A collection of events $A_1, \ldots, A_n \in \mathcal{A}$ is called *independent* if

$$P\left(\bigcap_{j \in J} A_j\right) = \prod_{j \in J} P(A_i) \tag{A.6}$$

holds for every set $J \subset \{A_1, \ldots, A_n\}$, and *mutually independent* if

$$P\left(\bigcap_{i=1}^{n} A_j\right) = \prod_{i=1}^{n} P(A_i) \tag{A.7}$$

holds. A collection of random variables $X_1, \ldots, X_n$ is called (mutually) *independent* if, for any numbers $a_1, \ldots, a_n \in \mathbb{R}$, the events $\{X_1 \leq a_1\}, \ldots, \{X_n \leq a_n\}$ are (mutually) independent.

If one considers a sequence of random variables, one can try to estimate the underlying PDF (see Section 2.7.2). In such a situation, one usually assumes that the random variables are *independent and identically distributed*, though:

Definition A.M: (Independent and identically distributed random variables) A sequence of random variables is *independent and identically distributed* (or short (*i.i.d.*)) if

- The random variables are mutually independent
- Each random variable has the same probability distribution

B Own Publications

"One of the symptoms of an approaching nervous breakdown is the belief that one's work is terribly important."

Bertrand Russell (1872 – 1970)
in *Conquest of Happiness*

Journal Papers

1. C. Schmaltz, P. Gwosdek, A. Bruhn, and J. Weickert. Electrostatic halftoning. *Computer Graphics Forum*, 29(8):2313–2327, December 2010.
   Revised version of Technical Report 260, Department of Mathematics, Saarland University, Feb. 2010.

2. C. Schmaltz, B. Rosenhahn, T. Brox, and J. Weickert. Region-based pose tracking with occlusions using 3D models. *Machine Vision and Applications*, 2011.
   DOI: 10.1007/s00138-010-0317-5. Online First.
   Revised version of Technical Report 277, Department of Mathematics, Saarland University, Oct. 2010.

3. P. Gwosdek, C. Schmaltz, J. Weickert, and T. Teuber. Fast electrostatic halftoning. *Journal of Real-Time Image Processing*, 2011.
   DOI: 10.1007/s11554-011-0236-3. Online First.
   Revised version of Technical Report 295, Department of Mathematics, Saarland University, June 2011.

4. T. Teuber, G. Steidl, P. Gwosdek, C. Schmaltz, and J. Weickert. Dithering by differences of convex functions. *SIAM Journal on Imaging Sciences*, 4(1):79–108, 2011.
   Revised version of Technical Report Pr-2010-01, Department of Mathematics, University of Mannheim, Mannheim, April 2010.

Conference Papers

5. C. Schmaltz, B. Rosenhahn, T. Brox, D. Cremers, J. Weickert, L. Wietzke, and G. Sommer. Region-based pose tracking. In J. Martí, J. M. Benedí, A. M. Mendonça, and J. Serrat, editors, *Pattern Recognition and Image Analysis*, volume 4478 of *Lecture Notes in Computer Science*, pages 56–63, June 2007. Springer.

6. C. Schmaltz, B. Rosenhahn, T. Brox, J. Weickert, D. Cremers, L. Wietzke, and G. Sommer. Occlusion modeling by tracking multiple objects. In F. Hambrecht, C. Schnörr, and B. Jähne, editors, *Pattern Recognition*, volume 4713 of *Lecture Notes in Computer Science*, pages 173–183, Berlin, September 2007. Springer.

7. C. Schmaltz, B. Rosenhahn, T. Brox, J. Weickert, L. Wietzke, and G. Sommer. Dealing with self-occlusion in region based motion capture by means of internal regions. In F. J. Perales and R. B. Fisher, editors, *Articulated Motion and Deformable Objects*, volume 5098 of *Lecture Notes in Computer Science*, pages 102–111, Berlin, July 2008. Springer.

8. C. Schmaltz, B. Rosenhahn, T. Brox, and J. Weickert. Localised mixture models in region-based tracking. In J. Denzler, G. Notni, and H. Süße, editors, *Pattern Recognition*, volume 5748 of *Lecture Notes in Computer Science*, pages 21–30, Berlin, 2009. Springer.

9. C. Schmaltz, J. Weickert, and A. Bruhn. Beating the quality of JPEG 2000 with anisotropic diffusion. In J. Denzler, G. Notni, and H. Süße, editors, *Pattern Recognition*, volume 5748 of *Lecture Notes in Computer Science*, pages 452–461, Berlin, 2009. Springer.

10. B. Rosenhahn, C. Schmaltz, T. Brox, J. Weickert, and H.-P. Seidel. Staying well grounded in markerless motion capture. In G. Rigoll, editor, *Pattern Recognition*, volume 5096 of *Lecture Notes in Computer Science*, pages 385–395, Berlin, June 2008. Springer.

11. B. Rosenhahn, C. Schmaltz, T. Brox, J. Weickert, D. Cremers, and H.-P. Seidel. Markerless motion capture of man-machine interaction. In *Proc. 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, June 2008. IEEE Computer Society Press.

12. D. Zang, L. Wietzke, C. Schmaltz, and G. Sommer. Dense optical flow estimation from the monogenic curvature tensor. In F. Sgallari, F. Murli, and N. Paragios, editors, *Scale Space and Variational Methods in Computer Vision*, volume 4485 of *Lecture Notes in Computer Science*, pages 239–250, Berlin, May/June 2007. Springer.

13. L. Wietzke, G. Sommer, C. Schmaltz, and J. Weickert. Analysis of the curvature tensor from the viewpoint of signal processing. In *Selected Papers from ICNAAM-2007 and ICCMSE-2007*, volume 1046 of *AIP Conference Proceedings*, pages 150–153, Corfu, Greece, September 2008.

14. L. Wietzke, G. Sommer, C. Schmaltz, and J. Weickert. Differential geometry of monogenic signal representations. In G. Sommer and R. Klette, editors, *Robot Vision*, volume 4931 of *Lecture Notes in Computer Science*, pages 454–465, February 2008. Springer.

15. L. Wietzke, G. Sommer, O. Fleischmann, and C. Schmaltz. The conformal monogenic signal of image sequences. In D. Cremers, B. Rosenhahn, A. L. Yuille, and F. R. Schmidt, editors, *Visual Motion Analysis*, volume 5604 of *Lecture Notes in Computer Science*, pages 305–322, Berlin, 2009. Springer.

Technical Reports

16. C. Schmaltz, P. Gwosdek, and J. Weickert. Multi-class electrostatic halftoning, 2012. Submitted to Computer Graphics Forum. Preliminary decision: Probably accept after minor revisions.
Revised version of Technical Report 301, Department of Mathematics, Saarland University, Oct. 2011.

17. C. Schmaltz, B. Rosenhahn, T. Brox, D. Cremers, J. Weickert, L. Wietzke, and G. Sommer. Region-based Pose Tracking. Technical Report 196, Department of Mathematics, Saarland University, July 2007.

Index

"One of the great defects of English books printed in the last century is the want of an index."

Lafcadio Hearn (1850 – 1904)

Glossary

"All abstract sciences are nothing but the study of relations between signs."

Denis Diderot (1713 – 1784)

Note that some signs have multiple meanings.

Notation	Description
$\mathcal{A}$	σ-algebra. 223
a	Free parameter in image compression used to compute the threshold T. 34
$\mathcal{B}$	Borel σ-algebra of $\mathbb{R}$. 223
C	Desired compression ratio for our image compression algorithm. 40 *or*: Centre of a camera. 23 *or*: Number of cameras in tracking. 70 *or*: Tradeoff parameter in multi-size halftoning. 156
c	Principal point of a camera. 23, 24
$c_{\boldsymbol{\chi}}$	Indicator function of the points occupied when projecting a 3-D model with pose χ into an image. 78
c_0	Speed of light in vacuum. 12 *or*: Indicator function of the background when tracking with multiple regions. 111
c_i	Image point corresponding to a vertex C_i in a 2-D–3-D point correspondence (c_i, C_i). 62
C_i	Vertex corresponding to an image point c_i in a 2-D–3-D point correspondence (c_i, C_i). 62
$c_{\chi_i,i}$	Indicator function of the i-th object with pose χ_i. 108
$c^j_{\chi_i,i}$	Indicator function of the j-th part of the i-th object with pose χ_i. 121
c_K	Indicator function of the point set K. 6
c_r	Circular curve with radius r. 12
D	Normal vector of a place in Hessian form. 22, 90
d	Current recursion depth while creating adaptive rectangular subdivision. 34 *or*: Signed distance of a plane in Hessian form from the origin. 22, 90 *or*: Standard Euclidean metric of a vector field. 109
$\boldsymbol{E}$	Electric field. 12
E	Any energy function. 67
$\boldsymbol{e}_{n,\boldsymbol{x}}$	Unit vector from $\boldsymbol{p}_n$ to the grid location $\boldsymbol{x} \in \Gamma$. 152
$\boldsymbol{e}_{i,j}$	Unit vector from the i-th particle to the j-th particle, see Equation (2.22). 12, 151
f	Input image (continuous). 5, 27 *or*: Focal length of a camera. 23
F	Symbol for "farad", the unit of capacitance. 12 *or*: Foreground region in our video compression codec *MB+mH*. 202
$\boldsymbol{F}$	Any kind of force. 12

Notation	Description
$\boldsymbol{F}_n$	Total force acting on the particle $n \in \mathcal{P}$. 152
$\boldsymbol{F}_{a,n}$	Attractive force acting on n-th particle due to the underlying image. 151
$\boldsymbol{F}_{r,n,m}$	Repulsive force acting on n-th particle due to the m-th particle. 13, 151
$\boldsymbol{F}_{r,n}$	Repulsive force acting on n-th particle due all other particles. 151
$\boldsymbol{F}_{\boldsymbol{g},n}$	Force between particle n and the closest grid point $\boldsymbol{g}$. 158, 240
f_g	Principal frequency. 174
G	Any group, or Lie group. 14, 15
g	Diffusivity function. 8
$GL(n, \mathbb{R})$	The group of invertible $n \times n$ matrices over $\mathbb{R}$. 15
H	(Regularised) Heaviside function. 67 *or*: Unit of inductance. 12
h	Bandwidth of a kernel. 28
I	Identity matrix (of matching dimension). 16, 63
i	Interaction matrix for halftoning with different sizes/colours. 157
j	Strength of the turbulence field T. 161
K	Inpainting mask used for image compression. 6, 35 *or*: Kernel. 27
k	Constant equal to twice Coulomb's constant, see Equation (2.28). 13
K_σ	Gaussian with standard deviation σ. 7
L	Differential operator. 6 *or*: A line in Plücker form. 20 *or*: Segmentation of an image. 133
l	Length of the force vectors used for tracking. 81 *or*: Number of parts an object model consists of when using additional internal silhouettes. 120 *or*: Free parameter in image compression used to compute the threshold T. 34
L_i	A line in Plücker form passing through the optical centre of the camera and the ith image-vertex point correspondence (c_i, C_i). 63
l_i	Number of parts the i-th object model consists of when using additional internal silhouettes. $l_0 = 1$ correspond to the background region. 121
M	Maximal depth allowed for the tree which is used to store the inpainting mask K in our image compression algorithm. 35 *or*: Number of feature channels in an image. 66 *or*: Region between fore- and background region in the video compression codec *MB+mH*. 202
m	Minimal depth allowed for the tree which is used to store the inpainting mask K in our image compression algorithm. 35 *or*: Momentum of a Plücker line. 20
M_i^j	j-th part of the i-th multi-component object model. 121
M^j	j-th part of an multi-component object model. 120
N	Number of objects when tracking multiple objects. 107 *or*: Number of prior joint configurations when using angle priors. 87

Notation	Description
n	Number of joint angles in a kinematic chain. 19 *or*: Normalised vector pointing in the direction of a line in Plücker form. 20
n_x	Width of a discrete image. 151
n_y	Height of a discrete image. 151
O_i^j	Set of points of the j-th part of the i-th object with pose χ_i which are projected to the image point x. 121
O^j	Set of points of the j-th internal part of the object with pose χ which are projected to the image point x. 121
O_i	Set of points of the i-th object with pose χ_i which are projected to the image point x. 108
$\mathcal{P}$	Set of particles used for halftoning. 151
P	Projection matrix. 24 *or*: Plane is Hessian form. 22, 90 *or*: Probability distribution. 26
$\boldsymbol{p}_n^k$	Position of the n-th particle at time k. 152
$\boldsymbol{p}, \boldsymbol{p}_i$	Position of a particle, or of the i-th particle, respectively. 12, 151
p	Probability density function. 26
$p(s, x)$	Probability density function estimated in a local Gaussian model. 30
p_{in}	PDF of the inside of the (only) object. 69, 78
$p_{i,\text{in}}$	PDF of the inside of the i-th object. 83, 108
p_i^j	PDF of the j part of the i-th object model. 121
p^j	PDF of M^j, i.e. the j part of an object model. 121
p_{out}	PDF of the outside of the (only) object. 69, 78
$p_{i,\text{out}}$	PDF of the outside of the i-th object. 83, 108
q, q_i	Charge of a particle, or of the i-th particle, respectively. 12, 151 *or*: Number quantised values when compressing. 37, 200
R	Retinal plane. 23
$\mathbb{R}$	The set of real numbers.
r	Ratio between one pixel in the image and one dot on the output device. 156 *or*: Part of the pose change threshold T. 81
$\mathbb{R}^+$	The set of positive real numbers.
$SE(3)$	The Lie group of rigid motions in 3-D. 15
$se(3)$	The Lie algebra corresponding to $SE(3)$. 15
$SO(3)$	The Lie group of rotations in 3-D. 15
$so(3)$	The Lie algebra corresponding to $SO(3)$. 15
T	Threshold below which the pose changes must be before the next frame is considered when tracking. 81, 239, 240 *or*: Threshold for the maximal distance from the projected object models considered in two of our video compression codecs. 200, 202 *or*: Threshold used for adaptive rectangular splitting when compressing images. 34 *or*: Turbulence field used to prevent regular patterns when performing halftoning. 161, 238

Notation	Description
t	Variable denoting time. 6, 88 *or*: Part of the pose change threshold T. 81
$u(x)$	Input image (discrete). 151
$u(x,t)$	Time dependent image evolution. 6
v_i^j	Visibility function of the j-th object part of the i-th object, whereby $j = 0$ denotes the background region. 121
v_i	Visibility function of the i-th object, whereby 0 denotes the background region. 108
v^j	Visibility function of the j-th object part, whereby 0 denotes the background region. 121
W	Potential energy of a set of charges particles in an electric field. 152
w_S	Fraction of charges located in small particles. 157
w_M	Fraction of charges located in medium-sized particles. 157
w_L	Fraction of charges located in large particles. 157
X	Random variable. 26 *or*: General purpose variable, often the first coordinate of a vector.
cot	Cotangent, i.e. the reciprocal of the tangent.
diag(A)	A vector containing the diagonal entries of the quadratic matrix A.
div(v)	Divergence of a vector $(v_1, \dots, v_n)^\top$: $\operatorname{div}(v) = \sum_{i=1}^{n} \partial_{x_i} v$.
$\exp(X)$	Exponential operator. Note that X can be a real number, a quadratic matrix, or an element of a Lie algebra. 15
$\log(X)$	Exponential operator. Note that X can be a real number, a quadratic matrix, or an element of a Lie algebra. 17
A^T	The transpose of the matrix or vector A.
tr(A)	The trace of the matrix A, i.e. the sum of the entries on the main diagonal.
α	Tradeoff parameter in multi-class halftoning. 162 *or*: Merging threshold used in the automatic component generation algorithm. 123
β	Regularisation weight for attractive grid forces. 158
χ	The $6 + n$ pose parameters of an articulated object. 19
χ_i	The pose parameters of the ith articulated object. 108
$\delta(x,y)$	Dirac delta distribution. 134
Δu	Laplacian of the image u. See Equation (2.7) for more details. 7
ε_0	Electric constant. 12
η	Normalised gradient. 9
Γ	Discrete image domain. 151
λ	Contrast parameter for nonlinear diffusion and inpainting operators. 8 *or*: Contrast weight used in the force vector $\boldsymbol{F}_{\boldsymbol{g},n}$. 158 *or*: Weighting parameter in combined segmentation and pose tracking. 69
μ	Mean of a Gaussian. 7, 28
$\tilde{\mu}$	Estimation of the mean of a Gaussian. 29
μ_0	Magnetic constant. 12

Notation	Description
∇	Gradient operator.
ν	Weighting parameter for image segmentation using level sets. 67
Ω	Continuous image domain. 5, 27
	or: A non-empty set for σ-algebras and probability spaces. 34
Ω_1, Ω_2	Parts of the image domain, e.g. from segmentation. 66
$\Omega_{\text{in}}, \Omega_2$	Part of the image in which the tracked object is visible. 79
$\Omega_{\text{out}}, \Omega_2$	Parts of the image in which the tracked object is not visible. 79
Ω_j	j-th subregion of the complete image obtained by segmentation. 135
Φ	Level-set function. 67
	or: Electrical flux. 12
Φ_0	Level-set function used as segmentation prior. 69
σ	Standard deviation, usually of a Gaussian. 7, 28
$\tilde{\sigma}$	Estimation of a standard deviation. 29
τ	Time step size. 88, 152
θ	Angle between the coordinate axes of the 2-D image coordinate system. 24
θ_i	i-th joint angle in a kinematic chain. 19
Θ	Vector containing all joint angles of a kinematic chain. 19
ξ	An element of the Lie Group SE(3), usually denoting the six pose parameter of a rigid object. 15
ξ_{last}	Initial pose estimation in a frame. 70
ξ_i	Rotation axis of the i-th joint of a kinematic chain. 19
$A * B$	Convolution of the two functions A and B.
$[\cdot, \cdot]$	Lie bracket or commutator. 14
$\exists$	Universal quantification.
$\forall$	Existential quantification.
∂_t	Partial differentiation with respect to t.
$\oint_{c_r}$	Line integral along the closed curve c_r. 12
$a \times b$	Cross product of the vectors a and b.
	or: The Cartesian product of the two sets a and b.
$\vee$	Vee operator (convertion from matrix to vector notation in Lie algebras). 16
$\omega^\vee$	Vee operator applied on ω. 16
$\wedge$	Wedge operator (convertion from vector to matrix notation in Lie algebras). 16
$\hat{\omega}, \omega^\wedge$	Wedge operator applied on ω. 16

List of Figures

Bibliography

"When a thing has been said and well said, have no scruple: take it and copy it."

Anatole France (1844 – 1924)

[1] M. A. Abidi and T. Chandra. Pose estimation for camera calibration and landmark tracking. In *Proc. International Conf. Robotics and Automation*, volume 1, pages 420–426, Cincinnati, May 1990.

[2] M. Abomhara, O. O. Khalifa, O. Zakaria, A. Zaidan, B. Zaidan, and A. Rame. Video compression techniques: An overview. *Journal of Applied Sciences*, 10(16):1834–1840, 2010.

[3] T. Acar and M. Gökmen. Image coding using weak membrane model of images. In A. K. Katsaggelos, editor, *Visual Communications and Image Processing '94*, volume 2308 of *Proceedings of SPIE*, pages 1221–1230. SPIE Press, Bellingham, 1994.

[4] A. Agarwal and B. Triggs. Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58, January 2006.

[5] J. K. Aggarwal and Q. Cai. Human motion analysis: a review. *Computer Vision and Image Understanding*, 73(3):428–440, 1999.

[6] F. Alter, S. Durand, and J. Froment. Adapted total variation for artifact free decompression of JPEG images. *Journal of Mathematical Imaging and Vision*, 23(2):199–211, Sept. 2005.

[7] L. Álvarez León, L. Gómez Déniz, P. Henríquez Castellano, and L. Mazorra Manrique de Lara. A variational approach to camera motion smoothing. *Differential Equations and Applications – DEA*, 3(4):555–564, 2011.

[8] L. Álvarez León, A. J. Salgado de la Nuez, and J. Sánchez Pérez. Robust detection and ordering of ellipses on a calibration pattern. *Pattern Recognition and Image Analysis*, 17(4):508–522, 2007.

[9] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[10] G. Aronsson. Extension of functions satisfying Lipschitz conditions. *Arkiv för Matematik*, 6(6):551–561, June 1967.

[11] X. Artigas and L. Torres. A model-based enhanced approach to distributed video coding. In *Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2005)*. Article No. 1127, 2005.

[12] G. Aubert, M. Marlaud, O. Faugeras, and S. Jehan-Besson. Image segmentation using active contours: calculus of variations or shape gradients? *SIAM Journal on Applied Mathematics*, 63(6):2128–2154, 2003.

[13] V. Aurich and U. Daub. Bilddatenkompression mit geplanten Verlusten und hoher Rate. In B. Jähne, P. Geißler, H. Haußecker, and F. Hering, editors, *Mustererkennung 1996*, pages 138–146. Springer, Berlin, 1996.

[14] B. Babenko, M. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *Proc. 2009 IEEE Computer Society Conference of Computer Vision and Pattern Recognition (CVPR)*, pages 983–990, New York, NY, USA, June 2009. IEEE Computer Society Press.

[15] E. Bae and J. Weickert. Partial differential equations for interpolation and compression of surfaces. In M. Daehlen, M. Floater, T. Lyche, J.-L. Merrien, K. Mørken, and L. L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces*, volume 5862 of *Lecture Notes in Computer Science*, pages 1–14. Springer, Berlin, 2010.

[16] M. Balzer. ccvt - project hosting on google code. `http://code.google.com/p/ccvt/`, 2010. Last visited January 14, 2010.

[17] M. Balzer, T. Schlömer, and O. Deussen. Capacity-constrained point distributions: A variant of Lloyd's method. *ACM Transactions on Graphics*, 28(3):Article No. 86, 2009.

[18] M. S. Bartlett. *An Introduction to Stochastic Processes with Special Reference to Methods and Applications*. Cambridge University Press, Cambridge, UK, third edition, 1978.

[19] B. E. Bayer. An optimum method for two-level rendition of continuous-tone pictures. In *IEEE 1973 International Conference on Communications*, volume 1, pages (26–11)–(26–15), 1973.

[20] Z. Belhachmi, D. Bucur, B. Burgeth, and J. Weickert. How to choose interpolation data in images. *SIAM Journal on Applied Mathematics*, 70(1):333–352, 2009.

[21] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proc. SIGGRAPH 2000*, pages 417–424, New Orleans, LI, July 2000.

[22] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[23] J. R. Beveridge. *Local Search Algorithms for Geometric Object Recognition: Optimal Correspondence and Pose*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, May 1993.

[24] A. Blake and M. Isard. *Active Contours*. Springer, London, 1998.

[25] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, MA, 1987.

[26] F. Bornemann and T. März. Fast image inpainting based on coherence transport. *Journal of Mathematical Imaging and Vision*, 28(3):259–278, July 2007.

[27] S. Bougleux, G. Peyré, and L. Cohen. Image compression with anisotropic triangulations. In *Proc. Tenth International Conference on Computer Vision*, Kyoto, Japan, Oct. 2009.

[28] A. C. Bovik, M. Clark, and W. S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):55–73, 1990.

[29] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proc. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8–15, Washington, DC, USA, 1998. IEEE Computer Society Press.

[30] T. Brox, B. Rosenhahn, D. Cremers, and H. Seidel. Nonparametric density estimation with adaptive anisotropic kernels for human motion tracking. In A. Elgammal, B. Rosenhahn, and R. Klette, editors, *Proc. 2nd International Workshop on Human Motion*, volume 4814 of *Lecture Notes in Computer Science*, pages 152–165, Rio de Janeiro, Brazil, October 2007. Springer.

[31] T. Brox, B. Rosenhahn, D. Cremers, and H.-P. Seidel. High accuracy optical flow serves 3-D pose tracking: exploiting contour and flow based constraints. In H. Bischof, A. Leonardis, and A. Pinz, editors, *Computer Vision – ECCV 2006, Part II*, volume 3952 of *Lecture Notes in Computer Science*, pages 98–111. Springer, Berlin, 2006.

[32] T. Brox, B. Rosenhahn, U. Kersting, and D. Cremers. Nonparametric density estimation for human pose tracking. In K. Franke, K. Müller, B. Nickolay, and R. Schäfer, editors, *Pattern Recognition*, volume 4174 of *Lecture Notes in Computer Science*, pages 546–555, Berlin, September 2006. Springer.

[33] T. Brox, B. Rosenhahn, and J. Weickert. Three-dimensional shape knowledge for joint image segmentation and pose estimation. In W. Kropatsch, R. Sablatnig, and A. Hanbury, editors, *Pattern Recognition*, volume 3663 of *Lecture Notes in Computer Science*, pages 109–116. Springer, Berlin, 2005.

[34] T. Brox, M. Rousson, R. Deriche, and J. Weickert. Unsupervised segmentation incorporating colour, texture, and motion. In N. Petkov and M. A. Westenberg, editors, *Computer Analysis of Images and Patterns*, volume 2756 of *Lecture Notes in Computer Science*, pages 353–360. Springer, Berlin, 2003.

[35] T. Brox and J. Weickert. Level set based image segmentation with multiple regions. In C. Rasmussen, H. Bülthoff, M. Giese, and B. Schölkopf, editors, *Pattern Recognition*, volume 3175 of *Lecture Notes in Computer Science*, pages 415–423. Springer, Berlin, 2004.

[36] T. Brox and J. Weickert. Level set segmentation with multiple regions. *IEEE Transactions on Image Processing*, 15(10):3213–3218, Oct. 2006.

[37] T. Brox and J. Weickert. A TV flow based local scale estimate and its application to texture discrimination. *Journal of Visual Communication and Image Representation*, 17(5):1053–1073, Oct. 2006.

[38] M. A. Brubaker, D. J. Fleet, and A. Hertzmann. Physics-based person tracking using the anthropomorphic walker. *International Journal of Computer Vision*, 87(1/2):140–155, March 2010.

[39] A. M. Bruckstein. On image extrapolation. Technical Report CIS9316, Computer Science Department, Technion, Haifa, Israel, Apr. 1993.

[40] A. Bruhn and J. Weickert. A confidence measure for variational optic flow methods. In R. Klette, R. Kozera, L. Noakes, and J. Weickert, editors, *Geometric Properties from Incomplete Data*, volume 31 of *Computational Imaging and Vision*, pages 283–297. Springer, Dordrecht, 2006.

[41] A. O. Bălan, L. Sigal, M. J. Black, J. E. Davis, and H. W. Haussecker. Detailed human shape and pose from images. In *Proc. 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[42] M. D. Buhmann. *Radial Basis Functions*. Cambridge University Press, Cambridge, UK, 2003.

[43] V. Caselles, J.-M. Morel, and C. Sbert. An axiomatic approach to image interpolation. *IEEE Transactions on Image Processing*, 7(3):376–386, Mar. 1998.

[44] F. Catté, P.-L. Lions, J.-M. Morel, and T. Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM Journal on Numerical Analysis*, 32:1895–1909, 1992.

[45] T. F. Chan and H. M. Zhou. Feature preserving lossy image compression using nonlinear PDE's. In F. T. Luk, editor, *Advanced Signal Processing Algorithms, Architectures, and Implementations VIII*, volume 3461 of *Proceedings of SPIE*, pages 316–327. SPIE Press, Bellingham, 1998.

[46] T. F. Chan and H. M. Zhou. Total variation improved wavelet thresholding in image compression. In *Proc. Seventh International Conference on Image Processing*, volume II, pages 391–394, Vancouver, Canada, Sept. 2000.

[47] J. Chang, B. Alain, and V. Ostromoukhov. Structure-aware error diffusion. *ACM Transactions on Graphics*, 28(5):Article No. 162, 2009.

[48] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on Image Processing*, 6(2):298–311, 1997.

[49] F. S. Chaudhry and D. C. Handscomb. Smooth motion of a rigid body in 2D and 3D. In *IV '97 Proceedings of the IEEE Conference on Information Visualisation*, pages 205–210, Washington, DC, USA, 1997. IEEE Computer Society Press.

[50] S. Y. Cheng and M. M. Trivedi. Articulated human body pose inference from voxel data using a kinematically constrained gaussian mixture model. In *2nd Workshop on Evaluation of Articulated Human Motion and Pose Estimation (EHuM$_2$), CVPR*. IEEE, 2007.

[51] M. F. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. *ACM Transactions on Graphics*, 22(3):287–294, 2003.

[52] Comité Consultatif International Téléphonique et Télégraphique. Codecs for videoconferencing using primary digital group transmission, 1984. CCITT Recommendation H.120, version 1. Latest version from 1993.

[53] Comité Consultatif International Téléphonique et Télégraphique. Video codec for audiovisual services at p x 384 kbit/s, 1988. CCITT Recommendation H.261, version 1. Latest version from 1993.

[54] R. L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, 1986.

[55] S. Corazza, L. Mündermann, E. Gambaretto, G. Ferrigno, and T. P. Andriacchi. Markerless motion capture through visual hull, articulated ICP and subject specific model generation. *International Journal of Computer Vision*, 87(1/2):156–169, March 2010.

[56] D. Cremers and S. Soatto. Motion competition: A variational framework for piecewise parametric motion segmentation. *International Journal of Computer Vision*, 62(3):249–265, May 2005.

[57] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. Bilayer segmentation of live video. In *Proc. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 53–60, Washington, DC, USA, 2006. IEEE Computer Society Press.

[58] S. Dambreville, R. Sandhu, A. Yezzi, and A. Tannenbaum. Robust 3D pose estimation and efficient 2D region-based segmentation from a 3D shape prior. In D. Forsyth, P. Torr, and A. Zisserman, editors, *Computer Vision – ECCV 2008, Part II*, volume 5303 of *Lecture Notes in Computer Science*, pages 169–182. Springer, 2008.

[59] L. Demaret, N. Dyn, and A. Iske. Image compression by linear splines over adaptive triangulations. *Signal Processing*, 86(7):1604–1616, 2006.

[60] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39(1):1–38, 1977.

[61] A. Dervieux and F. Thomasset. A finite element method for the simulation of Rayleigh–Taylor instability. In R. Rautman, editor, *Approximation Methods for Navier–Stokes Problems*, volume 771 of *Lecture Notes in Mathematics*, pages 145–158. Springer, Berlin, 1979.

[62] U. Y. Desai, M. M. Mizuki, I. Masaki, and B. K. P. Horn. Edge and mean based image compression. Technical Report 1584 (A.I. Memo), Artificial Intelligence Lab., Massachusetts Institute of Technology, Cambridge, MA, U.S.A., Nov. 1996.

[63] O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr, and P. Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In *Proc. SIGGRAPH '98*, pages 275–286, New York, NY, USA, 1998. ACM.

[64] M. Dipperstein. Michael Dipperstein's page o'stuff. http://michael.dipperstein.com/index.html, January 2010. Last visited August 3, 2010.

[65] R. Distasi, M. Nappi, and S. Vitulano. Image compression by B-tree triangular coding. *IEEE Transactions on Communications*, 45(9):1095–1100, Sept. 1997.

[66] J. Dong. The first JCT-VC meeting. `http://www.h265.net/2010/06/the-first-jct-vc-meeting-dresden-de.html`, 2009. Last visited August 24, 2010.

[67] T. Drummond and R. Cipolla. Real-time tracking of multiple articulated structures in multiple views. In D. Vernon, editor, *Computer Vision – ECCV 2000, Part II*, volume 1843 of *Lecture Notes in Computer Science*, pages 20–36, Berlin, 2000. Springer.

[68] J. Duchon. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *RAIRO Mathematical Models and Methods in the Applied Sciences*, 10:5–12, 1976.

[69] D. Dunbar and G. Humphreys. A spatial data structure for fast Poisson-disk sample generation. In *Proc. SIGGRAPH '06*, pages 503–508, New York, NY, USA, 2006. ACM.

[70] S. Durand and M. Nikolova. Restoration of wavelet coefficients by minimizing a specially designed objective function. In O. Faugeras and N. Paragios, editors, *Proc. Second IEEE Workshop on Geometric and Level Set Methods in Computer Vision*, pages 145–152, Nice, France, Oct. 2003. INRIA.

[71] P. Eisert and B. Girod. Facial expression analysis for model-based coding of video sequences. In *Proc. Picture Coding Symposium (PCS '97)*, pages 33–38, 1997.

[72] P. Eisert and B. Girod. Model-based coding of facial image sequences at varying illumination conditions. In *Proc. 10th Image and Multidimensional Digital Signal Processing Workshop IMDSP*, pages 119–122, 1998.

[73] J. H. Elder. Are edges incomplete? *International Journal of Computer Vision*, 34(2/3):97–122, 1999.

[74] C. Elkan. Using the triangle inequality to accelerate k-Means. In T. Fawcett and N. Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference on Machine Learning*, pages 147–153. AAAI Press, 2003.

[75] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno. 2D Euclidean distance transform algorithms: A comparative survey. *Journal of Computuing Surveys*, 40(1):1–44, 2008. See also companion website at http://distance.sourceforge.net/.

[76] Facebook statistics page. http://www.facebook.com/press/info.php?statistics. Last visited January 11, 2012.

[77] R. Fattal. Blue-noise point sampling using kernel density model. In *Proc. SIGGRAPH 2011*, volume 28, New York, NY, USA, 2011. ACM.

[78] O. Faugeras, Q.-T. Luong, and T. Papadopoulo. *The Geometry of Multiple Images*. MIT Press, Cambridge, MA, 2001.

[79] R. P. Feynman, R. B. Leighton, and M. Sands. *The Feynman Lectures on Physics*, volume 2. Addison–Wesley, third edition, Oct. 2005.

[80] R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial grey scale. *Proceedings of the Society of Information Display*, 17:75–77, 1976.

[81] R. Forchheimer and O. Fahlander. Low bit-rate coding through animation. In *Proceedings of Picture Coding Symposium (PCS '83)*, pages 113–114, March 1983.

[82] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *Proc. ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data*, pages 281–305, Interlaken, Switzerland, June 1987.

[83] D. A. Forsyth, O. Arikan, L. Ikemoto, J. O'Brien, and D. Ramanan. Computational studies of human motion: part 1, tracking and motion synthesis. *Foundations and Trends in Computer Graphics and Vision*, 1(2–3):77–254, 2005.

[84] A. Fossati, M. Dimitrijevic, V. Lepetit, and P. Fua. Bridging the gap between detection and tracking for 3D monocular video-based motion capture. In *Proc. 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, MI, June 2007. IEEE Computer Society Press.

[85] D. Gabor. Theory of communication. *The Journal of the Institution of of Electrical Engineers*, 93(3):429–457, 1946.

[86] J. Gailly and M. Adler. The gzip home page. http://www.gzip.org/, 2010. Last visited August 3, 2010.

[87] I. Galić, J. Weickert, M. Welk, A. Bruhn, A. Belyaev, and H.-P. Seidel. Towards PDE-based image compression. In N. Paragios, O. Faugeras, T. Chan, and C. Schnörr, editors, *Variational, Geometric and Level-Set Methods in Computer Vision*, volume 3752 of *Lecture Notes in Computer Science*, pages 37–48. Springer, Berlin, 2005.

[88] I. Galić, J. Weickert, M. Welk, A. Bruhn, A. Belyaev, and H.-P. Seidel. Image compression with anisotropic diffusion. *Journal of Mathematical Imaging and Vision*, 31(2–3):255–269, July 2008.

[89] J. Gall, B. Rosenhahn, T. Brox, and H. Seidel. Optimization and filtering for human motion capture. *International Journal of Computer Vision*, 87(1/2):75–92, March 2010.

[90] M. N. Gamito and S. C. Maddock. Accurate multidimensional poisson-disk sampling. *ACM Transactions on Graphics*, 29(1):Article No. 8, 2009.

[91] D. M. Gavrila. The visual analysis of human movement: a survey. *Computer Vision and Image Understanding*, 73(1):82–98, January 1999.

[92] P. Gehler. Mpikmeans. `http://mloss.org/software/view/48/`, 2007. Last visited August 24, 2010.

[93] R. Geist, R. Reynolds, and D. Suggs. A Markovian framework for digital halftoning. *ACM Transactions on Graphics*, 12(2):136–159, 1993.

[94] W. Gerbino. *The Perception of Illusory Contours*, chapter Quasiperceptual Margins in Homogeneously Stimulated Fields, pages 40–49. Springer, Berlin, 1987.

[95] B. Girod, K. B. Younes, R. Bernstein, P. Eisert, N. Färber, F. Hartung, U. Horn, E. Steinbach, K. Stuhlmüller, and T. Wiegand. Recent advances in video compression. *Proc. IEEE International Symposium on Circuits and Systems*, 2:580–583, 1996.

[96] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison–Wesley, Reading, second edition, 2002.

[97] W. M. Goodall. Television by pulse code modulation. *Bell System Technical Journal*, 30:33–49, Jan. 1951.

[98] A. Gothandaraman, R. Whitaker, and J. Gregor. Total variation for the removal of blocking effects in DCT based encoding. In *Proc. 2001 IEEE International Conference on Image Processing*, volume 2, pages 455–458, Thessaloniki, Greece, Oct. 2001.

[99] A. R. Gourlay. Hopscotch: a fast second-order partial differential equation solver. *IMA Journal of Applied Mathematics*, 6(4):375–390, 1970.

[100] L. Granai, T. Vlachos, M. Hamouz, J. R. Tena, and T. Davies. Model-based coding of 3D head sequences. In *3DTV07*, 2007.

[101] S. Grewenig, J. Weickert, and A. Bruhn. From box filtering to fast explicit diffusion. In M. Goesele, S. Roth, A. Kuijper, B. Schiele, and K. Schindler, editors, *Pattern Recognition*, volume 6376 of *Lecture Notes in Computer Science*, pages 533–542. Springer, Berlin, 2010.

[102] W. E. L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints*. MIT Press, Massachusetts Institute of Technology, Cambridge, MA, 1990.

[103] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. In *Proc. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 22–29, Washington, DC, USA, 1998. IEEE Computer Society Press.

[104] P. Gwosdek, C. Schmaltz, J. Weickert, and T. Teuber. Fast electrostatic halftoning. *Journal of Real-Time Image Processing*, 2011. Online First.

[105] J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2(1):84–90, 1960.

[106] K. M. Hanson. Halftoning and quasi-Monte Carlo. In K. M. Hanson and F. M. Hemez, editors, *Sensitivity Analysis of Model Output*, pages 430–442. Los Alamos Research Library, 2005.

[107] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2002.

[108] G. Hellwig. *Partial Differential Equations*. Teubner, Stuttgart, 1977.

[109] L. Herda, R. Urtasun, and P. Fua. Implicit surface joint limits to constrain video-based motion capture. In T. Pajdla and J. Matas, editors, *Computer Vision – ECCV 2004, Part II*, volume 3022 of *Lecture Notes in Computer Science*, pages 405–418, Prague, Czech Republic, May 2004. Springer.

[110] S. Hiller, O. Deussen, and A. Keller. Tiled blue noise samples. In *VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001*, pages 265–272. Aka, 2001.

[111] S. Hoffmann. Grey-value optimisation in PDE-based image compression. Bachelor's thesis, Department of Computer Science, Saarland University, 2010.

[112] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[113] N. Howe. Recognition-based motion capture and the HumanEva II test data. In *2nd Workshop on Evaluation of Articulated Human Motion and Pose Estimation (EHuM$_2$), CVPR*. IEEE, 2007.

[114] T. Huang and L. Tang. 3d model-based video coding: Computer vision meets computer graphics. In R. Chin, H. Ip, A. Naiman, and T. Pong, editors, *Image Analysis Applications and Computer Graphics*, volume 1024 of *Lecture Notes in Computer Science*, pages 1–8, Berlin, 1995. Springer.

[115] C. Hue, J. L. Cadre, and P. Perez. Tracking multiple objects with particle filtering. *IEEE Trans. on Aerospace and Electronic Systems*, 38(3):791–812, July 2002.

[116] D. A. Huffman. A method for the construction of minimum redundancy codes. *Proceedings of the IRE*, 40:1098–1101, 1952.

[117] R. Hummel and R. Moniot. Reconstructions from zero-crossings in scale space. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37:2111–2130, 1989.

[118] Z. Husz, A. M. Wallace, and P. R. Green. Evaluation of a hierarchical partitioned particle filter with action primitives. In *2nd Workshop on Evaluation of Articulated Human Motion and Pose Estimation ($EHuM_2$), CVPR*, pages 1006–1015. IEEE, 2007.

[119] T. Iijima. Basic theory of pattern observation. In *Papers of Technical Group on Automata and Automatic Control*. IECE, Japan, Dec. 1959. In Japanese.

[120] P. W. M. Ilbery. U.S. patent No. 6,124,844, 2000.

[121] International Telecommunication Union. Information technology – generic coding of moving picture and associated audio information: Video, 1995. ITU-T Recommendation H.262, version 1. Latest version from 2000.

[122] International Telecommunication Union. Video coding for low bit rate communication, 1996. ITU-T Recommendation H.263, version 1. Latest version from 2005.

[123] International Telecommunication Union. Information technology - coded representation of picture and audio information - lossy/lossless coding of bi-level images, 2000. ITU-T Recommendation T.88, version 1. Latest version from 2004.

[124] International Telecommunication Union. Advanced video coding for generic audiovisual services, 2003. ITU-T Recommendation H.264, version 1. Latest version from 2010.

[125] ISO/CIE. CIE colorimetry – part 4: 1976 L*a*b* colour space. ISO 11664-4:2008(E)/CIE S 014-4/E:2007, 2007.

[126] ISO/IEC. Information technology – coding of moving pictures and associated audio for digital storage media at up to about 1,5 mbit/s – part 2: Video, 1993. ISO/IEC 11172-2. Latest corrections in 2006.

[127] ISO/IEC. Information technology – generic coding of moving pictures and associated audio information, 1996. ISO/IEC 13818. Latest corrections in 2006.

[128] ISO/IEC. Information technology – lossy/lossless coding of bi-level images, 2001. ISO/IEC 14492. Latest corrections in 2004.

[129] ISO/IEC. Information technology – coding of audio-visual objects – part 10: Advanced video coding, 2003. ISO/IEC 14496-10. Latest corrections in 2009.

[130] J. Jacod and P. Protter. *Probability Essentials*. Universitext, Springer, 2000.

[131] J. F. Jarvis, C. N. Judice, and W. H. Ninke. A survey of techniques for the display of continuous tone pictures on bilevel displays. *Computer Graphics and Image Processing*, 5:13–40, 1976.

[132] J. F. Jarvis and C. S. Roberts. A new technique for displaying continuous tone images on a bilevel display. *IEEE Transactions on Communications*, 24:891–898, 1976.

[133] R. Javůrek. Model based facial video sequences coding. In *Radioelektronika 2003*, pages 115–118, 2003.

[134] H. Jin, D. Cremers, A. Yezzi, and S. Soatto. Shedding light in stereoscopic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 36–42, July 2004.

[135] P. Johansen, S. Skelboe, K. Grue, and J. D. Andersen. Representing signals by their top-points in scale space. In *Proc. Eighth International Conference on Pattern Recognition*, pages 215–217, Paris, France, Oct. 1986.

[136] T. Kadir and M. Brady. Unsupervised non-parametric region segmentation using level sets. In *Proc. Ninth International Conference on Computer Vision*, volume 2, pages 1267–1274, Washington, DC, USA, 2003. IEEE Computer Society Press.

[137] G. Kanizsa. Margini quasi-percettivi in campi con stimolazione omogenea. *Rivista di Psicologia*, 49(1):7–30, 1955. An translation into English was published in 1987, see [94].

[138] F. M. W. Kanters, M. Lillholm, R. Duits, B. J. P. Jansen, B. Platel, L. Florack, and B. M. ter Haar Romeny. On image reconstruction from multiscale top points. In R. Kimmel, N. Sochen, and J. Weickert, editors, *Scale Space and PDE Methods in Computer Vision*, volume 3459 of *Lecture Notes in Computer Science*, pages 431–439. Springer, Berlin, 2005.

[139] J. Kim, M. Çetin, and A. S. Willsky. Nonparametric shape priors for active contour-based image segmentation. *Signal Processing*, 87(12):3021–3044, 2007.

[140] H. Kipphan. *Handbook of print media: Technologies and production methods*. Springer, 2001.

[141] K. T. Knox. Edge enhancement in error diffusion. In *Paper Summaries from The Society for Imaging Science and Technology, 42nd Annual Conference*, pages 310–313, Boston, May 1989.

[142] T. Kollig and A. Keller. Efficient illumination by high dynamic range images. In P. Christensen and D. Cohen-Or, editors, *EGRW '03: Proc. 14th Eurographics Workshop on Rendering*, pages 45–50, Aire-la-Ville, Switzerland, 2003. Eurographics Association.

[143] J. Kopf, D. Cohen-Or, O. Deussen, and D. Lischinski. Recursive Wang tiles for real-time blue noise. *ACM Transactions on Graphics*, 25(3):509–518, 2006.

[144] I. Kopilovic and T. Szirányi. Artifact reduction with diffusion preprocessing for image compression. *Optical Engineering*, 44(2):1–14, Feb. 2005.

[145] U. Krengel. *Einführung in die Wahrscheinlichkeitstheorie und Statistik*. Vieweg, seventh edition, Juni 2003.

[146] D. J. Kriegman, B. Vijayakumar, and J. Ponce. Constraints for recognizing and locating curved 3D objects from monocular image features. In G. Sandini, editor, *Computer Vision – ECCV '92*, volume 588 of *Lecture Notes in Computer Science*, pages 829–833, Berlin, 1992. Springer.

[147] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[148] A. Lagae and P. Dutré. A procedural object distribution function. *ACM Transactions on Graphics*, 24(4):1442–1461, 2005.

[149] A. Lagae and P. Dutré. Template Poisson disk tiles. Technical Report 413, Departement Computerwetenschappen, Katholieke Universiteit Leuven, Heverlee, Belgium, 2005.

[150] A. Lagae and P. Dutré. A comparison of methods for generating poisson disk distributions. *Computer Graphics Forum*, 27(1):114–129, March 2008.

[151] S. Lankton and A. Tannenbaum. Localizing region-based active contours. *IEEE Transactions on Image Processing*, 17(11):2029–2039, November 2008.

[152] M. J. Leotta and J. L. Mundy. Predicting high resolution image edges with a generic, adaptive, 3-D vehicle model. In *Proc. 2009 IEEE Computer Society Conference of Computer Vision and Pattern Recognition (CVPR)*, pages 1311–1318, New York, NY, USA, June 2009. IEEE Computer Society Press.

[153] V. Lepetit and P. Fua. Monocular model-based 3D tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision*, 2005.

[154] H. Li, D. Nehab, L.-Y. Wei, P. Sander, and C.-W. Fu. Fast capacity constrained voronoi tessellation. Technical Report MSR-TR-2009-174, Microsoft Research, Oct. 2009.

[155] H. Li, D. Nehab, L.-Y. Wei, P. V. Sander, and C.-W. Fu. Fast capacity constrained voronoi tessellation. In *Proc. 2009 Symposium on Interactive 3D Graphics and Games*, pages 13:1–13:1, New York, NY, USA, 2009. ACM.

[156] M. Lillholm, M. Nielsen, and L. D. Griffin. Feature-based image analysis. *International Journal of Computer Vision*, 52(2/3):73–95, 2003.

[157] S. Lloyd. Computational capacity of the universe. *Physical Review*, 88(23):237901–1–4, 2002.

[158] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. First proposed in Bell Telephone Laboratories Paper in 1957.

[159] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[160] D. G. Lowe. Solving for the parameters of object models from image descriptions. In *Proc. ARPA Image Understanding Workshop*, pages 121–127, College Park, April 1980.

[161] D. G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(2):441–450, May 1991.

[162] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. Seventh International Joint Conference on Artificial Intelligence*, pages 674–679, Vancouver, Canada, Aug. 1981.

[163] Y. Ma, S. Soatto, J. Kosecká, and S. S. Sastry. *An Invitation to 3-D Vision*. Springer, New York, 2002.

[164] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. L. Cam and J. Neyman, editors, *Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[165] M. Mahoney. Data compression programs. http://mattmahoney.net/dc/, 2009. Last visited November 30, 2009.

[166] M. Mahoney. *Data Compression Explained*. 2010. Last update at time of last visit: Oct. 17, 2010. The book can be downloaded from http://mattmahoney.net/dc/dce.html.

[167] M. Mainberger, S. Hoffmann, J. Weickert, C. H. Tang, D. Johannsen, F. Neumann, and B. Doerr. Optimising spatial and tonal data for homogeneous diffusion inpainting. In A. M. Bruckstein, B. ter Haar Romeny, A. M. Bronstein, and M. M. Bronstein, editors, *Proc. Third International Conference on Scale Space and Variational Methods in Computer Vision*, volume 6667 of *Lecture Notes in Computer Science*. Springer, Berlin, June 2011. to appear.

[168] M. Mainberger and J. Weickert. Edge-based image compression with homogeneous diffusion. In X. Jiang and N. Petkov, editors, *Computer Analysis of Images and Patterns*, volume 5702 of *Lecture Notes in Computer Science*, pages 476–483. Springer, Berlin, 2009.

[169] G. N. N. Martin. Range encoder range encoding: An algorithm for removing redundancy from a digitized message. Presented at the Video & Data Recording Conference, Southampton, UK, 1979.

[170] S. Masnou and J.-M. Morel. Level lines based disocclusion. In *Proc. 1998 IEEE International Conference on Image Processing*, volume 3, pages 259–263, Chicago, IL, Oct. 1998.

[171] M. McCool and E. Fiume. Hierarchical Poisson disk sampling distributions. In K. S. Booth and A. Fournier, editors, *Proc. the conference on Graphics interface '92*, pages 94–105, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[172] G. G. Medioni and Y. Chen. Object modeling by registration of multiple range images. In *Proc. of the 1991 IEEE International Conference on Robotics and Automation*, pages 2724–2729, Washington, DC, USA, 1991. IEEE Computer Society Press.

[173] D. Meschede. *Gerthsen Physik*. Springer-Lehrbuch. Springer, Berlin, 23 (revised) edition, 2006.

[174] M. Meyer, P. Georgel, and R. T. Whitaker. Robust particle systems for curvature dependent sampling of implicit surfaces. In *Proceedings of the International Conference on Shape Modeling and Applications (SMI)*, pages 124–133. IEEE Computer Society, June 2005.

[175] A. Micilotta, E. Ong, and R. Bowden. Detection and tracking of humans by probabilistic body part assembly. In *Proc. of the British Machine Vision Conference (BMVC'05)*, pages 429–438, Oxford UK, September 2005.

[176] D. P. Mitchell. Generating antialiased images at low sampling densities. In M. C. Stone, editor, *Proc. SIGGRAPH 1987*, volume 21, pages 65–72, New York, NY, USA, July 1987. ACM.

[177] T. B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, November 2001.

[178] T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *International Journal of Computer Vision*, 104(2):90–126, November 2006.

[179] B. Mory, R. Ardon, and J. P. Thiran. Variational segmentation using fuzzy region competition and local non-parametric probability density functions. In *Proc. Eleventh International Conference on Computer Vision*, Washington, DC, USA, October 2007. IEEE Computer Society Press.

[180] D. Mould. Stipple placement using distance in a weighted graph. In D. W. Cunningham, G. W. Meyer, L. Neumann, A. Dunning, and R. Paricio, editors, *Computational Aesthetics 2007: Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging*, pages 45–52. Eurographics Association, 2007.

[181] MPEG-4 Industry Forum. MPEG-4 – the media standard. available at http://www.mpegif.org/public/documents/vault/m4-out-20027.pdf, November 2007.

[182] P. Mrázek. *Nonlinear Diffusion for Image Filtering and Monotonicity Enhancement*. PhD thesis, Czech Technical University, Prague, Czech Republic, June 2001.

[183] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, 1994.

[184] S. K. Nayar and Y. Nakagawa. Shape from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):824–831, 1994.

[185] H. Nguyen, editor. *GPU Gems 3*. Addison–Wesley Professional, Aug. 2007.

[186] J. Ohm. Advances in video compression. In *Proc. the European Signal Processing Conference EUSIPCO '05*, 2005.

[187] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[188] V. Ostromoukhov. A simple and efficient error-diffusion algorithm. In E. Fiume, editor, *Proc. SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 567–572, Los Angeles, CA, Aug. 2001.

[189] V. Ostromoukhov. Sample code for a simple and efficient error-diffusion algorithm. `http://www.iro.umontreal.ca/~ostrom/publications/abstracts.html#SIGGRAPH01_VarcoeffED`, 2010. Last visited January 14, 2010.

[190] V. Ostromoukhov, C. Donohue, and P. Jodoin. Fast hierarchical importance sampling with blue noise properties. *ACM Transactions on Graphics*, 23(3):488–495, 2004.

[191] V. Ostromoukhov and R. Hersch. Artistic screening. In *Proc. SIGGRAPH '95*, pages 219–228, New York, NY, Aug. 1995.

[192] M. Özuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature harvesting for tracking-by-detection. In *Computer Vision – ECCV 2006, Part III*, volume 3953 of *Lecture Notes in Computer Science*, pages 592–605. Springer, Graz, Austria, 2006.

[193] I. S. Pandzic and R. Forchheimer, editors. *MPEG-4 Facial Animation: The Standard, Implementation and Applications*. Wiley, New York, NY, USA, 2003.

[194] W. Pang, Y. Qu, T. Wong, D. Cohen-Or, and P. Heng. Structure-aware halftoning. *ACM Transactions on Graphics*, 27(3):89:1–89:8, Aug. 2008.

[195] N. Paragios and D. R. Geodesic active regions: A new paradigm to deal with frame partition problems in computer vision. *Journal of Visual Communication and Image Representation*, 13(1/2):249–268, 2002.

[196] M. Pardàs and A. Bonafonte. Facial animation parameters extraction and expression detection using hidden markov models. In *Signal Processing: Image Communication*, volume 17, pages 675–688, Amsterdam, 2002. Elvevier.

[197] E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, September 1962.

[198] D. E. Pearson. Developments in model-based video coding. *Proceedings of the IEEE*, 83(6):892–906, 1995.

[199] A. Peck. *Beginning GIMP: From Novice to Professional*. Apress, first edition, 2006.

[200] W. B. Pennebaker and J. L. Mitchell. *JPEG: Still Image Data Compression Standard*. Springer, New York, 1992.

[201] P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:629–639, 1990.

[202] I. G. Petrovsky. *Lectures on Partial Differential Equations*. Dover, New York, 1992.

[203] M. Pharr and G. Humphreys. *Physically Based Rendering – from Theory to Implementation*, chapter Chapter 7, "Sampling and Reconstruction". Morgan Kaufmann, 2004.

[204] M. Piccardi. Background subtraction techniques: a review. *IEEE Transactions on Systems, Man and Cybernetics*, 4:3099–3104, 2004.

[205] R. Pless, J. Larson, S. Siebers, and B. Westover. Evaluation of local models of dynamic backgrounds. In *Proc. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 73–78, 2003.

[206] Y. Pnueli and A. M. Bruckstein. Gridless halftoning: A reincarnation of the old method. *CVGIP: Graphical Models and Image Processing*, 58(1):38–64, Jan. 1996.

[207] R. Poppe. Evaluating example-based pose estimation: Experiments on the humaneva sets. In *2nd Workshop on Evaluation of Articulated Human Motion and Pose Estimation ($EHuM_2$), CVPR*. IEEE, 2007.

[208] R. Poppe. Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding*, 108(1-2):4–18, October 2007.

[209] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, second edition, 1992.

[210] M. Pressigout and E. Marchand. Real-time 3D model-based tracking: Combining edge and texture information. In *IEEE Int. Conf. on Robotics and Automation, ICRA'06*, pages 2726–2731, Orlando, Florida, May 2006.

[211] V. A. Prisacariu and I. D. Reid. PWP3D: Real-time segmentation and tracking of 3D objects. In *Proceedings of the 20th British Machine Vision Conference*, page Article No. 144, September 2009.

[212] W. Purgathofer, R. F. Tobler, and M. Geiler. Forced random dithering: Improved threshold matrices for ordered dithering. In *Proc. 1st IEEE International Conference on Image Processing*, volume 2, pages 1032–1035, Austin, Texas, Nov. 1994. Also in Graphics Gems 5, p 297–301.

[213] T. Riemersma. A balanced dithering technique. *C/C++ Users Journal*, 16(12):51–58, December 1998.

[214] J. J. Rissanen. Generalized Kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20(3):198–203, 1976.

[215] F. Rosenblatt. Remarks on some nonparametric estimates of a density function. *Annals of Mathematical Statistics*, 27:832–837, September 1956.

[216] A. Rosenfeld and J. L. Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM*, 13(4):471–494, 1966.

[217] B. Rosenhahn, T. Brox, D. Cremers, and H. Seidel. A comparison of shape matching methods for contour based pose estimation. In R. Reulke, U. Eckhardt, B. Flach, U. Knauer, and K. Polthier, editors, *Combinatorial Image Analysis*, volume 4040 of *Lecture Notes in Computer Science*, pages 263–276, Berlin, 2006. Springer.

[218] B. Rosenhahn, T. Brox, D. Cremers, and H. Seidel. Online smoothing for markerless motion capture. In F. Hambrecht, C. Schnörr, and B. Jähne, editors, *Pattern Recognition*, volume 4713 of *lncs*, pages 163–172, Berlin, September 2007. Springer.

[219] B. Rosenhahn, T. Brox, and J. Weickert. Three-dimensional shape knowledge for joint image segmentation and pose tracking. *International Journal of Computer Vision*, 73(3):243–262, July 2007.

[220] B. Rosenhahn, O. Granert, and G. Sommer. Monocular pose estimation of kinematic chains. In L. Dorst, C. Doran, and J. Lasenby, editors, *Applied Geometric Algebras for Computer Science and Engineering*, pages 373–383. Birkhäuser Verlag, 2001.

[221] B. Rosenhahn, H. Ho, and R. Klette. Texture driven pose estimation. In M. Sarfraz, Y. Wand, and E. Banissi, editors, *Proc. of the IEEE International Conference on omputer Graphics, Imaging and Visualization: New Trends*, pages 271–277, Los Alamitos, USA, July 2005. IEEE Computer Society Press.

[222] B. Rosenhahn, C. Schmaltz, T. Brox, J. Weickert, D. Cremers, and H. Seidel. Markerless motion capture of man-machine interaction. In *Proc. 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, June 2008. IEEE Computer Society Press.

[223] B. Rosenhahn, C. Schmaltz, T. Brox, J. Weickert, and H. Seidel. Staying well grounded in markerless motion capture. In G. Rigoll, editor, *Pattern Recognition*, volume 5096 of *Lecture Notes in Computer Science*, pages 385–395, Berlin, June 2008. Springer.

[224] B. Rosenhahn and G. Sommer. Adaptive pose estimation for different corresponding entities. In L. Van Gool, editor, *Pattern Recognition*, volume 2449 of *Lecture Notes in Computer Science*, pages 265–273, Berlin, September 2004. Springer.

[225] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004.

[226] M. Rousson and N. Paragios. Shape priors for level set representations. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Computer Vision – ECCV 2002*, volume 2351 of *Lecture Notes in Computer Science*, pages 78–92. Springer, Berlin, 2002.

[227] A. Roussos and P. Maragos. Vector-valued image interpolation by an anisotropic diffusion-projection PDE. In F. Sgallari, F. Murli, and N. Paragios, editors, *Scale Space and Variational Methods in Computer Vision*, volume 4485 of *Lecture Notes in Computer Science*, pages 104–115. Springer, Berlin, 2007.

[228] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[229] R. Sandhu, S. Dambreville, A. Yezzi, and A. Tannenbaum. Non-rigid 2D-3D pose estimation and 2D image segmentation. In *Proc. 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 786–793, Washington, DC, USA, June 2009. IEEE Computer Society Press.

[230] S. Särkkä, A. Vehtari, and J. Lampinen. Rao-Blackwellized Monte Carlo data association for multiple target tracking. In *Proc. 7th International Conference on Information Fusion*, volume 1, pages 583–590, 2004.

[231] K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann, San Francisco, third edition, 2005.

[232] C. Schmaltz. Diffusion based image compression using rectangular grids. Master's thesis, Deptartment of Mathematics, Saarland University, 2009.

[233] C. Schmaltz, P. Gwosdek, A. Bruhn, and J. Weickert. Electrostatic halftoning. *Computer Graphics Forum*, 29(8):2313–2327, Dec. 2010.

[234] C. Schmaltz, P. Gwosdek, and J. Weickert. Multi-class electrostatic halftoning, 2012. Submitted to Computer Graphics Forum. Preliminary decision: Probably accept after minor revisions.

[235] C. Schmaltz, B. Rosenhahn, T. Brox, D. Cremers, J. Weickert, L. Wietzke, and G. Sommer. Region-based pose tracking. In J. Martí, J. M. Benedí, A. M. Mendonça, and J. Serrat, editors, *Pattern Recognition and Image Analysis*, volume 4478 of *Lecture Notes in Computer Science*, pages 56–63, Berlin, June 2007. Springer.

[236] C. Schmaltz, B. Rosenhahn, T. Brox, and J. Weickert. Localised mixture models in region-based tracking. In J. Denzler, G. Notni, and H. Süße, editors, *Pattern Recognition*, volume 5748 of *Lecture Notes in Computer Science*, pages 21–30, Berlin, 2009. Springer.

[237] C. Schmaltz, B. Rosenhahn, T. Brox, and J. Weickert. Region-based pose tracking with occlusions using 3D models. *Machine Vision and Applications*, 2011. Online First.

[238] C. Schmaltz, B. Rosenhahn, T. Brox, J. Weickert, D. Cremers, L. Wietzke, and G. Sommer. Occlusion modeling by tracking multiple objects. In F. Hambrecht, C. Schnörr, and B. Jähne, editors, *Pattern Recognition*, volume 4713 of *Lecture Notes in Computer Science*, pages 173–183, Berlin, September 2007. Springer.

[239] C. Schmaltz, B. Rosenhahn, T. Brox, J. Weickert, L. Wietzke, and G. Sommer. Dealing with self-occlusion in region based motion capture by means of internal regions. In F. J. Perales and R. B. Fisher, editors, *Articulated Motion and Deformable Objects*, volume 5098 of *Lecture Notes in Computer Science*, pages 102–111, Berlin, July 2008. Springer.

[240] C. Schmaltz, J. Weickert, and A. Bruhn. Beating the quality of JPEG 2000 with anisotropic diffusion. In J. Denzler, G. Notni, and H. Süße, editors, *Pattern Recognition*, volume 5748 of *Lecture Notes in Computer Science*, pages 452–461, Berlin, 2009. Springer.

[241] A. Secord. Weighted Voronoi stippling. In *Proc. Second International Symposium on Non-photorealistic Animation and Rendering*, pages 37–43, New York, NY, USA, 2002. ACM.

[242] J. Selig. Lie groups and lie algebras in robotics. In J. Byrnes, editor, *Computational Noncommutative Algebra and Applications*, volume 136 of *NATO Science Series II: Mathematics, Physics and Chemistry*, pages 101–125, New York, Boston, Dordrecht, London, Moscow, 2005. Kluwer Academic Publishers.

[243] J. Seward. bzip2 and libbzip2, version 1.0.5, a program and library for data compression. http://www.bzip.org/., Mar. 2008. Documentation.

[244] J. Shade, M. F. Cohen, and D. P. Mitchell. Tiling layered depth images. Technical Report CSE 02-12-07, University of Washington, Washington, 2002.

[245] F. Shevlin. Analysis of orientation problems using Plücker lines. In *Proc. 14th International Conference on Pattern Recognition*, volume 1, pages 685–689, Washington, DC, USA, August 1998. IEEE Computer Society Press.

[246] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Proc. Twelfth International Conference on Computer Vision*. IEEE Computer Society Press, 2008.

[247] D. Shreiner, M. Woo, J. Neider, and T. Davis. *OpenGL programming guide*. Addison-Wesley, Upper Saddle River, 5th ed. edition, 2006.

[248] H. Sidenbladh, M. J. Black, and L. Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Computer Vision – ECCV 2002*, volume 2350 of *Lecture Notes in Computer Science*, pages 784–800. Springer, Berlin, 2002.

[249] E. Sifakis, C. Garcia, and G. Tziritas. Bayesian level sets for image segmentation. *Journal of Visual Communication and Image Representation*, 13(1/2):44–64, 2002.

[250] L. Sigal, A. O. Balan, and M. J. Black. HUMANEVA: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87(1/2), Mar. 2010.

[251] L. Sigal and M. J. Black. HumanEva: Synchronized video and motion capture dataset for evaluation of articulated motion. Technical Report CS-06-08, Department of Computer Science, Brown University, September 2006.

[252] L. Sigal, B. Sidharth, S. Roth, M. Black, and M. Isard. Tracking loose-limbed people. In *Proc. 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 421–428. IEEE Computer Society Press, June 2004.

[253] C. Sminchisescu and A. Jepson. Generative modeling for continuous non-linearly embedded visual inference. In C. E. Brodley, editor, *Machine Learning, Proceedings of the Twenty-first International Conference on Machine Learning*, pages 759–766, 2004.

[254] C. Sminchisescu, A. Kanaujia, and D. Metaxas. Learning joint top-down and bottom-up processes for 3D visual inference. In *Proc. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1743–1752, New York, NY, USA, June 2006. IEEE Computer Society Press.

[255] C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. *International Journal of Robotics Research*, 22(6):371–391, June 2003.

[256] R. L. Stevenson and G. R. Arce. Binary display of hexagonally sampled continuous-tone images. *Journal of the Optical Society of America A*, 2(7):1009–1013, Mar. 1985.

[257] D. Stirzaker. *Elementary Probability*. Cambridge University Press, 2nd edition, 2003.

[258] P. Strobach. Quadtree-structured recursive plane decomposition coding of images. *IEEE Transactions on Signal Processing*, 39(6):1380–1397, June 1991.

[259] T. Strutz. *Bilddatenkompression*. Vieweg + Teubner, Braunschweig, fourth edition, 2009.

[260] P. Stucki. MECCA – a multiple-error correcting computation algorithm for bilevel image hardcopy reproduction. Technical Report RZ1060, IBM Research Lab, Zurich, Switzerland, 1981.

[261] E. B. Sudderth, M. I. Mandel, W. T. Freeman, and A. S. Willsky. Distributed occlusion reasoning for tracking with nonparametric belief propagation. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, 17, pages 1369–1376, Cambridge, MA, 2005. MIT Press.

[262] C. Sul, S. Jung, and K. Wohn. Synthesis of human motion using kalman filter. In N. Magnenat-Thalmann and D. Thalmann, editors, *Modelling and Motion Capture Techniques for Virtual Environments*, volume 1537 of *Lecture Notes in Computer Science*, pages 100–112, Berlin, 1998. Springer.

[263] G. J. Sullivan and R. J. Baker. Efficient quadtree coding of images and video. *IEEE Transactions on Image Processing*, 3(3):327–331, May 1994.

[264] G. J. Sullivan and T. Wiegand. Video compression – from concepts to the H.264/AVC standard. *Proceedings of the IEEE*, 93(1):18–31, January 2005.

[265] J. Sun, W. Zhang, X. Tang, and H. Shum. Background cut. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision – ECCV 2006, Part II*, volume 3952 of *Lecture Notes in Computer Science*, pages 628–641, Berlin, May 2006. Springer.

[266] M. Sun, H. Su, S. Savarese, and L. Fei-Fei. A multi-view probabilistic model for 3D object classes. In *Proc. 2009 IEEE Computer Society Conference of Computer Vision and Pattern Recognition (CVPR)*, pages 1247–1254, New York, NY, USA, June 2009. IEEE Computer Society Press.

[267] A. Sundaresan and R. Chellappa. Multicamera tracking of articulated human motion using shape and motion cues. *IEEE Transactions on Image Processing*, 18(9):2114–2126, 2009.

[268] M. Sunkel, B. Rosenhahn, and H. Seidel. Silhouette based generic model adaptation for marker-less motion capturing. In A. Elgammal, B. Rosenhahn, and R. Klette, editors, *Human Motion – Understanding, Modeling, Capture and Animation : Second Workshop, Human Motion 2007*, volume 4814 of *Lecture Notes in Computer Science*, pages 119–135, Rio de Janeiro, Brazil, 2007. Springer.

[269] V. Surazhsky, P. Alliez, and C. Gotsman. Isotropic remeshing of surfaces: A local parameterization approach. In *Proc. 12th International Meshing Roundtable*, pages 215–224, 2003.

[270] D. S. Taubman and M. W. Marcellin, editors. *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Kluwer, Boston, 2002.

[271] L. Taycher, G. Shakhnarovich, D. Demirdjian, and T. Darrell. Conditional random people: Tracking humans with CRFs and grid filters. In *Proc. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 222–229, New York, NY, December 2006. IEEE Computer Society Press.

[272] A. Telea. An image inpainting technique based on the fast marching method. *Journal of graphics, gpu, and game tools*, 9(1):23–34, 2004.

[273] T. Teuber, G. Steidl, P. Gwosdek, C. Schmaltz, and J. Weickert. Dithering by differences of convex functions. *SIAM Journal on Imaging Sciences*, 4(1):79–108, 2011.

[274] S. Toelg and T. Poggio. Towards an example-based image compression architecture for video-conferencing. Technical Report AIM-1494, Massachusetts Institute of Technology, Cambridge, MA, USA, 1994.

[275] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[276] D. Tschumperlé. Fast anisotropic smoothing of multi-valued images using curvature-preserving PDE's. *International Journal of Computer Vision*, 68(1):65–82, June 2006.

[277] D. Tschumperlé (Project Leader). The CImg library. http://cimg.sourceforge.net/, 2010. Last visited August 27,2010.

[278] H. Tsuji, T. Sakatani, Y. Yashima, and N. Kobayashi. A nonlinear spatio-temporal diffusion and its application to prefiltering in MPEG-4 video coding. In *Proc. 2002 IEEE International Conference on Image Processing*, volume 1, pages 85–88, Rochester, NY, Sept. 2002.

[279] A. Ude and C. G. Atkeson. Online tracking and mimicking of human movements by a humanoid robot. *Journal of Advanced Robotics*, 17(2):165–178, 2003.

[280] R. A. Ulichney. Dithering with blue noise. *Proceedings of the IEEE*, 76(1):56–79, Jan. 1988.

[281] R. Urtasun, D. Fleet, and P. Fua. 3D people tracking with Gaussian process dynamical models. In *Proc. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 238–245, New York, September 2006. IEEE Computer Society Press.

[282] R. van den Boomgaard. The morphological equivalent of the Gauss convolution. *Nieuw Archief Voor Wiskunde*, 10(3):219–236, Nov. 1992.

[283] D. Vanderhaeghe and V. Ostromoukhov. Polyomino-based digital halftoning. In P. Isaías, editor, *IADIS International Conference on Computer Graphics and Visualization 2008*, pages 11–18, July 2008.

[284] W. E. Vieux, K. Schwerdt, and J. L. Crowley. Face-tracking and coding for video compression. In H. I. Christensen, editor, *Computer Vision Systems*, number 1542 in Lecture Notes in Computer Science, pages 151–161, Berlin, 1999. Springer.

[285] M. Vondrak, S. L, and O. C. Jenkins. Physical simulation for probabilistic motion tracking. In *Proc. 2008 IEEE Computer Society Conference of Computer Vision and Pattern Recognition (CVPR)*, New York, NY, USA, June 2008. IEEE Computer Society Press.

[286] L. Wei. Multi-class blue noise sampling. *ACM Transactions on Graphics*, 29(4):Article No. 79, 2010.

[287] J. Weickert. Theoretical foundations of anisotropic diffusion in image processing. *Computing Supplement*, 11:221–236, 1996.

[288] J. Weickert and T. Brox. Diffusion and regularization of vector- and matrix-valued images. In M. Z. Nashed and O. Scherzer, editors, *Inverse Problems, Image Analysis, and Medical Imaging*, volume 313 of *Contemporary Mathematics*, pages 251–268. AMS, Providence, 2002.

[289] J. Weickert, S. Ishikawa, and A. Imiya. Linear scale-space has first been proposed in Japan. *Journal of Mathematical Imaging and Vision*, 10(3):237–252, May 1999.

[290] J. Weickert and M. Welk. Tensor field interpolation with PDEs. In J. Weickert and H. Hagen, editors, *Visualization and Processing of Tensor Fields*, pages 315–325. Springer, Berlin, 2006.

[291] T. A. Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, 1984.

[292] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits, Systems and Video Technology*, 13(7):560–576, 2003.

[293] L. Wietzke, G. Sommer, O. Fleischmann, and C. Schmaltz. The conformal monogenic signal of image sequences. In D. Cremers, B. Rosenhahn, A. L. Yuille, and F. R. Schmidt, editors, *Visual Motion Analysis*, volume 5604 of *Lecture Notes in Computer Science*, pages 305–322, Berlin, 2009. Springer.

[294] L. Wietzke, G. Sommer, C. Schmaltz, and J. Weickert. Analysis of the curvature tensor from the viewpoint of signal processing. In *Selected Papers from ICNAAM-2007 and ICCMSE-2007*, volume 1046 of *AIP Conference Proceedings*, pages 150–153, Corfu, Greece, September 2008.

[295] L. Wietzke, G. Sommer, C. Schmaltz, and J. Weickert. Differential geometry of monogenic signal representations. In G. Sommer and R. Klette, editors, *Robot Vision*, volume 4931 of *Lecture Notes in Computer Science*, pages 454–465, Auckland, New Zealand, February 2008. Springer.

[296] E. R. Williams, J. E. Faller, and H. A. Hill. New experimental test of coulomb's law: A laboratory upper limit on the photon rest mass. *Physical Review Letters*, 26(12721–724), Mar. 1971.

[297] A. K. C. Wong and M. You. Entropy and distance of random graphs with application of structural pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(5):599–609, May 1985.

[298] S. Yang and Y. Hu. Coding artifact removal using biased anisotropic diffusion. In *Proc. 1997 IEEE International Conference on Image Processing*, volume 2, pages 346–349, Santa Barbara, CA, Oct. 1997.

[299] S. Yao, W. Lin, Z. Lu, E. Ong, and X. Yang. Adaptive nonlinear diffusion processes for ringing artifacts removal on JPEG 2000 images. In *Proc. 2004 IEEE International Conference on Multimedia and Expo*, pages 691–694, Taipei, Taiwan, June 2004.

[300] Z. Yao. *Model-based Coding – Initialization, Parameters Extraction and Evaluation*. PhD thesis, Department of Applied Physics and Electronics, Umeå University, Sweden, January 2005.

[301] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Journal of Computing Surveys*, 38(4):Article No. 13, 2006.

[302] Youtube fact sheet. http://www.youtube.com/t/fact_sheet. Last visited August 23, 2010.

[303] D. Zang, L. Wietzke, C. Schmaltz, and G. Sommer. Dense optical flow estimation from the monogenic curvature tensor. In F. Sgallari, F. Murli, and N. Paragios, editors, *Scale Space and Variational Methods in Computer Vision*, volume 4485 of *Lecture Notes in Computer Science*, pages 239–250, Berlin, May/June 2007. Springer.

[304] R. Zhang, P. Tsai, J. Cryer, and M. Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, August 1999.

[305] H. Zhao, T. Chan, B. Merriman, and S. Osher. A variational level set approach to multiphase motion. *Journal of Computational Physics*, 127:179–195, 1996.

[306] B. Zhou and X. Fang. Improving mid-tone quality of variable-coefficient error diffusion using threshold modulation. *ACM Transactions on Graphics*, 22(3):437–444, 2003.

[307] S. Zhu and A. Yuille. Region competition: unifying snakes, region growing, and Bayes/MDL for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):884–900, Sept. 1996.