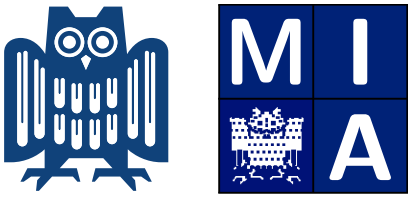
Variational Surface Reconstruction

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

vorgelegt von

Christopher Schroers

Saarbrücken, 2016



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

Tag des Kolloquiums

08.07.2016

Dekan

Prof. Dr. Frank-Olaf Schreyer

Prüfungsausschuss

Prof. Dr. Hans-Peter Seidel (Vorsitzender)

Prof. Dr. Joachim Weickert (1. Gutachter)

Prof. Dr. Thorsten Thormählen (2. Gutachter)
Philipps-Universität Marburg

Dr. Peter Ochs (akademischer Mitarbeiter)

Copyright

Kurzzusammenfassung

In den letzten Jahrzehnten ist die Nachfrage nach digitalen 3D Modellen von Objekten und Szenen ständig gestiegen und vieles spricht dafür, dass sich dies auch in Zukunft fortsetzt: Computergenerierte Spezialeffekte werden immer flächendeckender eingesetzt, der Druck von dreidimensionalen Gegenständen macht große Fortschritte, und die Darstellung dreidimensionaler Modelle im Webbrowser wird immer ausgereifter. Deshalb ist die 3D Rekonstruktion eines der wichtigsten Forschungsthemen im Bereich des maschinellen Sehens. Die Rekonstruktion von einem 3D Modell aus mehreren Bildern mit gegebenen Kameramatritzen ist hier eine der häufigsten Problemstellungen, bekannt als multi-view stereo. Wir leisten einen Beitrag zu den zwei wichtigen Schritten, die in multi-view stereo Ansätzen angewandt werden: Die Schätzung von Tiefenkarten aus mehreren Bildern und die Fusion von mehreren Tiefenkarten zu einem einzigen 3D Modell. Anschließend lockern wir die Voraussetzung, dass die Kameramatritzen bekannt sein müssen und präsentieren ein neues Verfahren zur 3D Rekonstruktion aus Bildsequenzen, das vollständig auf dichten Ansätzen beruht. Dies erweist sich als interessante Alternative zu populären Methoden, die mit einzelnen Merkmalen arbeiten. Verfahren, die auf einzelnen Merkmalen beruhen, erlauben die Schätzung von orientierten Punktwolken. Daher entwickeln wir zum Schluss ein allgemeines Rahmenwerk für die Berechnung von wasserdichten Oberflächen aus orientierten Punktwolken.

Short Abstract

The demand for capturing 3D models of real world objects or scenes has steadily increased in the past. Today, there are numerous developments that indicate an even greater importance in the future: Computer generated special effects are extensively used and highly benefit from such data, 3D printing is starting to become more affordable, and the ability to conveniently include 3D content in websites has quite matured. Thus, 3D reconstruction has been and still is one of the most important research topics in the area of computer vision. Here, the reconstruction of a 3D model from a number of colour images with given camera poses is one of the most common tasks known as multi-view stereo. We contribute to the two main stages that arise in popular strategies for solving this problem: The estimation of depth maps from multiple views and the integration of multiple depth maps into a single watertight surface. Subsequently, we relax the constraint that the camera poses have to be known and present a novel pipeline for 3D reconstruction from image sequences that solely relies on dense ideas. It proves to be an interesting alternative to popular sparse approaches and leads to competitive results. When relying on sparse features, this only allows to estimate an oriented point cloud instead of a surface. To this end, we finally propose a general higher order framework for the surface reconstruction from oriented points.

Zusammenfassung

In den letzten Jahrzehnten ist die Nachfrage nach digitalen 3D Modellen von Objekten und Szenen ständig gestiegen und vieles spricht dafür, dass sich dies auch in Zukunft fortsetzt: Computergenerierte Spezialeffekte werden immer flächendeckender eingesetzt, der Druck von dreidimensionalen Gegenständen macht große Fortschritte, und die Darstellung dreidimensionaler Modelle im Webbrowser wird immer ausgereifter. Außerdem könnten Anwendungem im Bereich der Augmented Reality beispielsweise durch Geräte wie die HoloLens Einzug in das alltägliche Leben nehmen und Türen für eine Vielzahl neuer Anwendungen öffnen. Deshalb ist die 3D Rekonstruktion eines der wichtigsten Forschungsthemen im Bereich des maschinellen Sehens. Dabei ist die Rekonstruktion von einem 3D Modell aus mehreren Bildern mit gegebenen Kameramatritzen eine der häufigsten Problemstellungen, bekannt als multi-view stereo.

Wir leisten einen Beitrag zu zwei wichtigen Schritten, die in Methoden zur Lösung dieses Problems angewandt werden: Die Schätzung von Tiefenkarten aus mehreren Bildern und die Fusion von mehreren Tiefenkarten zu einem einzigen 3D Modell. Bezüglich des ersten Schrittes generalisieren wir bestehende Methoden durch Einführung einer Parameterisierung der Tiefe. Dies erlaubt eine systematische Analyse verschiedener Parameterisierungen, so dass wir sowohl theoretisch also auch praktisch die Vorzüge der inversen Tiefe in Bezug auf den Datenterm und den Glattheitsterm herausstellen können. Bezüglich des zweiten Schrittes erweitern wir einen weit verbreiteten volumetrischen Ansatz durch Nutzung von anisotroper (richtungsabhängiger) Regularisierung. Außerdem arbeiten wir mit der Euklidschen Distanz anstelle der Distanz entlang des Sichtstrahls und präsentieren eine effiziente Implementierung auf einer Grafikkarte, die ein neues zyklisches Verfahren namens Fast Jacobi nutzt.

Während die vorherige Problemstellung davon ausgeht, dass die Kameramatritzen bekannt sind, lockern wir diese Voraussetzung im Folgenden und präsentieren ein neues Verfahren zur 3D Rekonstruktion aus Bildsequenzen, das vollständig auf dichten Ansätzen beruht. Jeder einzelne Schritt minimiert ein geeignetes Energiefunktional und sorgt so für transparente Modellannahmen. Der vorgeschlagene Ansatz erweist sich als interessante Alternative zu populären Methoden, die mit einzelnen Merkmalen arbeiten und führt zu sehr konkurrenzfähigen Ergebnissen.

Verfahren, die auf einzelnen Merkmalen beruhen, erlauben die Schätzung von orientierten Punktwolken. Daher entwickeln wir zum Schluss ein allgemeines Rahmenwerk höherer Ordnung für die Berechnung von wasserdichten Oberflächen aus orientierten Punktwolken. Hierdurch erreichen wir zwei Zie-

le: Zum einen können wir mehrere weit verbreitete Verfahren systematisch verstehen und klassifizieren. Zum anderen können wir neue Methoden herleiten, die relativ einfach sind und trotzdem Ergebnisse höchster Qualität liefern.

Abstract

The demand for capturing 3D models of real world objects or scenes has steadily increased in the past. Today, there are numerous developments that indicate an even greater importance in the future: Computer generated special effects are extensively used and highly benefit from such data, 3D printing is starting to become more affordable, and the ability to conveniently include 3D content in websites has quite matured. Last but not least, augmented reality is on the rise and might find a way into everyday life in the near future through devices such as the HoloLens for example. This might open up various new applications that require knowledge about the geometry of real world objects or scenes. Thus 3D reconstruction has been and still is one of the most important research topics in the area of computer vision. Here, the reconstruction of a 3D model from a number of colour images with given camera poses is a common task known as multi-view stereo. We contribute to the two main steps that arise in popular strategies for solving this problem: The estimation of depth maps from multiple views and the integration of multiple depth maps into a single watertight surface. Concerning the former one, we generalise existing methods by introducing a parameterisation of depth. This allows to thoroughly analyse different choices and to theoretically and practically point out the benefits of an inverse depth parameterisation w.r.t. both the data and the smoothness term. Concerning the second step, we extend a popular volumetric approach by using anisotropic (direction-dependent) regularisation and relying on the Euclidean instead of the directional signed distance. Furthermore, we present an efficient parallel implementation on the GPU that uses a recently introduced cyclic solver named Fast Jacobi.

While these two steps allow to obtain reconstructions in a multi-view stereo setting, we subsequently relax the constraint that the camera poses have to be known. To this end, we present a novel pipeline for 3D reconstruction from image sequences that solely relies on dense ideas. Each step minimises a suitable energy functional such that the modelling choices are transparent throughout the whole pipeline. This proves to be an interesting alternative to popular sparse approaches and leads to competitive results. When relying on sparse features, this only allows to estimate an oriented point cloud instead of a surface. Thus we finally propose a general higher order framework for the surface reconstruction from oriented points. This framework allows us to reach two goals: First, we can systematically understand and classify a number of existing methods. Second, it enables us to derive novel approaches to surface reconstruction from oriented points that are fairly simple and offer state-of-the-art performance.

Acknowledgements

First, I would like to thank Prof. Joachim Weickert for letting me join his Mathematical Image Analysis (MIA) group and for supervising me throughout my dissertation. I am also grateful to Prof. Thorsten Thormählen for reviewing my thesis and to Prof. Hans-Peter Seidel for being the chair of the committee. I appreciate that Dr. Peter Ochs agreed to be the academic assistant.

Writing a PhD thesis is a large scale project which spans several years. To me, it was a very rewarding endeavor, especially since I was fortunate enough to be in an amazing environment with great colleagues. Therefore, I would like to begin by thanking my close collaborators and coauthors Dr. Oliver Demetz, Dr. Sven Grewenig, David Hafner, Nico Persch, Dr. Simon Setzer, and Timm Schneevoigt. Furthermore, I thank Dr. Henning Zimmer, Dr. Levi Valgaerts, and Prof. Andres Bruhn for their advice towards the beginning of my thesis and I am also grateful to Sebastian Hoffmann, Martin Schmidt, Dr. Pascal Peter, and Laurent Hoeltgen for fruitful discussions. I am glad that I was still able to meet Marcelo Cardenas, Sarah Schäffer, and Sabine Müller in the final stage of my thesis. In fact, I would like to thank all current and former members of the MIA group, especially Dr. Pascal Gwosdek and Marcus Hargarter for joining us on our skiing trips. I also owe my thanks to our secretary Ms. Ellen Wintringer who always perfectly cared for us.

I am very grateful that I was given the chance to pursue an internship with Disney Research Zurich. Although it has not been integrated into this thesis, I still regard it as an essential experience within my PhD studies. I am extremely happy that I had the chance to work with Dr. Oliver Wang, Dr. Alexander Sorkine-Hornung, and Dr. Henning Zimmer and that I could meet Changil Kim, Federico Perazzi, Kaan Yücer and many more.

Finally, I cannot thank my family enough for their love and for their unconditional and endless support. I truly feel blessed to have my parents Rolf and Ingrid, my brother Robert, and my grandmother Christine, and I am deeply sad that my grandfather Albert has passed away. They have always stood by me and encouraged me my whole life. Last but certainly not least, I thank my girlfriend for her warmth and love. Ulli, thanks for letting me go to Saarbrücken and for bearing with me while I was gone for such a long time. I am incredibly fortunate to have you in my life.

Contents

Chapter 1

Introduction

1.1 Motivation

Over the last decades, the demand for digitally capturing the geometry of real world objects and scenes has steadily increased. The fact that computer generated special effects which greatly benefit from such data have become more and more widely used is one reason for this. However, there are numerous other developments that highlight a trend towards relying on 3D models which further underline the importance of digitally reconstructing the surface of real world objects: 3D printing is starting to become more affordable and is being used for manufacturing individual pieces that can contain real world reconstructions. Furthermore, the ability to conveniently include 3D content in websites has quite matured. For example XML3D allows to seamlessly integrate 3D content into HTML pages. Last but not least, augmented reality is on the rise and might find a way into everyday life in the near future through devices such as the HoloLens for example. This might open up various new applications that require knowledge about the geometry of real world objects or scenes. Under these considerations, it is not surprising that 3D reconstruction has been and still is a very important research topic in the area of computer vision and that there is a great variety of methods that deals with the reconstruction of surfaces such as [124, 63, 112, 40, 21, 58, 96].

This thesis aims at advancing the state-of-the-art in surface reconstruction both in theory and practice. To this end, we do not only follow popular 3D reconstruction pipelines and contribute to the important individual steps. We also propose a novel 3D reconstruction pipeline that consequently relies on dense approaches.

1.2 Scope and Overview

The scope of this thesis is the reconstruction of watertight 3D models of static objects or scenes. In order to compute such a 3D reconstruction, some kind of input data has to be recorded at first. Although light field cameras and time-of-flight cameras are becoming more readily available, nowadays standard digital cameras are still by far the most prominent device for capturing reality. Therefore, we use colour images as a starting point when considering the problem of 3D reconstruction. When further assuming that the colour images have been acquired from cameras with known poses, the problem scenario is referred to as *multi-view stereo*. Here, a very popular approach is given by first computing depth maps from the images and subsequently merging these depth maps into a unified 3D model. Such a modular formulation with two stages incorporates the inherent advantage that one is not restricted to working with colour images but can also directly use depth information if it is available by simply skipping the first stage. Computing 3D reconstructions from images without known camera poses also belongs into the scope of this thesis. Such problems are often referred to as *simultaneous localisation and mapping* (SLAM) or *structure from motion* (SfM). Last but not least, oriented point clouds resemble another common form of input data that has to be considered. In all cases, we consequently rely on variational models to allow for transparent modeling assumptions within a clean mathematical framework.

Organisation of this thesis. After covering important foundations in Chapter 2, we deal with the estimation of depth maps from multiple fully calibrated views in Chapter 3. Then Chapter 4 tackles the problem of merging multiple depth maps into a single 3D model. Subsequently, we relax the assumption that the camera poses have to be known in advance and propose a dense pipeline that can reconstruct objects given an image sequence and intrinsic camera parameters only in Chapter 5. While the previous approach is formulated in a fully dense manner, there are reconstruction pipelines that work with sparse features to create an oriented point cloud from multiple images [40]. Taking this output as a starting point results in the problem of reconstructing a surface given a finite number of oriented points. We consider this problem in Chapter 6 before we conclude in Chapter 7.

We now give a more detailed overview of the problems that we consider in the individual chapters and point out our contributions.

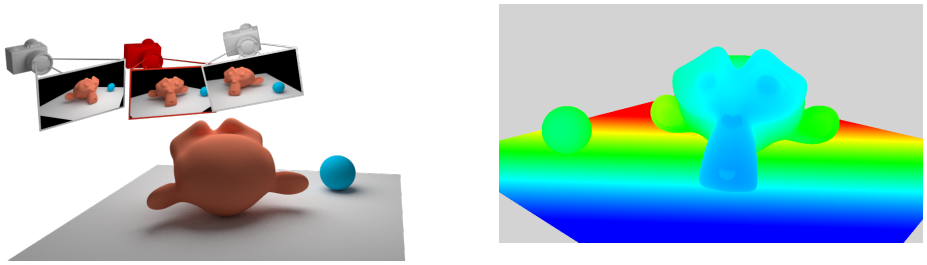


Figure 1.1: In multi-view depth estimation a number of images captured from a static scene (left) is used to estimate a single depth map (right) as seen from one selected reference camera which is shown in red. The depth map is visualised with a colour coding where blue is near and red is far.

Multi-View Depth Estimation. Here we consider variational methods that estimate a single depth map from multiple views assuming that the camera matrices are known. This scenario is illustrated in Figure 1.1 and we refer to it as *multi-view depth estimation*. Previous work has demonstrated that multiple views improve the depth reconstruction [88, 105, 96], and that higher order regularisers model a good prior for typical real-world scenes [86, 35, 52, 87]. While most existing variational multi-view formulations are formulated in terms of depth, we generalise them by introducing a parameterisation of depth. This allows us to efficiently analyse advantages and drawbacks of different parameterisations such as a direct depth parameterisation and an inverse depth parameterisation. More specifically, we analyse two important aspects: On the one hand, the choice of parameterisation is important when considering the linearisation of the data term in a variational framework. Here, we show that for common camera setups, the inverse depth parameterisation is preferable. On the other hand, the choice of parameterisation is also important in the smoothness term, especially in the presence of second order regularisation. Here, we show that for an inverse depth parameterisation, piecewise affine functions correspond to piecewise planar surfaces. This is not the case for a direct depth parameterisation. We give deep insights into the introduced bias by analysing the shape operator of the corresponding 3D surface. This work has been published in [8].

Surface Reconstruction from Depth Maps. While we previously dealt with computing a single depth map from multiple images, we now aim at combining multiple of such depth maps into a single 3D model as shown in Figure 1.2. During the last few years, this topic has attracted an increasing

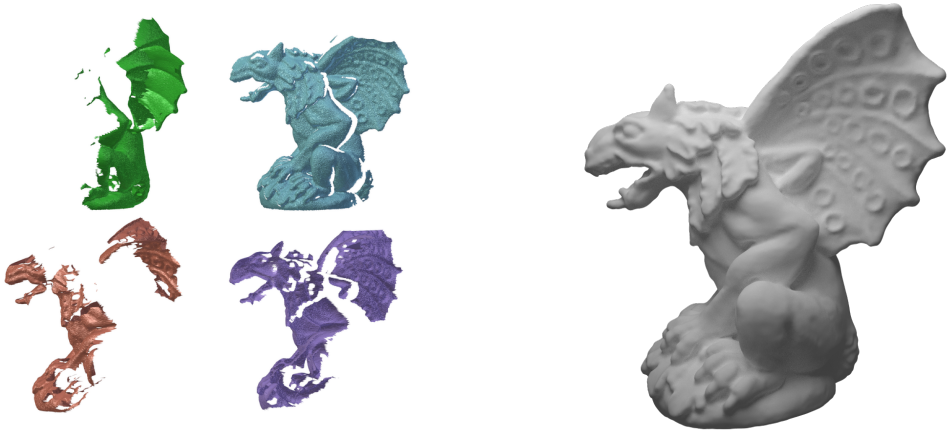


Figure 1.2: Surface reconstruction from depth maps aims at combining multiple registered depth maps (left) into a single 3D model (right). Here the depth maps are visualised in terms of the surface that they describe.

amount of attention. Surely, this is mainly due to the fact that range images are becoming more readily available through devices such as the Kinect or time-of-flight cameras [79]. The task of merging numerous depth maps into a single 3D model can pose many difficulties for several reasons: The depth maps may contain noise and outliers, parts of the surface can be missing when they have not been properly reached during the acquisition, and the sampling density might not be sufficient for a correct reconstruction.

A very promising approach to cope with these problems is given by variational techniques as proposed by Zach et al. [124]. They are able to deal with a substantial amount of noise and outliers, while regularising and thus creating smooth surfaces at the same time. We extend their variational range image integration approach in several aspects: The isotropic (space-variant) diffusion term is replaced by an anisotropic (direction-dependent) one, which is designed to smooth along the evolving surface and evolving ridges in the cumulative signed distance field but not across. This way it is possible to obtain very smooth surfaces from noisy range images while preserving ridges and corners. Furthermore, we do not use signed distances along the line of sight when converting range images into 3D distance fields. Instead, we compute the Euclidean signed distance to the range surface. Last but not least, we employ the novel nonstandard discretisation described in [118] extended to three dimensions and the recently introduced numerical solver called Fast Jacobi [10]. This allows for a fast parallel implementation which we experimentally demonstrate by means of a graphics processing unit (GPU) using

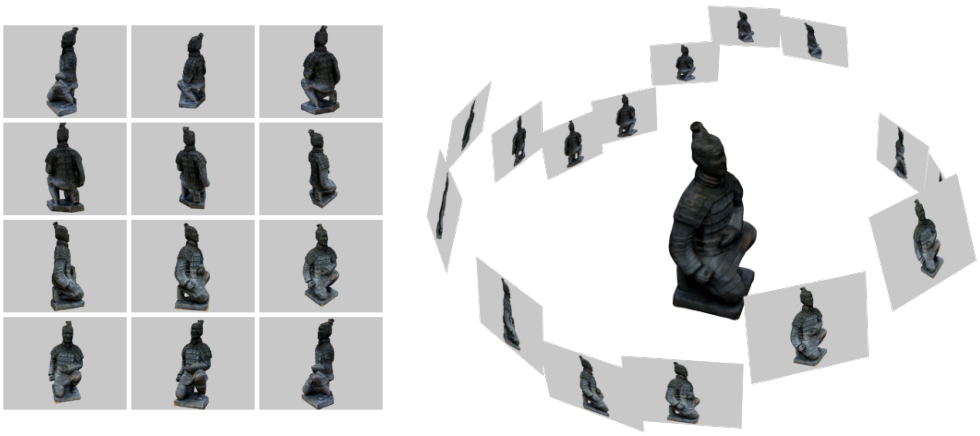


Figure 1.3: Given a number of input colour images captured by a camera following a continuous path along with its intrinsic camera parameters (left), the goal is to estimate the camera motion and a 3D reconstruction of the scene or object (right).

CUDA. Altogether, these changes allow for state-of-the-art results in the Middlebury benchmark at a competitive runtime. The main ideas of our work have been published in [6]. Our efficient GPU implementation is described in [10] and was presented at the NVIDIA GPU Technology conference 2014.

Surface Reconstruction from Image Sequences. Previously, we have assumed knowledge of the camera matrix to create a 3D reconstruction. Here we relax this requirement and consider the problem of estimating reconstructions from image sequences without explicit knowledge of camera poses; cf. Figure 1.3. Many of the existing algorithms for this problem rely on *sparse* features. Such methods have to select the most appropriate data carefully and eliminate outliers. On the other hand, *dense* computer vision methods have made enormous progress in the last decade. For optical flow computation they belong to the leading approaches; see e.g. [13]. Moreover, dense strategies can also be on par with sparse methods for other problems such as the estimation of the fundamental matrix [109]. They do not have to put effort into selecting the best data but rather draw their robustness from using all data.

Motivated by these achievements, we propose a novel pipeline for 3D reconstruction from image sequences that solely relies on dense methods. As

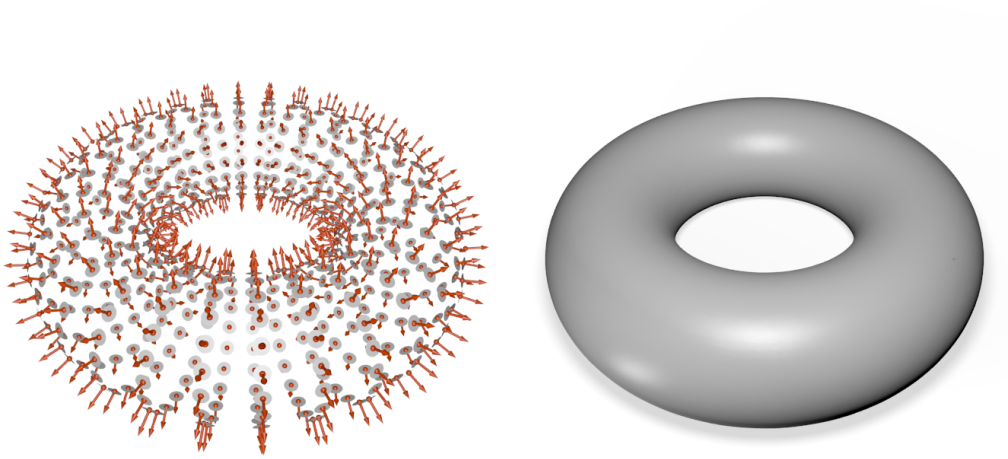


Figure 1.4: Given a finite number of oriented points (left), the goal is to estimate a watertight 3D model (right).

input we require a sequence of colour images capturing a static scene while following a continuous path. Furthermore, we only assume that an intrinsic camera calibration is known.

The pipeline can then be described in three steps: (1) First, we jointly estimate correspondences and stereo geometry for each two consecutive images. (2) Subsequently, we connect the individual pairwise estimates and globally refine them through bundle adjustment. As a result, all camera poses are merged into a consistent global model. This allows us to create accurate depth maps. (3) Finally, these depth maps are merged using variational range image integration techniques. Each of the three main steps minimises a suitable energy functional such that the modelling choices are transparent throughout the whole pipeline. Experiments show that our dense pipeline is an interesting alternative to sparse approaches. It yields accurate camera poses as well as 3D reconstructions. The main ideas of this chapter have been published in [5].

Surface Reconstruction from Oriented Points. An oriented point of a surface contains information about the position and the surface normal. Oriented point clouds can be obtained in numerous ways. For instance with active methods such as laser, structured light and time-of-flight scanning but also through passive methods within a 3D reconstruction pipeline [40]. Therefore, we also consider the problem of reconstructing a watertight 3D

model from a finite set of oriented points which is illustrated in Figure 1.4. It is a common practice to fit the oriented points using a level set of an implicit function. Such methods can produce approximating surfaces, which is preferable if noise and outliers are present. A great variety of different approaches exists [82, 114, 59, 74, 83, 21, 58], and commonly the implicit function is either an approximation of the indicator function or the signed distance function of the underlying surface.

We develop a general higher order variational framework for surface reconstruction. It is based on the idea that each oriented point allows us to construct a function that provides a good local description of an implicit representation of the unknown surface. With this framework, we are able to reach two goals: First, we can systematically understand and classify a number of existing methods. Second, it enables us to derive novel approaches to surface reconstruction that are fairly simple and offer state-of-the-art performance. We show with the recent reconstruction benchmark of Berger et al. [15] that our novel approach yields favourable results when compared to the most popular and widely used methods, namely (screened) Poisson surface reconstruction [59, 58] and smooth signed distance surface reconstruction [21]. Furthermore, we introduce a hull constraint that encourages the surface to stay within a given region. This improves reconstructions in difficult real world scenarios where point clouds have been estimated from colour images. Our framework is implemented on the GPU using a recent cyclic scheme called Fast Jacobi, which combines low implementational effort with high efficiency. This work has been published in [7].

Chapter 2

Foundations

This chapter deals with basic knowledge that is important throughout the whole thesis. It covers camera geometry as well as the calculus of variations and iterative approaches to numerically solve the resulting systems of equations. While we merely focus on the aspects required in this thesis, camera geometry is extensively studied by Hartley and Zisserman [49]. For exhaustive coverage of the calculus of variations the work of Gelfand and Fomin [42] is a good resource. Young [122] offers a thorough introduction to iterative solvers.

2.1 Pinhole Camera Model

The pinhole camera model shown in Figure 2.1 allows to explain the mapping from 3D world coordinates to the 2D image plane. For now, let us assume that the camera is aligned with the world coordinate system, i.e. the centre of the projection $\boldsymbol{c}$ corresponds to the origin of the Euclidean world coordinate system and the Z-axis is perpendicular to the image plane. Then the Z-axis is called *optical axis* and intersects the image plane in the *principal point* $\boldsymbol{p}$. The principal point describes the origin of the coordinate system of the 2D image. The distance between camera centre $\boldsymbol{c}$ and image plane is called *focal length* and is further on denoted by f. The theorem of intersecting lines yields the relationships

$$\frac{x}{X} = \frac{f}{Z} \quad \text{and} \quad \frac{y}{Y} = \frac{f}{Z}. \tag{2.1}$$

This leads to the nonlinear mapping

$$\begin{pmatrix} x \\ y \end{pmatrix} = f \cdot \begin{pmatrix} X/Z \\ Y/Z \end{pmatrix} \tag{2.2}$$

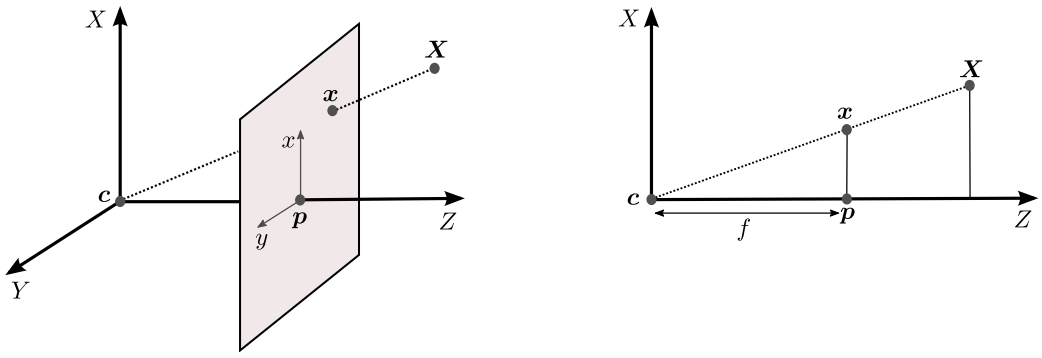


Figure 2.1: Pinhole camera geometry for the case that the Z-axis is perpendicular to the image plane and the camera centre $\boldsymbol{c}$ corresponds to the origin of the world coordinate system. **From left to right**: **(a)** Overview. **(b)** Cross section in X-Z-plane.

from a 3D point $\boldsymbol{X} = (X, Y, Z)^\top$ to the corresponding point $\boldsymbol{x} = (x, y)^\top$ in the image plane. However, when using the concept of the projective space, this nonlinear mapping can be expressed as a linear one with homogeneous coordinates:

$$\tilde{\boldsymbol{x}} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \underbrace{\operatorname{diag}(f, f, 1) \begin{pmatrix} \boldsymbol{I}_3 & \boldsymbol{0}_3 \end{pmatrix}}_{\boldsymbol{P}} \tilde{\boldsymbol{X}}. \tag{2.3}$$

Here, $\boldsymbol{I}_3$ denotes the 3×3 identity matrix and $\boldsymbol{0}_3 = (0, 0, 0)^\top$ the zero vector. The tilde is used to indicate homogeneous quantities. The 3×4 matrix $\boldsymbol{P}$ is the *camera projection matrix* or *camera matrix*.

Let us now derive a more general form of the camera matrix that does not make the previously imposed assumptions that camera centre and world coordinate origin coincide and that the Z-axis is perpendicular to the image plane.

2.1.1 Intrinsic Parameters

In a more general model, it is required to allow for a principal point offset, i.e. a translation of the image plane by x_0 and y_0 as well as a scaling by s_x and s_y in both axial directions. These transformations correspond to the internal characteristics of a physical camera device, and therefore they are called *intrinsic camera parameters*. They can be encoded by the upper

triangular matrix

$$\boldsymbol{K} = \begin{pmatrix} fs_x & 0 & x_0 \\ 0 & fs_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.4}$$

where $fs_x =: k_x$ and $fs_y =: k_y$ describe the non uniform scaling that is usually related to the pixel size. Sometimes a shearing is considered as well, in which case $\boldsymbol{K}$ possesses 5 degrees of freedom but for typical cameras that we consider the shearing is zero. If all entries of $\boldsymbol{K}$ are known, the camera is said to be *internally calibrated*. The camera is said to be *fully calibrated* if the intrinsic and extrinsic parameters are known.

2.1.2 Extrinsic Parameters

The *extrinsic camera parameters* describe the camera pose relative to the world coordinate system. In the case that only a single camera is considered, one can choose camera coordinate system and world coordinate system to coincide, as it was done in Figure 2.1 for simplicity. However, if several different cameras are involved this is only possible for a single camera. *Translation* and *rotation* of the others have to be taken into account with respect to the world coordinate system. The mapping between world coordinate system and camera coordinate system can be expressed as

$$\boldsymbol{X}_c = \boldsymbol{R}(\boldsymbol{X} - \boldsymbol{c}) = \boldsymbol{R}\boldsymbol{X} - \boldsymbol{R}\boldsymbol{c} = \boldsymbol{R}\boldsymbol{X} + \boldsymbol{t}, \tag{2.5}$$

where the coordinates of a world point $\boldsymbol{X}$ with respect to the camera coordinate system are denoted by $\boldsymbol{X}_c$. The 3×3 rotation matrix $\boldsymbol{R}$ combined with the translation $\boldsymbol{t} = -\boldsymbol{R}\boldsymbol{c}$ align camera and world coordinate system, and $\boldsymbol{c}$ denotes the camera centre specified in the world coordinate system. The extrinsic parameters offer 6 degrees of freedom of which 3 arise from the rotation and 3 from the translation. With homogeneous coordinates, the mapping from world to camera coordinate system can be rewritten:

$$\tilde{\boldsymbol{X}}_c = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{0} \\ \boldsymbol{0}^\top & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{X} \\ 1 \end{pmatrix} + \begin{pmatrix} \boldsymbol{t} \\ 0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^\top & 1 \end{pmatrix} \tilde{\boldsymbol{X}}. \tag{2.6}$$

Representing Extrinsic Parameters

There are several possibilities to express the extrinsic camera parameters. For example, quaternions or Euler angles are common choices to represent rotations [50]. However, using Lie groups for parameterising motion has become increasingly popular in the area of computer vision and [80, 32]

are two recent works that illustrate this trend. Lie groups allow to exactly parameterise the available degrees of freedom and to conveniently compute derivatives of motion. As such, they offer a very elegant approach when it comes to estimating camera motion. Therefore, we now treat the most important notions related to Lie Groups. A detailed introduction can be found in Hall [48].

Lie Group. A Lie group is both a smooth manifold and a group such that the group product and the inverse map are smooth. Here, we are especially interested in real matrix Lie groups, where the elements can be represented by means of real matrices. Suppose that G is a matrix Lie group with k degrees of freedom and elements in $\mathbb{R}^{n\times n}$. Then the groups multiplication and inversion operation are simply given by matrix multiplication and matrix inversion, respectively. It is well known that the set of all invertible $n \times n$ matrices with the operation of matrix multiplication is a Lie group referred to as *general linear group* and that all matrix Lie groups are closed subgroups of it.

Lie Algebra. The corresponding Lie algebra $\mathfrak{g}$ is the tangent space around the identity element of G. It is a k-dimensional vector space, spanned by basis vectors $\boldsymbol{V}_1, \ldots, \boldsymbol{V}_k$. These basis vectors can be represented by matrices in $\mathbb{R}^{n\times n}$ as well and are commonly referred to as generators. It may be more intuitive to think of the tangent vectors in terms of their k coefficients. Assume that $\boldsymbol{\omega} \in \mathbb{R}^k$ is a vector of such coefficients. Then the corresponding element of the Lie algebra, i.e. the tangent space, is given by the linear combination

$$\hat{\boldsymbol{\omega}} = \sum_{i=1}^{k} \omega_i \boldsymbol{V}_i. \tag{2.7}$$

Exponential Map. The exponential map, $\exp : \mathfrak{g} \to G$, maps an element from the Lie algebra to an element of the Lie group. In the case of matrix Lie groups, the exponential map is simply given by the matrix exponential defined by the usual power series

$$\exp(\hat{\boldsymbol{\omega}}) = \sum_{i=0}^{\infty} \frac{\hat{\boldsymbol{\omega}}^i}{i!}. \tag{2.8}$$

Now it is easy to see that the partial derivative of $\exp(\hat{\boldsymbol{\omega}})$ w.r.t. ω_i at $\boldsymbol{\omega} = \mathbf{0}$ is simply given by the corresponding generator

$$\left.\frac{\partial \exp(\hat{\boldsymbol{\omega}})}{\partial \omega_i}\right|_{\boldsymbol{\omega}=\mathbf{0}} = \boldsymbol{V}_i. \tag{2.9}$$

For dealing with rotations, we are interested in the special orthogonal group SO(3). It is the group of rotations in 3D space given by the orthogonal 3×3 matrices with determinant 1. Its group generators are given by

$$\boldsymbol{V}_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \boldsymbol{V}_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{V}_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{2.10}$$

These generators are the tangent vectors around the identity element and correspond to derivatives of rotation around each of the axes. Let $\boldsymbol{R}_x(\alpha)$ perform a rotation around the X-axis, then $\boldsymbol{V}_1 = \partial_\alpha \boldsymbol{R}_x(\alpha) \mid_{\alpha=0}$. The same interpretation works for $\boldsymbol{V}_2$ and $\boldsymbol{V}_3$ by considering rotations around the Y-axis and Z-axis, respectively. In this case, the linear combination of the generators $\hat{\boldsymbol{\omega}}$ is a skew-symmetric matrix. More specifically, we have $\hat{\boldsymbol{\omega}} = [\boldsymbol{\omega}]_x$ where

$$[\boldsymbol{\omega}]_\times = \begin{pmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{pmatrix} \tag{2.11}$$

is the skew symmetric matrix that allows to express the cross product by matrix vector multiplication $\boldsymbol{\omega} \times \boldsymbol{v} = [\boldsymbol{\omega}]_\times \boldsymbol{v}$. This allows to evaluate the corresponding exponential map in closed form when using the fact that

$$[\boldsymbol{\omega}]_x^3 = -|\boldsymbol{\omega}|^2 \cdot [\boldsymbol{\omega}]_x \tag{2.12}$$

and considering the Taylor expansions of the sine and cosine functions. The exponential map then reads

$$\exp(\hat{\boldsymbol{\omega}}) = \boldsymbol{I} + \left(\frac{\sin(|\boldsymbol{\omega}|)}{|\boldsymbol{\omega}|}\right) \hat{\boldsymbol{\omega}} + \left(\frac{1 - \cos(|\boldsymbol{\omega}|)}{|\boldsymbol{\omega}|}\right) \hat{\boldsymbol{\omega}}^2, \tag{2.13}$$

which is the well known Rodrigues formula. By $|\cdot|$, we always denote the Euclidean norm unless explicitly stated otherwise. Being able to evaluate derivatives at $\boldsymbol{\omega} = \mathbf{0}$ and mapping from elements of the Lie algebra $\hat{\boldsymbol{\omega}}$ to elements of the Lie group, in this example a rotation matrix $\boldsymbol{R} \in \mathrm{SO}(3)$, is sufficient for the tasks required in this thesis. Generators and the corresponding exponential maps for other Lie groups such as SO(2), SE(2) or SE(3) can be derived in an analogous way [48].

General Pinhole Camera Model

Combining intrinsic and extrinsic camera parameters, the camera matrix $\boldsymbol{P}$ of a general pinhole camera can be obtained as

$$\tilde{\boldsymbol{x}} = \boldsymbol{K}\begin{pmatrix}\boldsymbol{I}_3 & \boldsymbol{0}_3\end{pmatrix}\tilde{\boldsymbol{X}}_c = \boldsymbol{K}\begin{pmatrix}\boldsymbol{R} & \boldsymbol{t}\end{pmatrix}\tilde{\boldsymbol{X}}. \tag{2.14}$$

Accordingly, the camera matrix $\boldsymbol{P}$ is given by

$$\boldsymbol{P} = \boldsymbol{K}\begin{pmatrix}\boldsymbol{R} & \boldsymbol{t}\end{pmatrix} = \begin{pmatrix}\boldsymbol{KR} & \boldsymbol{Kt}\end{pmatrix}. \tag{2.15}$$

In total, the general camera matrix offers 11 degrees of freedom: 5 arise from the calibration matrix $\boldsymbol{K}$ and 6 from rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$.

2.1.3 Backprojection

In the previous section, we have discussed how 3D points are projected to the image plane. In this section, we are concerned with the inverse operation, i.e. a backprojection. Obviously, the camera matrix is not invertible in the classical sense. However, the row vectors of the camera matrix are linearly independent such that its Moore-Penrose inverse can be computed as

$$\boldsymbol{P}^+ = \boldsymbol{P}^\top(\boldsymbol{P}\boldsymbol{P}^\top)^{-1}. \tag{2.16}$$

Obviously, the point $\boldsymbol{P}^+\tilde{\boldsymbol{x}}$ projects to $\tilde{\boldsymbol{x}}$ because $\boldsymbol{P}\boldsymbol{P}^+ = \boldsymbol{P}\boldsymbol{P}^\top(\boldsymbol{P}\boldsymbol{P}^\top)^{-1} = \boldsymbol{I}$. Thus, the ray that projects to $\tilde{\boldsymbol{x}}$ can be written as the join of this point and the camera centre

$$\tilde{\boldsymbol{X}}(\lambda) = \boldsymbol{P}^+\tilde{\boldsymbol{x}} + \lambda\tilde{\boldsymbol{c}}. \tag{2.17}$$

While this relation is quite general and holds for general camera matrices, it is possible to develop a more convenient expression for the pinhole camera model previously described. Let us consider a location $\boldsymbol{x}$ in the image plane. Then we know that the point

$$\boldsymbol{d} = (\boldsymbol{KR})^{-1}(\tilde{\boldsymbol{x}} - \boldsymbol{Kt}) \tag{2.18}$$

lies on the optical ray because it projects to $\tilde{\boldsymbol{x}}$:

$$\boldsymbol{P}\tilde{\boldsymbol{d}} = \begin{pmatrix}\boldsymbol{KR} & \boldsymbol{Kt}\end{pmatrix}\begin{pmatrix}(\boldsymbol{KR})^{-1}(\tilde{\boldsymbol{x}} - \boldsymbol{Kt}) \\ 1\end{pmatrix} = \tilde{\boldsymbol{x}}. \tag{2.19}$$

The camera centre $\tilde{\boldsymbol{c}}$ is the right null-space of $\boldsymbol{P}$ and can be expressed as

$$\boldsymbol{c} = -(\boldsymbol{KR})^{-1}\boldsymbol{Kt} \tag{2.20}$$

as described in [49]. Thus, we can define the optical ray as

$$\boldsymbol{s}(\lambda) = \boldsymbol{c} + \lambda(\boldsymbol{d} - \boldsymbol{c}) = \boldsymbol{c} + \lambda(\boldsymbol{K}\boldsymbol{R})^{-1}\tilde{\boldsymbol{x}}, \tag{2.21}$$

where $\lambda \in \mathbb{R}_+$. Here, λ describes the distance of the resulting point $\boldsymbol{s}(\lambda)$ to the camera centre $\boldsymbol{c}$. By construction, this distance is measured along the optical axis. In order to measure along the optical ray, i.e. the line of sight, we need to normalise $\boldsymbol{d} - \boldsymbol{c}$ such that we obtain the equation

$$\boldsymbol{l}(\lambda) = \boldsymbol{c} + \lambda \frac{(\boldsymbol{K}\boldsymbol{R})^{-1}\tilde{\boldsymbol{x}}}{|(\boldsymbol{K}\boldsymbol{R})^{-1}\tilde{\boldsymbol{x}}|} \,. \tag{2.22}$$

2.2 Two View Geometry

While the previous section was concerned with the projection and back-projection operation performed by a single perspective camera, this section treats important relations that occur in setups where two pinhole cameras are involved. In this case, one can consider the projections $\boldsymbol{x}$ and $\boldsymbol{x}'$ of a single 3D point $\boldsymbol{X}$ onto each of the image planes. Given a location $\boldsymbol{x}$, it is then interesting to understand in which way the corresponding point $\boldsymbol{x}'$ is constrained. Before we deal with this constraint in detail and describe how to express it by means of the *fundamental matrix* or the *essential matrix*, let us first go over some important terminology.

As illustrated in Figure 2.2, the line connecting the centres of both cameras is referred to as *baseline*. It allows to define the *epipole* $\boldsymbol{e}$ as the intersection of the baseline with the image plane. Every plane that contains the baseline is called *epipolar plane*. Thus, there is a family of epipolar planes that offers one degree of freedom. By intersecting the epipolar plane with an image plane, we are able to describe an *epipolar line*. Thus, by construction all epipolar lines have to intersect in the epipole.

2.2.1 Epipolar Constraint

Let us consider an image point $\boldsymbol{x}$ in the left camera. Then the corresponding scene point $\boldsymbol{X}$ has to lie on the optical ray through $\boldsymbol{x}$. Therefore, the position $\boldsymbol{x}'$ of the projection of $\boldsymbol{X}$ in the right camera cannot be arbitrary. In fact, $\boldsymbol{x}'$ has to be located on the epipolar line $\boldsymbol{l}'$. This constraint is known as the *epipolar constraint* and can be expressed as

$$\boldsymbol{x}'^{\top}\boldsymbol{l}' = 0. \tag{2.23}$$

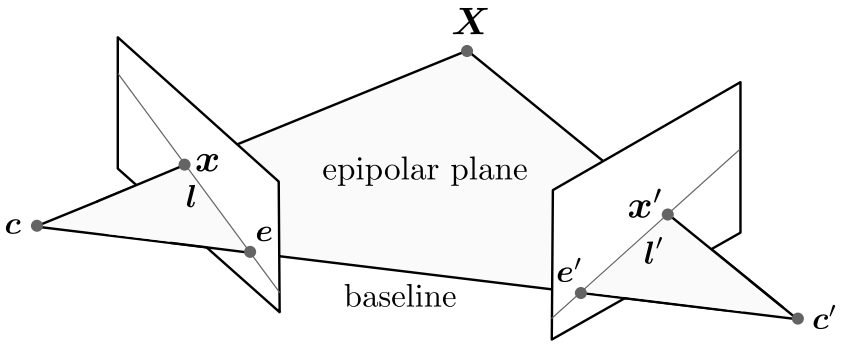


Figure 2.2: Epipolar geometry between to views.

Now we are are interested in a mapping

$$\boldsymbol{x} \mapsto \boldsymbol{l}' \tag{2.24}$$

that allows to determine the epipolar line $\boldsymbol{l}'$ given an image point $\boldsymbol{x}$. In fact, the epipolar line can be obtained by projecting the optical ray through $\boldsymbol{x}$ into the right camera. Looking at the formula for backprojection (2.17) immediately reveals that the two points $\boldsymbol{e}' = \boldsymbol{P}'\tilde{\boldsymbol{c}}$ and $\boldsymbol{P}'\boldsymbol{P}^{+}\tilde{\boldsymbol{x}}$ have to lie on the epipolar line. Since the cross product of two homogeneous points yields the line passing through them, the epipolar line can be defined as

$$\boldsymbol{l}' = [\boldsymbol{e}']_{\times}\boldsymbol{P}'\boldsymbol{P}^{+}\tilde{\boldsymbol{x}}. \tag{2.25}$$

2.2.2 Fundamental Matrix

The fundamental matrix $\boldsymbol{F}$ is the algebraic representation of the mapping from an image point to the associated epipolar line given by

$$\boldsymbol{F} = [\boldsymbol{e}']_{\times}\boldsymbol{P}'\boldsymbol{P}^{+}. \tag{2.26}$$

For the case of the pinhole camera model from the previous section, where the first camera is aligned with the world coordinate system, the fundamental matrix reads

$$\boldsymbol{F} = \boldsymbol{K}'^{-\top}[\boldsymbol{t}]_{\times}\boldsymbol{R}\boldsymbol{K}^{-1}. \tag{2.27}$$

By means of the fundamental matrix, the epipolar constraint (2.23) for two corresponding points can be expressed as

$$\tilde{\boldsymbol{x}}'^{\top}\boldsymbol{F}\tilde{\boldsymbol{x}} = 0. \tag{2.28}$$

This shows that it is also possible to define $\boldsymbol{F}$ merely by point correspondences without any knowledge of the camera matrices.

Properties of the Fundamental Matrix

Assume that $\boldsymbol{l}$ and $\boldsymbol{l}'$ are a pair of epipolar planes obtained by intersecting the images planes with an epipolar plane. Then it is easy to see that all points on $\boldsymbol{l}$ are required to have a correspondence that lies on $\boldsymbol{l}'$. For this reason, the fundamental matrix $\boldsymbol{F}$ is not injective. This means it cannot be inverted and is not of full rank. In fact it has rank 2 and 7 degrees of freedom. This is because it is only defined up to a scale in addition to the rank deficiency. A pair of cameras is called *weakly calibrated* if the fundamental matrix is known.

Projective Reconstruction Theorem

As we have seen in Equation 2.25, the fundamental matrix can be uniquely defined by the camera matrices $\boldsymbol{P}$ and $\boldsymbol{P}'$. Now it is interesting to ask to what degree a given fundamental matrix determines both involved camera matrices. To this end, let us introduce the projective transformation matrix $\boldsymbol{H} \in \mathbb{R}^{4\times4}$. Then

$$\tilde{\boldsymbol{x}} = \boldsymbol{P}\tilde{\boldsymbol{X}} = (\boldsymbol{P}\boldsymbol{H})(\boldsymbol{H}^{-1}\tilde{\boldsymbol{X}}), \tag{2.29}$$

$$\tilde{\boldsymbol{x}'} = \boldsymbol{P}'\tilde{\boldsymbol{X}} = (\boldsymbol{P}'\boldsymbol{H})(\boldsymbol{H}^{-1}\tilde{\boldsymbol{X}}) \tag{2.30}$$

holds. This means that the points $\tilde{\boldsymbol{X}}$ or $\boldsymbol{H}^{-1}\tilde{\boldsymbol{X}}$ with corresponding camera pairs $(\boldsymbol{P}, \boldsymbol{P}')$ or $(\boldsymbol{P}\boldsymbol{H}, \boldsymbol{P}'\boldsymbol{H})$ yield the same correspondences. For this reason, both camera pairs share the same fundamental matrix. In fact it is known that this is the only degree of freedom such that a given fundamental matrix allows for camera extraction up to a projective transformation [49]. This implies that also a 3D reconstruction based on a fundamental matrix can only be estimated up to a projective transformation unless further information is supplied to impose additional constraints.

2.2.3 Essential Matrix

The essential matrix allows to establish a relation between normalised image coordinates

$$\boldsymbol{K}^{-1}\tilde{\boldsymbol{x}} \quad \text{and} \quad \boldsymbol{K}'^{-1}\tilde{\boldsymbol{x}}'. \tag{2.31}$$

Such normalised image points can be understood as projections of a camera

$$\boldsymbol{K}^{-1}\boldsymbol{P} = \begin{pmatrix}\boldsymbol{R} & \boldsymbol{t}\end{pmatrix}, \tag{2.32}$$

that has been normalised by reversing the effect of the intrinsic camera parameters. Alternatively, such a normalised camera matrix can also be seen

as a camera with the calibration matrix $\boldsymbol{K}$ equivalent to the identity matrix $\boldsymbol{I}$. We can obtain the *essential matrix*

$$\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R} \tag{2.33}$$

using Equation 2.27 and the fact that $\boldsymbol{K} = \boldsymbol{K}' = \boldsymbol{I}$ in this case. Thus, we see that the essential matrix can be uniquely defined by the camera pose $\begin{pmatrix}\boldsymbol{R} & \boldsymbol{t}\end{pmatrix}$. However, just as with the fundamental matrix, also the essential matrix can be defined by relating coordinates:

$$(\boldsymbol{K}'^{-1}\tilde{\boldsymbol{x}}')^\top \; \boldsymbol{E} \; (\boldsymbol{K}^{-1}\tilde{\boldsymbol{x}}) = 0, \tag{2.34}$$

where in this case the normalised coordinates are used. By comparing this to Equation 2.27, we can relate the essential matrix to the fundamental matrix by

$$\boldsymbol{E} = \boldsymbol{K}'^\top \boldsymbol{F} \boldsymbol{K}. \tag{2.35}$$

Properties of the Essential Matrix

As a homogeneous quantity, the essential matrix offers 5 degrees of freedom. Six degrees of freedom stem from $\boldsymbol{R}$ and $\boldsymbol{t}$ but one has to be subtracted due to scale ambiguity. From the decomposition $\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R}$, it can be shown that the first two singular values have to be equivalent and the third one has to be zero [49]. While the camera pose $\begin{pmatrix}\boldsymbol{R} & \boldsymbol{t}\end{pmatrix}$ uniqueley determines the essential matrix, it is again interesting to understand to what degree a known essential matrix constrains the camera pair. Compared to the fundamental matrix, the reduced degree of freedom leads to additional constraints. While the fundamental matrix only allows camera extraction up to a projective ambiguity, the essential matrix allows to retrieve the involved camera matrices up to scale. To be more exact, the essential matrix still leaves room for a four-fold ambiguity. This means that four solutions exist additional to the scale ambiguity. However, the four-fould ambiguity can easily be resolved using a single point correspondence. In order to extract a camera pair from an essential matrix $\boldsymbol{E}$, one has to understand how it can be decomposed into a skew-symmetric matrix $[\boldsymbol{t}]_\times$ and a rotation matrix $\boldsymbol{R}$.

Decomposition

To this end, it is important to know that a skew-symmetric matrix $[\boldsymbol{t}]_\times$ may be decomposed as [49]

$$[\boldsymbol{t}]_\times = k \, \boldsymbol{U} \boldsymbol{Z} \boldsymbol{U}^\top, \tag{2.36}$$

where $k \in \mathbb{R}$, $\boldsymbol{U}$ is an orthogonal matrix and

$$\boldsymbol{Z} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{2.37}$$

Then it is possible to construct an orthogonal matrix

$$\boldsymbol{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.38}$$

satisfying $\boldsymbol{Z} = \mathbf{diag}(1, 1, 0)\boldsymbol{W}$ up to sign. Thus, $[\boldsymbol{t}]_\times = \boldsymbol{U}\mathbf{diag}(1, 1, 0)\boldsymbol{W}\boldsymbol{U}^\top$ holds up to scale. With this we have the singular value decomposition (SVD)

$$\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R} = \boldsymbol{U}\mathbf{diag}(1, 1, 0)\left(\boldsymbol{W}\boldsymbol{U}^\top \boldsymbol{R}\right) \tag{2.39}$$

with the orthogonal matrices $\boldsymbol{U}$ and $\boldsymbol{V}^\top := \boldsymbol{W}\boldsymbol{U}^\top\boldsymbol{R}$. Thus, the SVD contains exactly two equal singular values that are non-zero. On the other hand, it is also possible to decompose a matrix with two equal non-zero singular values into $[\boldsymbol{t}]_\times\boldsymbol{R}$ in this way. In fact, this allows to extract a pair of camera matrices up to a four-fold and scale ambiguity. Assuming that just a single correspondence is known, which is usually easy to obtain, the fourfold ambiguity can be eliminated such that just the scale ambiguity remains. The following section explains the camera extraction in more detail.

Extracting a Camera Pair from the Essential Matrix

From the previous decomposition (2.39) we can find

$$[\boldsymbol{t}]_\times = \boldsymbol{U}\boldsymbol{Z}\boldsymbol{U}^\top \tag{2.40}$$

and

$$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{W}^\top\boldsymbol{V}^\top. \tag{2.41}$$

In fact, also $\boldsymbol{R} = \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^\top$ is a possible solution, due to the open sign [49]. Having found $[\boldsymbol{t}]_\times$ from $\boldsymbol{E}$, we still have to find $\boldsymbol{t}$. To this end, one can use the fact $[\boldsymbol{t}]_\times\boldsymbol{t} = 0$ and see from (2.36) that $\boldsymbol{t} = \boldsymbol{U}(0, 0, a)^\top$ with $a \in \mathbb{R}$ fulfils this. The scalar value a reflects the open scale and is usually chosen equal to one such that the baseline also has a length of one and corresponds to the

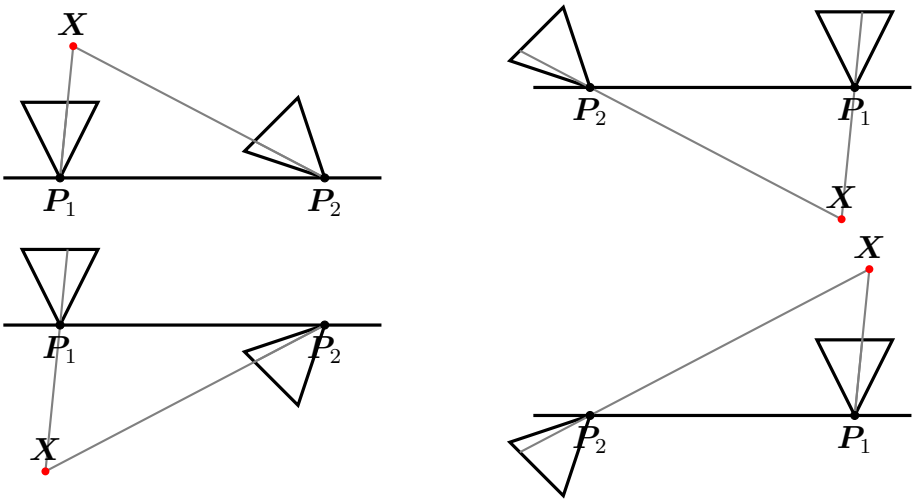


Figure 2.3: Four different solutions for camera pairs extracted from the essential matrix $\boldsymbol{E}$. Only for one of the configurations the reconstructed 3D point $\boldsymbol{X}$ is in front of both cameras.

third column of $\boldsymbol{U}$, i.e. $\boldsymbol{t} = \pm\boldsymbol{u}_3$. Summing up, this leaves four solutions besides the scale ambiguity:

$$\boldsymbol{P}' \in \Big\{ \begin{array}{lll} (\, \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^\top & | & +\boldsymbol{u}_3 \,), \\ (\, \boldsymbol{U}\boldsymbol{W}^\top\boldsymbol{V}^\top & | & +\boldsymbol{u}_3 \,), \\ (\, \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^\top & | & -\boldsymbol{u}_3 \,), \\ (\, \boldsymbol{U}\boldsymbol{W}^\top\boldsymbol{V}^\top & | & -\boldsymbol{u}_3 \,) \end{array} \Big\}.$$

This four-fould ambiguity can easily be resolved using a single point correspondence because only for one configuration, the reconstructed 3D point will be in front of both cameras, see Fig 2.3. The different configurations can be related by a rotation of 180° around the baseline and reversing the translation.

2.3 Calculus of Variations

Functionals are an important tool for describing problems that arise in various disciplines such as mathematics or physics and are also the basis of the approaches we present. Thus, we now cover the most important concepts that are required within this thesis. Gelfand and Fomin [42] offer an extensive treatment of this topic.

A functional maps each function belonging to some space of functions to a real number. As such, a functional can be interpreted as a function, where the variable itself is a function. Problems that consist of finding maxima or minima of functionals are referred to as *variational problems*. The functional

$$E(\boldsymbol{u}) = \int_\Omega F(\boldsymbol{x}, u_1, \boldsymbol{\nabla} u_1, \boldsymbol{H} u_1, \dots, u_m, \boldsymbol{\nabla} u_m, \boldsymbol{H} u_m) \, \mathrm{d}\boldsymbol{x}, \tag{2.42}$$

of order two with $\Omega \subset \mathbb{R}^n$ and $\boldsymbol{u} : \Omega \to \mathbb{R}^m$ is sufficient to cover the models used within this thesis and usually we will have $m \in \{1, 2\}$ and $n \in \{2, 3\}$. All models in this thesis are posed in such a way that we are interested in finding the minimiser of a given functional. It is well known that a minimiser must necessarily fulfil the corresponding Euler-Lagrange equations

$$\frac{\delta E}{\delta u_k} = 0 \tag{2.43}$$

for $k = 1, \dots, m$ along with the natural boundary conditions. To find the functional derivatives $\frac{\delta E}{\delta u_k}$ one computes the Gâteaux derivative

$$\delta E(u_k, v) = \frac{\mathrm{d}}{\mathrm{d}\varepsilon} \, E(u_1, \dots, u_k + \varepsilon v, \dots u_m) \Big|_{\varepsilon=0}, \tag{2.44}$$

which can be interpreted as an extension of the directional derivative to the infinite dimensional case and is also commonly referred to as the *first variation* of a functional. Using multidimensional integration by parts and the definition

$$\delta E(u_k, v) = \left\langle \frac{\delta E}{\delta u_k}, v \right\rangle \tag{2.45}$$

allows to identify the functional gradients for (2.42) as

$$\frac{\delta E}{\delta u_k} = F_{u_k} - \sum_{i=1}^{n} \partial x_i F_{\partial x_i u_k} + \sum_{i,j=1}^{n} \partial x_i \partial x_j F_{\partial x_j \partial x_i u_k} \tag{2.46}$$

with the natural boundary conditions

$$\sum_{i=1}^{n} \left(F_{\partial x_i u_k} - \sum_{j=1}^{n} \partial x_j F_{\partial x_j \partial x_i u_k} \right) n_i = 0 \tag{2.47}$$

and

$$\sum_{j=1}^{n} F_{\partial x_j \partial x_i u_k} n_j = 0. \tag{2.48}$$

Here, $\boldsymbol{n}$ is the outer normal vector to the boundary of Ω. In order to solve these partial differential equations numerically, they have to be discretised, for example with a finite differences scheme or the finite element method [42]. In this way, we can turn the originally infinite dimensional problem into a finite dimensional one. An alternative approach is given by discretising the energy functional in the first step before computing its gradient. However, either strategy will lead to a (non)linear system of equations. In fact in many cases both strategies even yield the exact same system of equations. In our case, it is always of the form

$$\boldsymbol{M}(\boldsymbol{x})\, \boldsymbol{x} = \boldsymbol{b}(\boldsymbol{x}), \tag{2.49}$$

where $\boldsymbol{M}(\boldsymbol{x})$ and $\boldsymbol{b}(\boldsymbol{x})$ are matrix valued and vector valued operators, respectively. For each $\boldsymbol{x} \in \mathbb{R}^n$, the value of $\boldsymbol{M}(\boldsymbol{x})$ is a symmetric positive definite $n \times n$ matrix and the value of $\boldsymbol{b}(\boldsymbol{x})$ is a vector with n components. This nonlinear problem can then be tackled in form of a fixed point iteration

$$\boldsymbol{x}^{k+1} = \phi\left(\boldsymbol{x}^k\right) \tag{2.50}$$

for $k \in \mathbb{N}$ with

$$\phi(\boldsymbol{x}) = \left(\boldsymbol{M}(\boldsymbol{x})\right)^{-1} \boldsymbol{b}(\boldsymbol{x}). \tag{2.51}$$

In this case, solving the nonlinear problem comes down to solving a series of linear systems of the form

$$\boldsymbol{M}(\boldsymbol{x}^k)\, \boldsymbol{x}^{k+1} = \boldsymbol{b}(\boldsymbol{x}^k), \tag{2.52}$$

where the basic idea is to evaluate the nonlinear operators with the old solution $\boldsymbol{x}^k$. This idea corresponds to the *Lagged Diffusivity* or *Kačanov* method [41, 25]. Since solving linear systems is at the core of this fixed point iteration, let us discuss efficient approaches for this task.

2.4 Numerical Solvers for Linear Systems

Iterative methods have become more and more popular for solving large and sparse linear systems of equations as they can offer several advantages when compared to direct solvers: Iterative methods are usually simpler to implement and it is straightforward to exploit the sparsity pattern of the problem at hand. This allows to efficiently solve large and sparse problems with essentially no overhead. As we will later see, all problems discussed in this thesis lead to very large and sparse system matrices where the number

of unknowns is typically larger than 10^6 and there are only a few nonzero elements compared to the total number of elements. Furthermore, iterative methods are well suited for parallel hardware which is a very important aspect nowadays as GPUs have become quite cheap and are widely used in scientific computing.

Iterative methods can be put into two categories [90]: On the one hand, there are the basic iterative methods such as the Jacobi method or the Gauss-Seidel method. On the other hand, there are the projection methods such as the conjugate gradient algorithm which is based on a projection process onto Krylov subspaces. In the following, we will focus on the basic iterative methods. More specifically, we will first discuss stationary methods and then present a recently proposed nonstationary method called Fast Jacobi which is able to drastically speed up convergence and requires almost no extra efforts in implementation compared to the stationary methods.

2.4.1 Stationary Iterative Methods

Let us assume we are given a linear system of equations

$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{b}, \tag{2.53}$$

with a symmetric positive definite matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$, a vector $\boldsymbol{b} \in \mathbb{R}^n$ and an unknown $\boldsymbol{x} \in \mathbb{R}^n$. Stationary iterative methods [122, 90] can then be explained by splitting the system matrix

$$\boldsymbol{M} = \boldsymbol{B} + (\boldsymbol{M} - \boldsymbol{B}) \tag{2.54}$$

with an invertible matrix $\boldsymbol{B} \in \mathbb{R}^{n\times n}$. Then we have

$$(\boldsymbol{B} + (\boldsymbol{M} - \boldsymbol{B}))\boldsymbol{x} = \boldsymbol{b}, \tag{2.55}$$

which can be rephrased as the fixed point iteration

$$\boldsymbol{x}^{k+1} = \boldsymbol{B}^{-1}(\boldsymbol{b} - (\boldsymbol{M} - \boldsymbol{B})\boldsymbol{x}^k) = (\boldsymbol{I} - \boldsymbol{B}^{-1}\boldsymbol{M})\boldsymbol{x}^k + \boldsymbol{B}^{-1}\boldsymbol{b}. \tag{2.56}$$

In general, there is a tradeoff one has to consider when choosing the matrix $\boldsymbol{B}$: On the one hand, it should be simple to invert it and on the other hand it should be a good approximation of $\boldsymbol{M}$. Table 2.1 lists the choices of $\boldsymbol{B}$ for several popular methods, where

$$\boldsymbol{M} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U} \tag{2.57}$$

Method	$\boldsymbol{B}$
Richardson	$\boldsymbol{I}$
Relaxed Richardson	$\omega\boldsymbol{I}$
Jacobi	$\boldsymbol{D}$
Relaxed Jacobi	$\frac{1}{\omega}\boldsymbol{D}$
Gauss-Seidel	$\boldsymbol{D}+\boldsymbol{L}$
Successive over-relaxation	$\frac{1}{\omega}\boldsymbol{D}+\boldsymbol{L}$

Table 2.1: Choices of $\boldsymbol{B}$ for several iterative linear solvers.

is composed of a strictly lower triangular, a diagonal, and a strictly upper triangular matrix.

In order to ensure convergence, it is well known that the spectral radius of the iteration matrix has to be smaller than one, i.e.

$$\rho\left(\boldsymbol{I}-\boldsymbol{B}^{-1}\boldsymbol{M}\right)<1. \tag{2.58}$$

Since the spectral radius is the largest modulus of the eigenvalues of a matrix, this implies that all eigenvalues should lie in the interval $(-1,1)$.

2.4.2 Fast Jacobi

Before explaining the Fast Jacobi algorithm, let us first consider an iteration of the relaxed Jacobi method [110]. It reads

$$\boldsymbol{x}^{k+1}=(\boldsymbol{I}-\omega\boldsymbol{D}^{-1}\boldsymbol{M})\boldsymbol{x}^{k}+\omega\boldsymbol{D}^{-1}\boldsymbol{b}, \tag{2.59}$$

which corresponds to a choice of $\boldsymbol{B}=\frac{1}{\omega}\boldsymbol{D}$ as shown in Table 2.1. For the choice of $\omega=1$ it is equivalent to the standard Jacobi method and concerning convergence according to (2.58), we require

$$\rho\left(\boldsymbol{I}-\omega\boldsymbol{D}^{-1}\boldsymbol{M}\right)<1. \tag{2.60}$$

Let us assume that $\mu_{max}(\boldsymbol{D}^{-1}\boldsymbol{M})$ denotes the largest eigenvalue of $\boldsymbol{D}^{-1}\boldsymbol{M}$, where $\boldsymbol{M}$ is positive definite. Then we can easily see that (2.60) is fulfilled as long as we have

$$\omega<\frac{2}{\mu_{max}(\boldsymbol{D}^{-1}\boldsymbol{M})}=:\omega_{max}. \tag{2.61}$$

The relaxed Jacobi method is a stationary iterative method and keeps the iteration matrix fixed. In contrast to this, nonstationary methods adapt the iteration matrix as they proceed. A subclass of these methods uses cycles of varying iteration matrices. Such approaches are known as *periodic nonstationary* methods and the Fast Jacobi algorithm belongs to this class. One cycle of length n reads

$$\boldsymbol{x}^{i+1} = (\boldsymbol{I} - \omega_i \boldsymbol{D}^{-1}\boldsymbol{M})\boldsymbol{x}^i + \omega_i \boldsymbol{D}^{-1}\boldsymbol{b}, \tag{2.62}$$

with $i = 0, \dots, n-1$ and the relaxation parameters

$$\omega_i = \omega \cdot \frac{1}{2\cos^2\left(\pi \cdot \frac{2i+1}{4n+2}\right)}\,, \tag{2.63}$$

where ω fulfils (2.61). These relaxation parameters differ from classical choices and have been derived via a factorisation of a box filter. In the setting of fast explicit diffusion, this offers a good compromise between an accurate approximation of a Gaussian and a large stopping time when compared to factorisations of other symmetric filters. In the elliptic case this translates to a good compromise between error damping or smoothing properties and fast convergence [10]. A cycle of length n of Fast Jacobi (2.62) can also be written directly in terms of $\boldsymbol{x}^0$ as

$$\boldsymbol{x}^n = \underbrace{\left(\prod_{i=0}^{n-1}\left(\boldsymbol{I} - \omega_i \boldsymbol{D}^{-1}\boldsymbol{M}\right)\right)}_{\boldsymbol{G}}\boldsymbol{x}^0 + \underbrace{\sum_{i=0}^{n-1}\omega_i\left(\prod_{j=i+1}^{n-1}\left(\boldsymbol{I} - \omega_j \boldsymbol{D}^{-1}\boldsymbol{M}\right)\right)\boldsymbol{D}^{-1}\boldsymbol{b}}_{\boldsymbol{r}}. \tag{2.64}$$

Then performing several cycles of Fast Jacobi can be interpreted as the stationary method

$$\boldsymbol{x}^{k+1} = \boldsymbol{G}\boldsymbol{x}^k + \boldsymbol{r} \tag{2.65}$$

that is a composite of Fast Jacobi cycles and converges if $\rho(\boldsymbol{G}) < 1$. Let $\mu_1, \dots, \mu_N$ be the positive eigenvalues of $\boldsymbol{D}^{-1}\boldsymbol{M} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{v}_1, \dots, \boldsymbol{v}_N$ its eigenvectors. Then one can see that

$$\boldsymbol{G}\boldsymbol{v}_j = \prod_{i=0}^{n-1}\left(\boldsymbol{I} - \omega_i \boldsymbol{D}^{-1}\boldsymbol{M}\right)\boldsymbol{v}_j = \prod_{i=0}^{n-1}\left(1 - \omega_i \cdot \mu_j\right)\boldsymbol{v}_j, \tag{2.66}$$

which means that $\boldsymbol{D}^{-1}\boldsymbol{M}$ and $\boldsymbol{G}$ share the same eigenvectors and the eigenvalues of $\boldsymbol{G}$ are given by

$$\lambda_j = \prod_{i=0}^{n-1}\left(1 - \omega_i \cdot \mu_j\right) \tag{2.67}$$

Algorithm 1: Fast Jacobi method [10].

Input: Positive definite system matrix $\boldsymbol{M}$,
Right hand side $\boldsymbol{b}$,
Cycle length n,
Relaxation parameter $\omega \in (0, \omega_{max})$,
Accuracy parameter $\epsilon > 0$

Output: Numerical solution of linear system $\boldsymbol{x_0} = \boldsymbol{M}^{-1}\boldsymbol{b}$

```
   /* Initialise                                                    */
1  initialise(x_0);
2  for i = 0 to n − 1 do
3  |  ω_i = ω · 1 / (2cos²(π · (2i+1)/(4n+2)))
4  end
5  reorderWeights(ω_0, ..., ω_{n−1});                  /* See [10] */
   /* Perform cycles of Fast Jacobi                                */
6  repeat
7  |  for i = 0 to n − 1 do
8  |  |  x_1 = (I − ω_i D^{−1} M) x_0 + ω_i D^{−1} b;
9  |  |  x_0 = x_1;
10 |  end
11 until converged(M, b, x_0, x_1);
```

for $j = 1, \ldots, N$. This is related to the filter factorisation [10] and it was shown that $\lambda_j \in (-1, 1)$ for all j when $\omega < \omega_{max}$ according to (2.61).

Of course the matrix $\boldsymbol{G}$ is never actually computed in order to perform the Fast Jacobi algorithm. Instead one just repeats the cycles of varying parameters as in Equation 2.62. Algorithm 1 outlines this behaviour. Here it is important to note that one always has to perform a full cycle. It is not allowed to stop in between. Although the ordering of relaxation parameters does not matter in exact arithmetic, it influences the result in practice as rounding errors can lead to instability, especially when n is large. As a remedy the relaxation weights are usually rearranged within the sequence (cf. Line 5 of Algorithm 1) such that small ones and large ones are more evenly

distributed. Two strategies for this, namely κ-cycles and Leja ordering, are described in [10].

There are different criteria that can be considered to check for convergence which is required in Line 11 of Algorithm 1. One prominent example is checking whether the relative residual is below some threshold [60]

$$\frac{|\boldsymbol{b} - \boldsymbol{M}\boldsymbol{x}_k|}{|\boldsymbol{b}|} < \varepsilon \ . \tag{2.68}$$

Here, $\boldsymbol{x}_k$ denotes the solution after k iterations. However, the relative error of the solution depends on the condition number of $\boldsymbol{M}$. Considering the relation between the initial residual and the current one instead, i.e.

$$\frac{|\boldsymbol{b} - \boldsymbol{M}\boldsymbol{x}_k|}{|\boldsymbol{b} - \boldsymbol{M}\boldsymbol{x}_0|} < \varepsilon \ , \tag{2.69}$$

can lead to a more appropriate stopping criterion. Another approach is to check whether the norm of the difference in solutions between iterations is smaller than a threshold, i.e. $|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}| < \varepsilon$.

In general, an iterative method does not have to depend on the last computed solution only. It could also make use of several older solutions. In [46] some interesting variants have been proposed that also yield the advantage that they do not require a reordering for numerical stability. However in the scenarios present in this thesis it can already be prohibitive to rely on more than one old solution due to the fact the we consider very large problems that are quickly constrained by the memory available on a GPU.

Chapter 3

Multi-View Depth Estimation

The task of reconstructing 3D scenes from a number of images along with corresponding camera poses is commonly referred to as *multi-view stereo*. One can approach the multi-view stereo problem by dividing it into the following two steps: First, one computes depth maps for a number of input images. Second, these depth maps are merged with a volumetric approach, see e.g. [29, 124, 123]. In this case, the multi-view stereo problem constitutes a common example, where one is interested in obtaining a depth map given multiple views. This is the problem we focus on in this chapter. It is illustrated in Figure 3.1 and we refer to it as *multi-view depth estimation*. In this scenario, it is assumed that all camera parameters have already been estimated such that only the unknown depth as seen from one reference camera has to be recovered.

Previous work has demonstrated that multiple views improve the depth reconstruction, and that higher order regularisers model a good prior for typical real-world 3D scenes. We build on these findings and specifically analyse an important aspect that has not been considered in variational multi-view depth estimation so far: We investigate several parameterisations of the unknown depth. While most existing methods directly work with depth values, we show, both analytically and experimentally, that this introduces an undesirable bias. As a remedy, we reveal that an inverse depth parameterisation is generally preferable. Our analysis clearly points out its benefits w.r.t. the data and the smoothness term of a variational multi-view depth estimation approach. This work has been published in [8].

Organisation of this Chapter. After discussing related work as well as pointing out our contributions, we present a variational formulation for the estimation of depth maps from multiple views with an arbitrary parameterisation in Section 3.3. Subsequently, we analyse different parameterisations

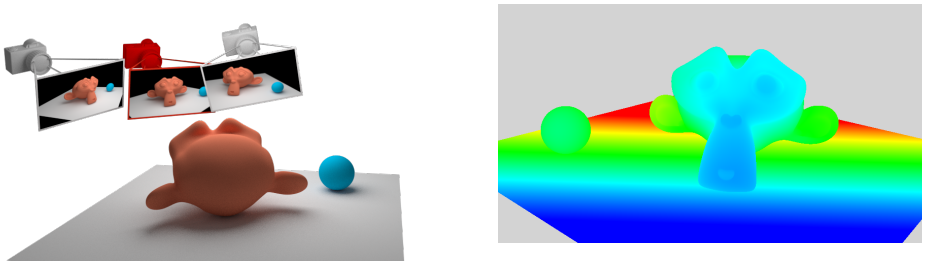


Figure 3.1: In multi-view depth estimation a number of images captured from a static scene (left) is used to estimate a single depth map (right) as seen from one selected reference camera which is shown in red. Here the depth map is visualised using a colour coding.

in detail (Section 3.4). In Section 3.5, we discuss the minimisation. Finally, we show experimental results (Section 3.6) as well as a discussion of limitations (Section 3.7) before we summarise the most important insights of this chapter (Section 3.8).

3.1 Related Work

Ignoring the fact that multiple views are available, variational stereo algorithms that consider image pairs can be regarded as related work, see e.g. [76, 75, 91, 102, 126, 108]. While these variational formulations compute disparities relying on a first order regularisation, higher order regularisation has proven to be a very successful strategy for many applications [86, 35, 52, 87]. Often, coupled formulations are used instead of directly implementing a higher order regulariser. Popular variants for this are total generalised variation [18] or an approach as in [52]. Also infimal convolution is a much related alternative, where first ideas of this can be found in [23]. Recently, Ranftl et al. demonstrated the benefits of second order regularisation in the context of optic flow [87] and stereo [86].

However, considering only two of the multiple views discards a lot of the available information. Unfortunately, it is not convenient to extend the concept of computing disparities to a general multi-view setting. Hence, there are a number of variational formulations that directly estimate depth from multiple views. Such methods have shown the benefits of using multiple images in the process of depth map estimation. To the best of our knowledge, the basic idea of considering multiple views to estimate a single depth map

within a variational formulation is almost two decades old and goes back to Robert and Deriche [88]. They employed a quadratic data term along with a nonquadratic regulariser that is able to preserve depth discontinuities. More recently, Stühmer et al. [105] presented a similar formulation with a robust penaliser for the smoothness term as well as the data term. Instead of the brightness constancy, assumed by [88] and [105], Semerijan [96] uses a gradient constancy assumption and a finite element discretisation.

While all aforementioned approaches that are able to estimate a depth map from multiple images are directly parameterised by the unknown depth, Strecha and Gool [104] proposed a PDE-based formulation with a depth related parameter that essentially comes down to an inverse depth parameterisation. Furthermore, in related problems such as monocular simultaneous localisation and mapping (SLAM), an inverse depth parameterisation of point features has been shown to be beneficial [28]. Also the dense tracking and mapping approach of Newcombe et al. uses inverse depth to compute cost values in a discrete cost volume [80] and the recently developed LSD-SLAM estimates probabilistic semi-dense inverse depth maps [32].

3.2 Contributions

While most existing variational multi-view formulations [88, 105, 96] directly compute the unknown depth from a number of arbitrarily placed cameras, we generalise them by introducing a depth parameterisation. This allows us to efficiently analyse advantages and drawbacks of different parameterisations such as a direct depth parameterisation and an inverse depth parameterisation. More specifically, we analyse two important aspects: On the one hand, the choice of parameterisation is important when considering the linearisation of the data term in a variational framework. Here, we show that for common camera setups, the inverse depth parameterisation is preferable. On the other hand, the choice of parameterisation is also important in the smoothness term, especially in the presence of second order regularisation. Here, we show that for an inverse depth parameterisation, piecewise affine functions correspond to piecewise planar surfaces. This is not the case for a direct depth parameterisation. We give deep insights into the introduced bias by analysing the shape operator of the corresponding 3D surface.

3.3 Variational Formulation

In this section, we describe a variational framework that allows the estimation of a depth map d from multiple views under an arbitrary parameterisation. To this end, we express d as the composition of an unknown $\rho : \Omega \to \mathbb{R}_+$ and a parameterisation $\phi : \mathbb{R}_+ \to \mathbb{R}_+$ such that $d = \phi \circ \rho$. Then our energy functional has the form

$$E(\rho) = \int_\Omega D(\phi \circ \rho)\, \mathrm{d}\boldsymbol{x} \;+\; \alpha\, R(\rho), \tag{3.1}$$

with a data term $D(\phi \circ \rho)$ that enforces photoconsistency, a regulariser $R(\rho)$ that prefers (piecewise) affine functions, and a positive regularisation weight α. In the following sections, we explain our model components in more detail.

Data Term

Let us assume we are given n colour images $\boldsymbol{f}_1, \dots, \boldsymbol{f}_n$ and a reference image $\boldsymbol{f}_0$. The task of the data term $D(\phi \circ \rho)$ is to enforce photoconsistency between all available views. To this end, we first introduce a function $\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho)$ that maps a location $\boldsymbol{x} \in \Omega$ in the reference frame $\boldsymbol{f}_0$ with its depth $(\phi \circ \rho)(\boldsymbol{x})$ to the corresponding location in another image $\boldsymbol{f}_i$. This mapping is illustrated in Figure 3.2 and can be described as a composition of a backprojection and a projection operation explained in Section 2.1. This allows to model the assumption that corresponding points $\boldsymbol{x}$ and $\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho)$ have similar colour values as follows:

$$D(\phi \circ \rho) = \frac{1}{n} \sum_{i=1}^{n} \Psi\left(|\boldsymbol{f}_i(\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho)) - \boldsymbol{f}_0(\boldsymbol{x})|^2\right), \tag{3.2}$$

where the function $\Psi : \mathbb{R}_+ \to \mathbb{R}_+$ provides a robust penalisation. A common choice is $\Psi(s^2) = \sqrt{s^2 + \epsilon^2}$, which approximates an L_1 data term for $\epsilon \to 0$.

Smoothness Term

Higher order regularisation has shown its potential in several applications. Essentially, there are two possibilities to design such regularisers: Either by a direct penalisation of higher order derivatives or by introducing a coupling variable $\boldsymbol{w} : \Omega \to \mathbb{R}^2$. We opt for the second choice that results in the regulariser

$$R(\rho) = \inf_{\boldsymbol{w}} \left\{ \int_\Omega \left(\Psi\left(|\boldsymbol{\nabla}\rho - \boldsymbol{w}|^2\right) + \beta\, \Psi\left(|\boldsymbol{\mathcal{J}}\boldsymbol{w}|_F^2\right) \right) \mathrm{d}\boldsymbol{x} \right\}, \tag{3.3}$$

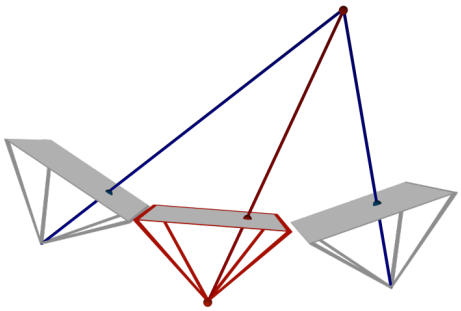


Figure 3.2: The mapping $\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho)$ allows to find the corresponding point in the i-th image plane when given a location and depth value in the reference image (red). It can be described by first performing a backprojection from the reference view followed by a projection into the i-th image plane.

where $\boldsymbol{\nabla}$ is the spatial gradient, $\boldsymbol{\mathcal{J}}$ the Jacobian, and $|\cdot|_F$ the Frobenius norm. This regulariser is similar to total generalised variation of second order (TGV^2) [18] and allows to obtain piecewise affine functions ρ.

Since our main focus is the analysis of different parameterisations, we restrict ourselves to the discussed model assumptions. Once the parameterisations are well understood, they can be incorporated in more sophisticated methods with more elaborate photoconsistency assumptions and regularisers that rank favourably in public benchmark systems.

3.4 Depth Parameterisations

We use the pinhole camera model described in Section 2.1 in order to analyse the effect of different parameterisations. With this model, the projection of a 3D point $\boldsymbol{X} \in \mathbb{R}^3$ to a point $\boldsymbol{x} \in \mathbb{R}^2$ in the image plane is given by

$$\boldsymbol{x} = \boldsymbol{\pi}\big(\boldsymbol{P}\tilde{\boldsymbol{X}}\big), \tag{3.4}$$

where $\tilde{\boldsymbol{X}} = (\boldsymbol{X}^\top, 1)^\top$ is the homogeneous version of $\boldsymbol{X}$, and

$$\boldsymbol{\pi}(a, b, c) = \begin{pmatrix} a/c \\ b/c \end{pmatrix} \tag{3.5}$$

maps a homogeneous coordinate to its Euclidean counterpart. Let us now look at the resulting surfaces when backprojecting depth maps under different parameterisations.

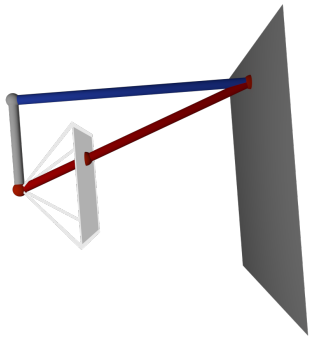


Figure 3.3: There are two common ways to measure the distance in the backprojection step. Either the distance is measured along the optical axis (blue) or it is measured along the line of sight (red).

3.4.1 Backprojection of Parameterised Depth Maps

We analyse the following parameterisations:

$$\text{(i) direct depth:} \quad \phi(r) = r, \tag{3.6}$$

$$\text{(ii) inverse depth:} \quad \phi(r) = 1/r. \tag{3.7}$$

In each case, there is a further design choice that we want to analyse, namely the choice of the distance, in which we measure. Basically, there are two meaningful possibilities to compute a backprojection:

$$\text{(a) along the line of sight:} \quad \boldsymbol{\ell}\big(\boldsymbol{x}, \phi \circ \rho\big) = \frac{\boldsymbol{K}^{-1}\tilde{\boldsymbol{x}}}{|\boldsymbol{K}^{-1}\tilde{\boldsymbol{x}}|} \cdot (\phi \circ \rho)(\boldsymbol{x}), \tag{3.8}$$

$$\text{(b) along the optical axis:} \quad \boldsymbol{s}\big(\boldsymbol{x}, \phi \circ \rho\big) = \boldsymbol{K}^{-1}\tilde{\boldsymbol{x}} \cdot (\phi \circ \rho)(\boldsymbol{x}). \tag{3.9}$$

These possibilities are illustrated in Figure 3.3. Figure 3.4 shows the resulting surfaces when backprojecting a constant and an affine function along the line of sight. Note that both parameterisations (i) and (ii) map constant and affine functions to curved surfaces. This means that a first order regulariser would already introduce an unwanted bias because it favours a (piecewise) constant ρ and thus curved surfaces. Therefore, we will not further consider parameterisations along the line of sight in our context.

In contrast, Figure 3.5 shows that both parameterisations along the optical axis map constant depth functions to surfaces with constant depth, and

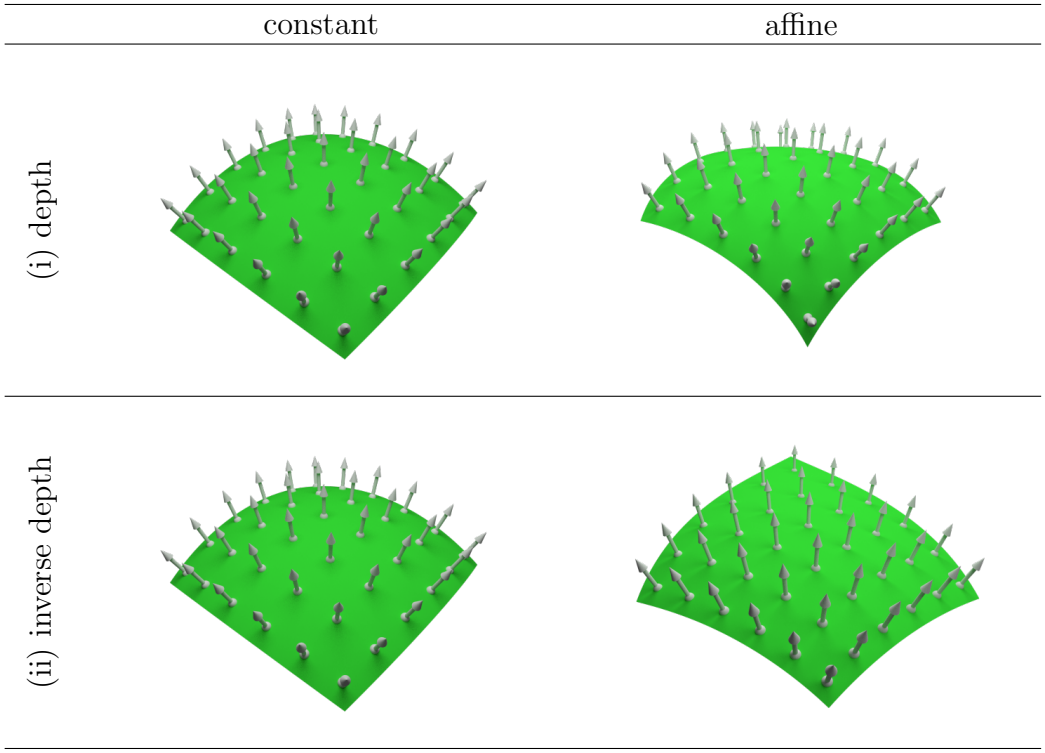


Figure 3.4: Resulting surfaces when backprojecting along the line of sight.

thus seem to be reasonable choices when employing a first order regularisation. However, considering an affine function (with a nonzero slope), we see that the depth parameterisation does not create a planar surface whereas the inverse depth parameterisation does. In the following sections, we analyse this in detail to get a better understanding of both choices.

3.4.2 Analysis of Backprojected Depth Maps

Let us consider (3.9) as a mapping from some parameter space Ω to a surface M, i.e. $\boldsymbol{s} : \Omega \subset \mathbb{R}^2 \to M \subset \mathbb{R}^3$. Generally, the tangent plane of a regular parameterised surface corresponding to a point $(x_0, y_0)^\top$ is spanned by the two tangent vectors

$$\boldsymbol{s}_x = \frac{\partial \boldsymbol{s}}{\partial x} \quad \text{and} \quad \boldsymbol{s}_y = \frac{\partial \boldsymbol{s}}{\partial y}, \tag{3.10}$$

evaluated at $(x_0, y_0)^\top$. The *first fundamental form* describes the inner product of two tangent vectors. It can be represented by the symmetric matrix

$$\mathbf{I} = \begin{pmatrix} E & F \\ F & G \end{pmatrix} = \begin{pmatrix} \langle \boldsymbol{s}_x, \boldsymbol{s}_x \rangle & \langle \boldsymbol{s}_x, \boldsymbol{s}_y \rangle \\ \langle \boldsymbol{s}_y, \boldsymbol{s}_x \rangle & \langle \boldsymbol{s}_y, \boldsymbol{s}_y \rangle \end{pmatrix}, \tag{3.11}$$

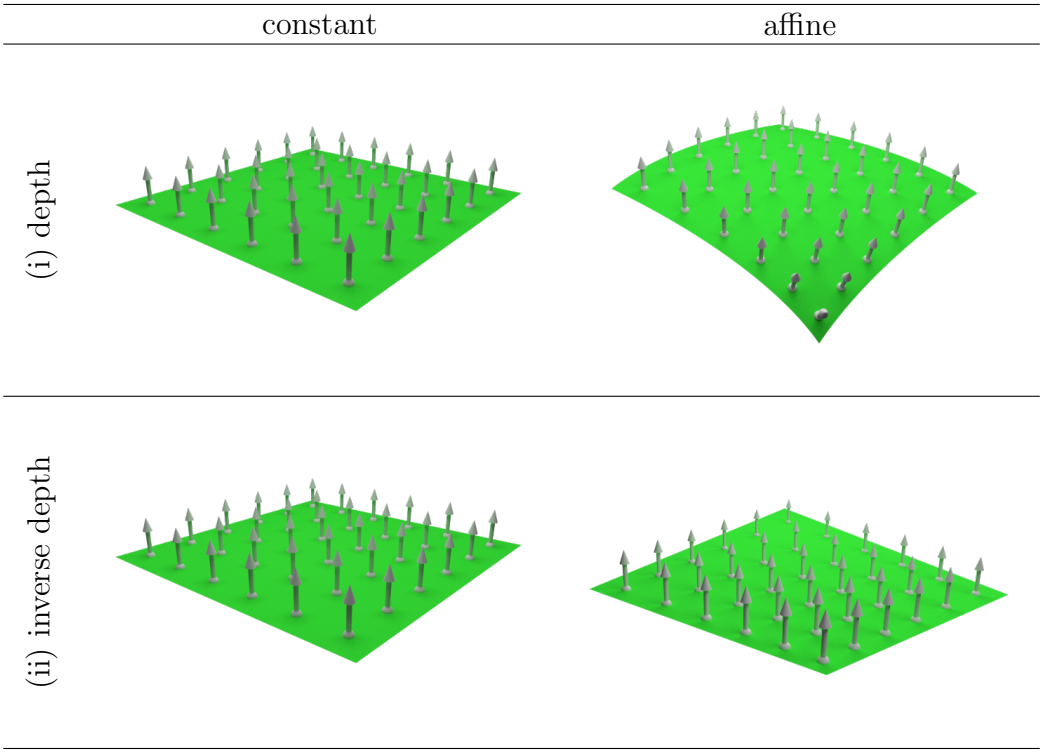


Figure 3.5: Resulting surfaces when backprojecting along the optical axis.

and allows the evaluation of metric properties such as the surface area. Similarly, the *second fundamental form* is important for describing curvatures. It can be represented by the symmetric matrix

$$\mathbf{II} = \begin{pmatrix} e & f \\ f & g \end{pmatrix} = \begin{pmatrix} \langle \boldsymbol{n}, \boldsymbol{s}_{xx} \rangle & \langle \boldsymbol{n}, \boldsymbol{s}_{xy} \rangle \\ \langle \boldsymbol{n}, \boldsymbol{s}_{yx} \rangle & \langle \boldsymbol{n}, \boldsymbol{s}_{yy} \rangle \end{pmatrix}, \tag{3.12}$$

where $\boldsymbol{n}$ is the unit surface normal

$$\boldsymbol{n} = \frac{\boldsymbol{s}_x \times \boldsymbol{s}_y}{|\boldsymbol{s}_x \times \boldsymbol{s}_y|}. \tag{3.13}$$

Figure 3.6 shows the tangent plane spanned by two tangent vectors along with the corresponding surface normal. The composition of the first and second fundamental form defines the *shape operator* [45]

$$\boldsymbol{S} = \mathbf{I}^{-1}\,\mathbf{II} = (EG - F^2)^{-1} \begin{pmatrix} eG - fF & fG - gF \\ fE - eF & gE - fF \end{pmatrix}, \tag{3.14}$$

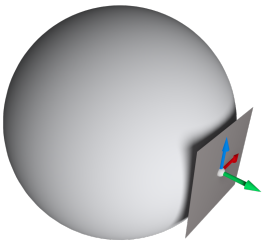


Figure 3.6: Tangent plane spanned by two tangent vectors $\boldsymbol{s}_x$ (red) and $\boldsymbol{s}_y$ (blue) along with the corresponding surface normal $\boldsymbol{n}$ (green).

which can be expressed in terms of the components of the first and second fundamental form. It allows to evaluate the Gaussian curvature

$$K = \det(\boldsymbol{S}) = \frac{\det(\mathbf{II})}{\det(\mathbf{I})} \tag{3.15}$$

and the mean curvature

$$H = \frac{1}{2} \cdot \operatorname{tr}(\boldsymbol{S}) = \frac{1}{2} \frac{eG - 2fF + gE}{EG - F^2}. \tag{3.16}$$

Direct Depth

Let us first consider the direct depth parameterisation where the unknown ρ corresponds to the sought depth. With this, we analyse the resulting surface in the case that the *depth is affine*: $\rho(\boldsymbol{x}) = \langle \boldsymbol{a}, \tilde{\boldsymbol{x}} \rangle$ with $\boldsymbol{a} = (a, b, c)^\top$. This is a reasonable and interesting case because a second order regulariser favours (piecewise) affine functions. For this case we obtain the two tangent vectors

$$\boldsymbol{s}_x = \boldsymbol{K}^{-1} \begin{pmatrix} \langle \boldsymbol{a}, \tilde{\boldsymbol{x}} \rangle + ax \\ ay \\ a \end{pmatrix} \quad \text{and} \quad \boldsymbol{s}_y = \boldsymbol{K}^{-1} \begin{pmatrix} bx \\ \langle \boldsymbol{a}, \tilde{\boldsymbol{x}} \rangle + by \\ b \end{pmatrix}, \tag{3.17}$$

such that

$$\hat{\boldsymbol{n}} = \boldsymbol{K}^\top \begin{pmatrix} -a \\ -b \\ 2ax + 2by + c \end{pmatrix} \tag{3.18}$$

points along the surface normal (3.13), i.e. $\hat{\boldsymbol{n}} = |\hat{\boldsymbol{n}}| \cdot \boldsymbol{n}$. Equation 3.18 shows that the normal direction depends on the location $\boldsymbol{x} = (x, y)^\top$ when backprojecting an affine depth function. To get deeper insights on how the surface normals vary, let us consider the surface curvature by means of the

shape operator. With (3.12), the second fundamental form for this example reads

$$\mathbf{II} = -\frac{2}{|\hat{\boldsymbol{n}}|}\begin{pmatrix} a^2 & ab \\ ab & b^2 \end{pmatrix}. \tag{3.19}$$

Since this matrix is singular, we can directly conclude that the determinant of the shape operator (3.14) and consequently the Gaussian curvature K is zero. This further implies that at least one of the principal curvatures is zero. To check if both principal curvatures are zero, let us additionally consider the mean curvature

$$\begin{aligned} H =& \frac{1}{2}\operatorname{tr}(\boldsymbol{S}) && (3.20)\\ =& -\frac{a^2|\boldsymbol{s}_y|^2 - 2ab\langle \boldsymbol{s}_x, \boldsymbol{s}_y\rangle + b^2|\boldsymbol{s}_x|^2}{\left(\det\left(\boldsymbol{K}^{-1}\right)\langle \boldsymbol{a}, \tilde{\boldsymbol{x}}\rangle\right)^2 |\hat{\boldsymbol{n}}|^3} && (3.21)\\ =& -\frac{|a\boldsymbol{s}_y - b\boldsymbol{s}_x|^2}{\left(\det\left(\boldsymbol{K}^{-1}\right)\langle \boldsymbol{a}, \tilde{\boldsymbol{x}}\rangle\right)^2 |\hat{\boldsymbol{n}}|^3} && (3.22)\\ =& -\frac{(ak_x)^2 + (bk_y)^2}{|\hat{\boldsymbol{n}}|^3}. && (3.23) \end{aligned}$$

Here, we have used the relation

$$\det(\mathbf{I}) = EG - F^2 = |\boldsymbol{s}_x \times \boldsymbol{s}_y|^2 \tag{3.24}$$

between the coefficients of the first fundamental form and the length of the cross product of both tangent vectors and the fact that

$$|\boldsymbol{s}_x \times \boldsymbol{s}_y| = \det\left(\boldsymbol{K}^{-1}\right)\langle \boldsymbol{a}, \hat{\boldsymbol{x}}\rangle\, |\hat{\boldsymbol{n}}|. \tag{3.25}$$

This shows that the mean curvature is in general not equal to zero, i.e. the surface is bent in one direction. Only for constant functions, i.e. with a and b equal to zero, we also obtain a vanishing mean curvature and thus, a planar surface.

Inverse Depth

Let us now consider the alternative parameterisation $\phi(r) = 1/r$. Then the unknown ρ corresponds to the *inverse depth*. Again we assume that the unknown, in this case the inverse depth, is affine. Accordingly, we obtain the two tangent vectors

$$\boldsymbol{s}_x = \frac{\boldsymbol{K}^{-1}}{\langle \boldsymbol{a}, \tilde{\boldsymbol{x}}\rangle^2}\begin{pmatrix} by + c \\ -ay \\ -a \end{pmatrix} \quad \text{and} \quad \boldsymbol{s}_y = \frac{\boldsymbol{K}^{-1}}{\langle \boldsymbol{a}, \tilde{\boldsymbol{x}}\rangle^2}\begin{pmatrix} -bx \\ ax + c \\ -b \end{pmatrix}, \tag{3.26}$$

Table 3.1: Preservation of planarity.

	direct depth		**inverse depth**	
	line of sight	optical axis	line of sight	optical axis
constant	no	yes	no	yes
affine	no	no	no	yes

such that

$$\hat{\boldsymbol{n}} = \boldsymbol{K}^\top \boldsymbol{a} \tag{3.27}$$

points along the surface normal (3.13). Thus, the surface normal points in the same direction in all considered cases for the inverse depth parameterisation. In other words, backprojecting an affine inverse depth always results in a planar surface. This imples that both the Gaussian and the mean curvature of the surface are zero.

Summary

Table 3.1 summarises the discussed findings for all four parameterisations. In conclusion, this shows that the inverse depth parameterisation along the optical axis is preferable w.r.t. the smoothness term when using a second order regularisation.

3.4.3 Linearity Analysis of the Data Term

Previously, we analysed the influence of parameterisations w.r.t. the smoothness term. Now we analyse its effects on the data term. Since the unknown ρ appears as argument of $\boldsymbol{f}_i$, the presented energy (3.1) is non-convex. To cope with this, most minimisation strategies perform a linearisation. In this regard, we analyse how the different depth parameterisations affect this linearisation. In particular, we are interested in the deviation from linearity of $\boldsymbol{g}_i$ in ρ because this quantity depends on the chosen parameterisation. As introduced in Section 3.3, the function $\boldsymbol{g}_i$ maps a location $\boldsymbol{x} \in \Omega$ in the reference frame $\boldsymbol{f}_0$ with its depth $(\phi \circ \rho)(\boldsymbol{x})$ to the corresponding location in another image $\boldsymbol{f}_i$. This mapping can be described as a composition of a backprojection (3.9) and a projection (3.4):

$$\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho) = \boldsymbol{\pi}\big(\boldsymbol{P}_i \cdot \tilde{\boldsymbol{s}}_i(\boldsymbol{x}, \phi \circ \rho)\big). \tag{3.28}$$

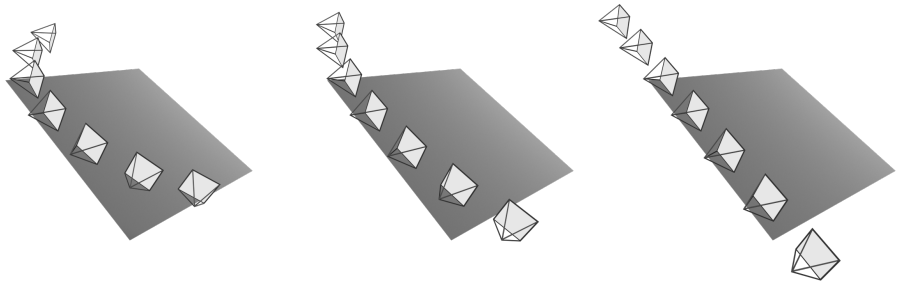


Figure 3.7: Three different camera setups that are common for multi-view setups. For all of them the Z-component of $\boldsymbol{t}_i$ is zero.

Since scaled homogeneous coordinates are equivalent, it is possible to multiply $\tilde{\boldsymbol{s}}_i$ by $(\phi \circ \rho)(\boldsymbol{x})^{-1}$ and rewrite Equation 3.28 as

$$\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho) = \boldsymbol{\pi}\left(\boldsymbol{K}_i \boldsymbol{R}_i \boldsymbol{K}^{-1}\, \tilde{\boldsymbol{x}} + \boldsymbol{K}_i \boldsymbol{t}_i\, (\phi \circ \rho)(\boldsymbol{x})^{-1}\right). \tag{3.29}$$

Direct Depth

For common setups, the camera offsets in Z-direction are much smaller than the occurring depth values. This is because one typically walks around an object mainly with lateral motion while roughly keeping the distance with only small rotations between views. This causes converging camera setups that keep the object in the middle of the view. Hence, we assume in the following analysis that the Z-component of $\boldsymbol{t}_i$ is zero. Please note the relation $\boldsymbol{t} = -\boldsymbol{R}\boldsymbol{c}_i$ between $\boldsymbol{t}_i$ and the camera centre $\boldsymbol{c}_i$ and that setting the Z-component of $\boldsymbol{t}_i$ to zero does not restrict us to camera motions in the X-Y-plane. Figure 3.7 shows three exemplary camera setups where the Z-component of $\boldsymbol{t}_i$ is zero. This allows to simplify (3.29) to

$$r_3^{-1} \cdot \left(\begin{pmatrix} r_1 \\ r_2 \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} (\phi \circ \rho)(\boldsymbol{x})^{-1} \right), \tag{3.30}$$

with the abbreviations $\boldsymbol{r} = \boldsymbol{K}_i \boldsymbol{R}_i \boldsymbol{K}^{-1} \tilde{\boldsymbol{x}}$ and $\boldsymbol{z} = \boldsymbol{K}_i \boldsymbol{t}_i$ that do not depend on $\rho(\boldsymbol{x})$. With the direct depth parameterisation $(\phi \circ \rho)(\boldsymbol{x})^{-1} = \rho(\boldsymbol{x})^{-1}$, we obtain a hyperbola and thus expect an additional linearisation error.

Inverse Depth

This is not the case for the inverse depth parameterisation with $(\phi \circ \rho)(\boldsymbol{x})^{-1} = \rho(\boldsymbol{x})$. In fact, Equation 3.30 reveals that $\boldsymbol{g}_i$ is linear in $\rho(\boldsymbol{x})$ in this case.

Thus, no error is introduced when linearising $\boldsymbol{g}_i$ w.r.t. the inverse depth. To summarise, also the linearisation analysis shows that an inverse depth parameterisation turns out to be more appropriate for multi-view depth estimation than the standard direct depth parameterisation.

3.5 Minimisation and Implementation

To solve the energy (3.1), we perform a linearisation around ρ_0 in the data term (3.2):

$$\boldsymbol{f}_i(\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho)) \approx \boldsymbol{f}_i(\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho_0)) + (\rho - \rho_0) \cdot \partial_\rho \boldsymbol{f}_i(\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho)|_{\rho=\rho_0}. \quad (3.31)$$

Applying the chain rule gives

$$\partial_\rho \boldsymbol{f}_i(\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho)) = \boldsymbol{\mathcal{J}}\boldsymbol{f}_i(\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho)) \cdot \boldsymbol{\mathcal{J}}\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho), \quad (3.32)$$

where the image derivatives $\boldsymbol{\mathcal{J}}\boldsymbol{f}_i$ are independent of the parameterisation. The second term in (3.32) is given by

$$\boldsymbol{\mathcal{J}}\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho) = \boldsymbol{\mathcal{J}}\boldsymbol{\pi} \begin{pmatrix} \boldsymbol{P}_i \begin{pmatrix} \boldsymbol{K}^{-1}\tilde{\boldsymbol{x}} \\ (\phi \circ \rho)(\boldsymbol{x})^{-1} \end{pmatrix} \end{pmatrix} \boldsymbol{K}_i \boldsymbol{t}_i \, \partial_\rho (\phi \circ \rho)(\boldsymbol{x})^{-1}, \quad (3.33)$$

where for the direct depth parameterisation

$$\partial_\rho (\phi \circ \rho)(\boldsymbol{x})^{-1} = -\rho(\boldsymbol{x})^{-2}, \quad (3.34)$$

and for the inverse depth parameterisation

$$\partial_\rho (\phi \circ \rho)(\boldsymbol{x})^{-1} = 1. \quad (3.35)$$

With the abbreviations

$$\boldsymbol{m}_i = \partial_\rho \boldsymbol{f}_i(\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho))|_{\rho=\rho_0} \quad (3.36)$$

and

$$\boldsymbol{b}_i = \boldsymbol{m}_i \, \rho_0 + \boldsymbol{f}_0(\boldsymbol{x}) - \boldsymbol{f}_i(\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho_0)) \quad (3.37)$$

the energy (3.1) with the linearised data term reads

$$E(\rho, \boldsymbol{w}) = \int_\Omega \frac{1}{n} \sum_{i=1}^{n} \Psi\left(|\boldsymbol{m}_i \, \rho - \boldsymbol{b}_i|^2\right) + \alpha \Big(\Psi\left(|\boldsymbol{\nabla}\rho - \boldsymbol{w}|^2\right) + \beta \, \Psi\left(|\boldsymbol{\mathcal{J}}\boldsymbol{w}|_F^2\right) \Big) \mathrm{d}\boldsymbol{x}. \quad (3.38)$$

Euler-Lagrange Equations

The minimiser of the linearised energy functional fulfils the corresponding Euler-Lagrange equations w.r.t. ρ and $\boldsymbol{w}$. With

$$\begin{aligned} \Psi'_{Di} &= \Psi'\left(|\boldsymbol{m}_i\rho - \boldsymbol{b}_i|^2\right), \\ \Psi'_{C} &= \Psi'\left(|\boldsymbol{\nabla}\rho - \boldsymbol{w}|^2\right), \\ \Psi'_{S} &= \Psi'\left(|\boldsymbol{\mathcal{J}}\boldsymbol{w}|_F^2\right), \end{aligned} \tag{3.39}$$

they are given by

$$\begin{aligned} \frac{1}{n}\sum_{i=1}^{n} \Psi'_{Di} \cdot \langle \boldsymbol{m}_i, \boldsymbol{m}_i\,\rho + \boldsymbol{b}_i\rangle - \alpha\ \mathrm{div}(\Psi'_C \cdot (\boldsymbol{\nabla}\rho - \boldsymbol{w})) &= 0, \\ \Psi'_C \cdot (w_1 - p_x) - \beta\ \mathrm{div}(\Psi'_S \cdot \boldsymbol{\nabla} w_1) &= 0, \\ \Psi'_C \cdot (w_2 - p_y) - \beta\ \mathrm{div}(\Psi'_S \cdot \boldsymbol{\nabla} w_2) &= 0 \end{aligned} \tag{3.40}$$

with boundary conditions $(\boldsymbol{\nabla}\rho - \boldsymbol{w})^\top \boldsymbol{n} = 0$ and $\boldsymbol{\mathcal{J}}\boldsymbol{w}\,\boldsymbol{n} = \boldsymbol{0}$, where $\boldsymbol{n}$ is the 2D outer normal here.

Implementation

We discretise (3.40) with finite differences on a regular grid using a standard discretisation. This results in a nonlinear system of equations, which we solve with two nested loops. While we update the nonlinear terms Ψ_D, Ψ_C, and Ψ_S (3.39) in the outer loop, we solve the linear system in the inner loop with the Fast-Jacobi algorithm [10]. Furthermore, we employ a coarse-to-fine approach to overcome linearisation errors.

3.6 Experimental Results

Our evaluation consists of two main parts. In the first part, we underpin our theoretical findings from Section 3.4 by means of experiments with synthetic data. Figure 3.8(a) shows a 3D scene with a planar surface and three cameras, and Figure 3.8(b) depicts the images captured with the corresponding cameras. Figure 3.8(c) and (d) show the computed reconstructions with a direct depth and an inverse depth parameterisation, respectively. We clearly see that performing second order regularisation on the *depth* introduces a bias towards curved surfaces as discussed in Section 3.4. In contrast, performing second order regularisation on the *inverse depth* does not introduce such as

Table 3.2: Root mean square errors for six data sets from the Middlebury benchmark [92].

	direct depth	inverse depth
Barn 1	0.67	**0.25**
Barn 2	1.48	**0.51**
Bull	0.50	**0.23**
Poster	0.48	**0.22**
Sawtooth	1.09	**0.43**
Venus	0.58	**0.29**

bias and thus yields a significantly better reconstruction. In Figure 3.8(c) and (d), we apply the following colour code to visualise the reconstruction errors: Green represents an error of zero, whereas red and blue correspond to behind and in front of the ground truth surface, respectively.

In the second part of our evaluation, we run tests on six publicly available multi-view data sets from the Middlebury benchmark [92] to obtain a quantitative comparison between both parameterisations. More specifically, we use five images for each 3D scene to compute the depth map. We have optimised the smoothness parameters α and β for each parameterisation, but kept them fixed over the individual scenes. Here we measure for both parameterisations the reconstruction quality in terms of the root mean square error in depth. Table 3.2 shows that the inverse depth parameterisation provides a significantly better reconstruction quality than the direct depth parameterisation in all cases. These quantitative experiments confirm our findings from Section 3.4. The inverse depth parameterisation does not only have advantages in theory, but also practically achieves superior reconstructions. Besides the discussed benefits for the linearisation and second order regularisation, there is another advantage that we have not stressed so far: It is a natural choice to initialise the inverse depth with zero. This corresponds to a depth of infinity. Thus, an initialisation of the direct depth with a large constant seems desirable but turns out to be problematic.

3.7 Limitations and Discussion

In contrast to modelling smoothness on the image domain, directly enforcing prior assumptions on the resulting depth surface is an alternative approach [56, 96, 44]. The benefit of such a strategy is that it renders the effects of regularisation on the resulting surface more explicit. This can be of advantage since one is interested in the appearance of the surface and not in an intermediate representation of depth.

However, we have shown that an inverse depth parameterisation allows to model the assumption that the scene is composed of piecewise planar surfaces by imposing a regulariser defined on the image domain. Since the smoothness assumption is defined on the image domain, a great variety of well known regularisers is available. Therefore, it is straightforward to introduce more sophisticated extensions such as ITGV [86, 35] or non-local variants [87]. On the other hand, it is usually more difficult to extend approaches that define the regularisation on the surface in a similar way. Thus, defining regularisation on the depth surface may introduce some additional freedom in the modelling but it is usually more cumbersome to deal with.

Next we would like to discuss how each of the multiple views captured influences the final result. In order to get a better understanding of this, it helps to interpret the depth estimation problem as a joint denoising and inpainting approach.

Interpretation as Joint Denoising and Inpainting. For simplicity, let us assume that we are dealing with grey value images. Now we consider the argument of Ψ in the linearised data term of Equation 3.38 in more detail. The argument then reads

$$(m_i \rho - b_i)^2 \,, \tag{3.41}$$

where $m_i = \boldsymbol{\nabla} \boldsymbol{f}_i(\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho))^\top \boldsymbol{\mathcal{J}} \boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho)$ (cf. Equations 3.36 and 3.32). It is clear that at locations where $m_i = 0$, the data term does not impose a constraint on the unknown ρ such that the depth values are solely filled in by the smoothness term in these regions. At locations where $m_i \neq 0$ we can rewrite the above expression as

$$m_i^2 \left(\rho - \frac{b_i}{m_i} \right)^2 \,. \tag{3.42}$$

Thus, the data term penalises deviations of ρ from b_i/m_i with a weight of m_i^2. In this way, the multi-view depth estimation problem exhibits the same

structure as a joint denoising and inpainting problem of the form

$$E(u) = \int_{\Omega} w\,(u - f)^2 \,\mathrm{d}\boldsymbol{x} + R(u), \tag{3.43}$$

where w is a confidence function, f some input data, and $R(u)$ a regularisation term. In this case we have $w = m_i^2$ and $f = {}^{b_i}/_{m_i}$ in the relevant locations where the confidence is larger than zero. By the form of m_i, we can observe that in the multi-view depth estimation scenario both the image gradient as well as the Jacobian of $\boldsymbol{g}_i$ determine the confidence.

Since the influence of the weighting induced by the image gradient and the effects of a normalisation have been discussed in related scenarios such as optical flow [101, 66, 127], let us focus on the second term $\boldsymbol{\mathcal{J}}\boldsymbol{g}_i(\boldsymbol{x}, \phi \circ \rho)$ in more detail. Intuitively speaking, the Jacobian of the mapping $\boldsymbol{g}_i$ allows to describe the rate of change along the epipolar line in the i-th image plane depending on the unknown ρ. Thus, in camera configurations where this rate of change is zero, the data term is automatically switched off. This case occurs if the i-th camera centre corresponds to the reference camera centre for example. Accordingly, camera pairs that allow a reliable estimation of the unknown ρ tend to be weighted higher than unsuitable pairs with a too small baseline.

Summing up, the interpretation as a joint denoising and inpainting problem allows to get a better insight of how the data term is weighted for different camera pairs in a multi-view depth estimation scenario and reveals that suitable camera pairs tend to be preferred while less reliable or degenerate camera pairs are weighted down or even completely switched off.

3.8 Summary

In this chapter, we have analysed different depth parameterisations within the context of multi-view depth estimation with higher order regularisation. Our first finding is that parameterisations along the line of sight are not suitable for such a scenario. In fact, we show that parameterisations along the optical axis are much more reasonable. For them, we present a detailed analysis of a direct depth and an inverse depth parameterisation. We point out several advantages of the inverse depth parameterisation: First, it is compatible with second order regularisation. Piecewise affine inverse depth leads to piecewise planar 3D surfaces. On the contrary, this is not the case for the direct depth parameterisation. It introduces a bias which we quantify both theoretically by means of the shape operator as well as by experiments.

Second, we show that an inverse depth parameterisation is not only advantageous for the smoothness term. It is also preferable for the linearisation required in the data term compared to a direct depth parameterisation. Last but not least, the inverse depth approach additionally admits a more meaningful initialisation. Based on our findings, we recommend the inverse depth parameterisation along the optical axis as the parameterisation of choice for variational multi-view depth estimation.

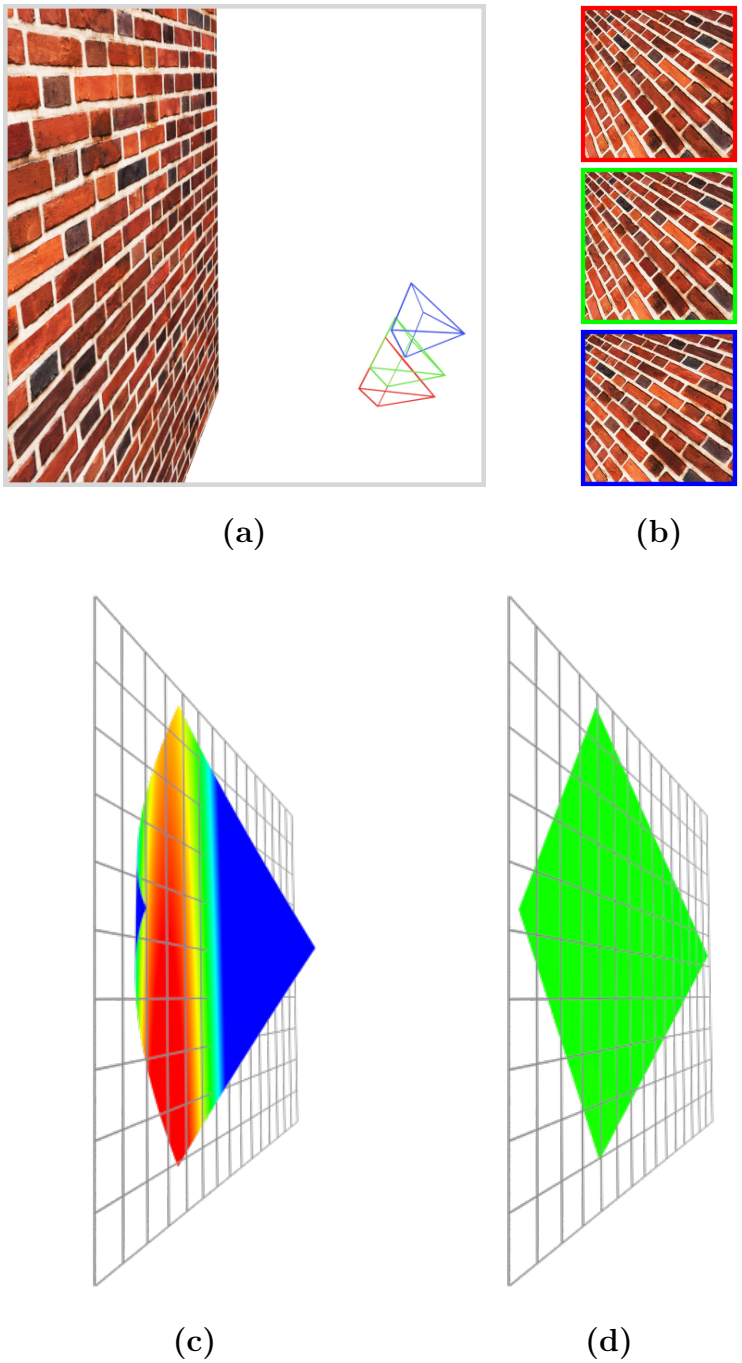


Figure 3.8: **From top left to bottom right**: **(a)** Camera setup and geometry. **(b)** Corresponding input images. **(c)** Reconstruction with direct depth parameterisation. **(d)** Reconstruction with inverse depth parameterisation. See text for details.

Chapter 4

Surface Reconstruction from Depth Maps

While we were concerned with computing a single depth map from multiple images in the last chapter, this chapter aims at combining multiple depth maps, also often referred to as range images, into a single 3D model as shown in Figure 4.1. Often both steps are performed subsequently in order to solve the *multi-view stereo* problem. However, since depth maps are becoming more readily available through devices such as the Kinect or time-of-flight cameras, the topic of range image integration itself has also attracted an increasing amount of attention [79]. Performing this integration can be difficult for several reasons: The range images may contain noise and outliers, parts of the surface can be missing when they have not been properly reached during the acquisition, and the sampling density might not be sufficient for a correct reconstruction.

A very promising approach is given by variational range image integration methods as propsed by Zach et al. [124]: They are able to deal with a substantial amount of noise and outliers, while regularising and thus creating smooth surfaces at the same time. In this chapter, we extend their state-of-the-art approach in several aspects and present an efficient GPU implementation. The main ideas of our work have been published in [6]. The GPU implementation has been published in [10] and presented at the NVIDIA GPU Technology conference 2014.

Organisation of this Chapter. After covering important related work, we describe the accurate computation of signed distance fields from range images in Section 4.3. Section 4.4 explains how the signed distance fields are integrated into a globally optimal cumulative signed distance field using anisotropic regularisation. Subsequently, Section 4.5 describes implementa-

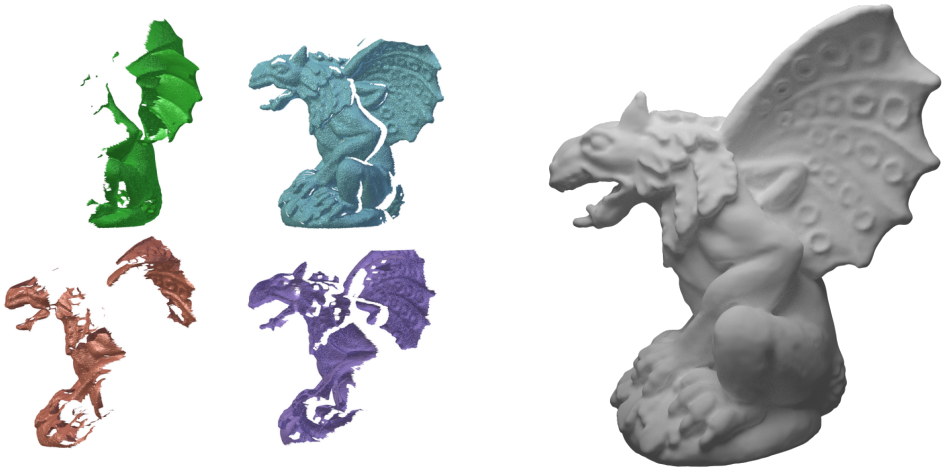


Figure 4.1: Surface reconstruction from depth maps aims at combining multiple registered depth maps (left) into a single 3D model (right). Here the depth maps are visualised in terms of the surface that they describe.

tion aspects concerning the computation and storage of signed distance fields, the discretisation, and efficient numerics on the GPU. We display experimental results in Section 4.6 before we discuss limitations in Section 4.7 and sum up the most important insights in Section 4.8.

4.1 Related Work

There is a vast amount of work that deals with the integration of registered depth maps into a unified surface representation. Bernadini and Rushmeier [16] order existing techniques based on how the surface is reconstructed. Their classification is comprised of four categories: Delauney based methods, surface based methods, deformable surfaces and volumetric methods. To focus on the most closely related techniques, we restrict ourselves to the class of volumetric methods in the following as our approach belongs to this category.

The goal of volumetric range image integration methods is to estimate an implicit representation of the desired surface. Such intermediate volumetric representations are often advantageous when integrating range images because they allow handling meshes of arbitrary genus. Furthermore, one can spare the effort of finding a suitable surface parameterisation. The implicit function is usually defined on a regular 3D grid that contains all registered

depth maps. Once it is found, the sought surface can be extracted as a level line of the implicit function using algorithms such as marching cubes [71]. Volumetric methods differ in how the implicit representation is obtained and we group them into two categories: The first category typically minimises an L^p data term possibly accompanied by a smoothness assumption. The second category comprises methods that are derived in a probabilistic way leading to binary labelling problems.

L^p Data Term. Curless and Levoy [29] estimate the signed distance for each voxel near the surface for each scan by casting a ray from the sensor through the voxel. This way they obtain an intersection of ray and surface and can infer the distance between intersection and voxel along the ray. Subsequently, the signed distance values from all scans are accumulated, where they are weighted depending on surface normal and viewing direction. Let us denote the 3D signed distance fields by $f_1(\boldsymbol{x}), \dots, f_n(\boldsymbol{x})$ and the weighting functions by $w_1(\boldsymbol{x}), \dots, w_n(\boldsymbol{x})$. Then the unified signed distance function $u(\boldsymbol{x})$ is computed as the pointwise weighted average

$$u(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} w_i(\boldsymbol{x}) f_i(\boldsymbol{x})}{\sum_{i=1}^{n} w_i(\boldsymbol{x})} \,. \tag{4.1}$$

This effectively minimises an energy with a weighted L^2 data term

$$E(u) = \sum_{i=1}^{n} \int_{\Omega_3} w_i(\boldsymbol{x}) \big(u(\boldsymbol{x}) - f_i(\boldsymbol{x})\big)^2 \, \mathrm{d}\boldsymbol{x} \,. \tag{4.2}$$

While Curless and Levoy are concerned with merging depth maps, Goesele et al. [43] embed this idea in a multi-view stereo context and are able to show convincing results in this setting. Hilton et al. [53] follow a similar approach as Curless and Levoy also minimising an energy as (4.2). However, these pointwise approaches suffer from several problems: A solution can only be computed at locations where data is given. Thus, Curless and Levoy have to employ a hole filling algorithm in a postprocessing step in order to deal with unseen portions of the surface. Still the resulting reconstruction does not necessarily have to be watertight. Furthermore, it is known that averaging without regularisation leads to inconsistent surfaces due to frequent sign changes within the cumulative signed distance field. Wheeler et al. [119] try to deal with incorrect surface locations by assigning the distance to the consensus surface to each voxel instead. The consensus surface is given by a weighted average of the nearby measurements. Zach et al. [124] address the previously mentioned problems by computing the cumulative signed distance

field as the global minimiser of a suitable energy functional that incorporates a total variation (TV) [89] smoothness term along with a robust L^1 data term.

In general, volumetric methods are well suited for large datasets because their memory requirement and time complexity mainly depends on the resolution of the implicit function. More recent works show reconstructions in real time [57] or of larger scenes [125]. When choosing a high resolution, this corresponds to small voxels which often lead to an unnecessarily large number of triangles in the resulting 3D model. On the other hand large voxels imply a loss of small scale features. An adaptive sampling employing octrees [85, 27] can be a remedy. Besides octrees, the voxel hashing algorithm of Nießner et al. [81] also offers an interesting alternative for dealing with large scenes.

Probabilistic Approaches. The basic idea of the probabilistic approaches is to find the most probable surface S that best explains some given input data $r_1, \ldots, r_n$ by maximising

$$E(S) = P(S \,|\, \{r_1, \ldots, r_n\}) \tag{4.3}$$

over the set of watertight surfaces lying inside the volume Ω_3 [69, 112, 63, 64]. Here, P denotes the probability for a surface S given a set of observations $\{r_1, \ldots, r_n\}$. The observations do not necessarily have to be limited to a set of scalar valued range images $r_i : \Omega_2 \to \mathbb{R}_+$. They can also contain additional confidence information or be comprised of colour images for example. The Bayes formula implies that

$$P(S \,|\, \{r_1, \ldots, r_n\}) \propto P(\{r_1, \ldots, r_n\} \,|\, S) \cdot P(S), \tag{4.4}$$

and since any constant factor larger than zero does not affect the solution, both expressions $P(S \,|\, \{r_1, \ldots, r_n\})$ and $P(\{r_1, \ldots, r_n\} \,|\, S) \cdot P(S)$ can be interchanged in this context. This allows to introduce the a priori probability $P(S)$ that prefers certain types of surfaces. A very basic choice is

$$P(S) = \exp\left(-\alpha \int_S \mathrm{d}s\right), \tag{4.5}$$

where α is a weighting parameter and $\int_S \mathrm{d}s$ corresponds to the Euclidean surface area. This allows to prefer smooth surfaces with simple topology. For a more general formulation, the Euclidean metric can be replaced by a Riemannian one [112, 64]. Since minimising for the surface area alone would

simply lead to the trivial solution given by the empty set, also the modelling of $P(\{r_1, \ldots, r_n\} \,|\, S)$ is a crucial issue. To this end, the 3D domain Ω_3 is decomposed into an interior region Ω_3^{in} corresponding to points lying inside an object and an exterior region Ω_3^{out} denoting the remaining points such that $\Omega_3 = \Omega_3^{in} \cup \Omega_3^{out}$. Furthermore, one usually assumes that observations of separate voxels are independent from each other and constructs functions $P_{in}(\boldsymbol{x})$ and $P_{out}(\boldsymbol{x})$ based on the input data. By taking the negative logarithm, the maximisation problem (4.3) can then be converted into the energy minimisation problem

$$E(S) = \int_{\Omega_3^{in}} \rho_{in}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} + \int_{\Omega_3^{out}} \rho_{out}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} + \alpha \int_S \mathrm{d}s, \tag{4.6}$$

with the regional terms $\rho_{in}(\boldsymbol{x}) = -\ln\big(P_{in}(\boldsymbol{x})\big)$ and $\rho_{out}(\boldsymbol{x}) = -\ln\big(P_{out}(\boldsymbol{x})\big)$. The simplest idea is to use constant functions for the regional terms which corresponds to *ballooning terms* as in [69, 112]. This can successfully prevent the empty set as a solution because the ballooning term allows to prefer surfaces of larger volume. However, it suffers from oversmoothing effects and tends to destroy small structures. Other works define the regional terms based on silhouettes [63] or based on depth maps [51, 62]. An equivalent formulation to (4.6) is given by introducing the implicit function u as the characteristic function of the surface interior :

$$\begin{aligned} E(u) = & \int_{\Omega_3} \rho_{in}(\boldsymbol{x}) u(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \\ & \int_{\Omega_3} \rho_{out}(\boldsymbol{x})(1 - u(\boldsymbol{x})) \mathrm{d}\boldsymbol{x} + \\ & \alpha \int_{\Omega_3} |\boldsymbol{\nabla} u(\boldsymbol{x})| \mathrm{d}\boldsymbol{x} \\ & \text{s.t. } u : \Omega_3 \to \{0, 1\}. \end{aligned} \tag{4.7}$$

Due to the constraint, this is a non-convex optimisation problem since it is carried out over the set of binary functions. However, it was shown that it is possible to relax the problem by minimising over all functions that fulfil $u : \Omega_3 \to [0, 1]$ and subsequently threshold the minimiser of the relaxed problem at some value within $(0, 1)$ in order to obtain the minimiser of (4.7) [26, 63]. This is an interesting result because such relaxation techniques often sacrifice optimality in the sense that the thresholded solution is not optimal for the original binary labelling problem. However, here the constraint can

be relaxed without this disadvantage and one only has to solve a convex problem of the form

$$E(u) = \int_{\Omega_3} g(\boldsymbol{x}) \cdot u(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} + \alpha \int_{\Omega_3} |\boldsymbol{\nabla} u(\boldsymbol{x})| \, \mathrm{d}\boldsymbol{x} \tag{4.8}$$

s.t. $u : \Omega_3 \to [0, 1]$ with $g(\boldsymbol{x}) = \rho_{in}(\boldsymbol{x}) - \rho_{out}(\boldsymbol{x})$. However, it was noted that for such models the data term strongly pushes to the interval boundaries and Zach [123] has shown that this leads to aliasing artefacts.

4.2 Contributions

We extend the variational range image integration approach presented by Zach et al. [124] in several aspects. The isotropic (space-variant) diffusion term is replaced by an anisotropic (direction-dependent) one, which is designed to smooth along the evolving surface and evolving ridges in the cumulative signed distance field but not across. This way it is possible to obtain very smooth surfaces from noisy range images while preserving ridges and corners. In contrast to Zach et al. and our previous work [94], we do not use signed distances along the line of sight when converting range images into 3D distance fields. Instead, we compute the Euclidean signed distance to the range surface. Furthermore, we adopt a more efficient storage of signed distance values and we employ a different discretisation and a new numerical solver called Fast Jacobi. This allows for a fast parallel implementation which we experimentally demonstrate by means of a GPU using CUDA. Altogether, these changes allow for state-of-the-art results in the Middlebury benchmark at a very competitive runtime.

4.3 Signed Distance Fields

A range image maps each location of the image domain $\Omega_2 \subset \mathbb{R}^2$ to a depth value, which describes the distance from the camera centre to the surface of the scene along the corresponding optical ray. Let us assume that a range image $r : \Omega_2 \to \mathbb{R}_+$ and the corresponding camera matrix

$$\boldsymbol{P} = \boldsymbol{K} \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \end{pmatrix} \tag{4.9}$$

are given. In volumetric range image integration methods [29, 124, 123], a range image r is converted into a 3D signed distance field f by computing the signed distance ℓ of a point $\boldsymbol{x} \in \Omega_3 \subset \mathbb{R}^3$ along the line of sight. Computing

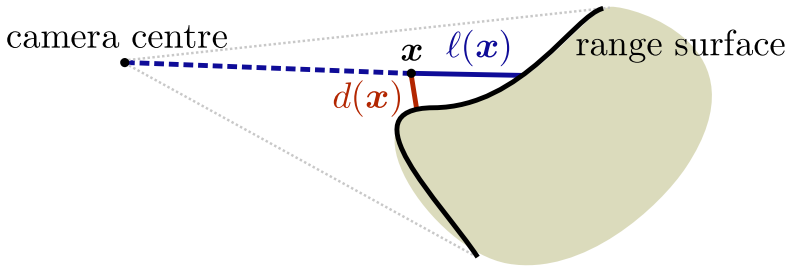


Figure 4.2: Generally, the directional signed distance $\ell(\boldsymbol{x})$ overestimates the Euclidean signed distance $d(\boldsymbol{x})$ to the range surface $\mathcal{S}$ and is a less accurate approximation of the true distance to the object.

this *directional signed distance* is computationally inexpensive because it can directly be evaluated as

$$\ell(\boldsymbol{x}) = r\big(\boldsymbol{\pi}(\boldsymbol{P}\tilde{\boldsymbol{x}})\big) - |\boldsymbol{x} - \boldsymbol{c}|, \tag{4.10}$$

where $\boldsymbol{c}$ corresponds to the camera centre and locations in front of the surface are arbitrarily given a positive value. By $\tilde{\boldsymbol{x}}$, we denote the homogeneous version of $\boldsymbol{x}$ and $\boldsymbol{\pi}$ maps from homogeneous to Euclidean coordinates. However, Figure 4.2 illustrates that the directional signed distance $\ell(\boldsymbol{x})$ generally overestimates the Euclidean distance. Although directional distance and Euclidean distance can coincide at certain locations, e.g. at the range surface itself, problems occur when averaging and regularising multiple directional signed distance values. Therefore, we propose to use a more accurate approximation of the Euclidean distance to the object by computing the *Euclidean signed distance* to the range surface $\mathcal{S}$:

$$d(\boldsymbol{x}) = \operatorname{sgn}\big(\ell(\boldsymbol{x})\big) \cdot \inf_{\boldsymbol{y} \in \mathcal{S}} |\boldsymbol{x} - \boldsymbol{y}|. \tag{4.11}$$

Since the range image and its corresponding projection are given, the range surface $\mathcal{S}$ can directly be evaluated. The sign of the Euclidean distance is determined by the sign of the directional signed distance. Alternatively, the sign could also be determined using range surface normals as in [119].

We follow [124] for the remaining part of this section by scaling the signed distance values with a factor of $1/\delta$ and truncating them to the interval $[-1, 1]$:

$$f(\boldsymbol{x}) = \psi\big(d(\boldsymbol{x})\big) \quad \text{with} \quad \psi(d) = \begin{cases} \operatorname{sgn}(d) & \text{if } |d| \geq \delta \\ d/\delta & \text{else.} \end{cases} \tag{4.12}$$

The parameter δ thus reflects the expected uncertainty of the depth values. We also use a binary weight $w : \Omega_3 \to \{0, 1\}$ associated with the signed distance field f in order to assign low confidence to f at locations behind the surface where $d(\boldsymbol{x}) < -\eta$. The parameter $\eta > 0$ thus specifies how much of the occluded region behind a surface is assumed to be solid.

4.4 Variational Signed Distance Field Integration

The cumulative signed distance function $u : \Omega_3 \to \mathbb{R}$ is computed as the minimiser of the energy

$$E(u) = \int_{\Omega_3} \Big(D(\boldsymbol{f}, \boldsymbol{w}, u) + \alpha \, S(\boldsymbol{\nabla} u) \Big) \, d\boldsymbol{x}, \tag{4.13}$$

containing n signed distance fields $\boldsymbol{f} = (f_1, \ldots, f_n)^\top$ and the associated weights $\boldsymbol{w} = (w_1, \ldots, w_n)^\top$. The *data term* $D(\boldsymbol{f}, \boldsymbol{w}, u)$ models the assumption that u should be similar to all signed distance fields $\boldsymbol{f}$, while the *smoothness term* or *regulariser* enforces u to be smoothly varying in space by penalising large gradients of u. Its influence is steered by the smoothness weight $\alpha > 0$. The desired surface geometry is then given by the zero level line of the global minimiser u. Figure 4.3 illustrates a slice of such a global minimiser u for a given 3D model.

Zach et al. [124] employ a robust L^1 data term along with a total variation (TV) smoothness term, such that data term and smoothness term are not continuously differentiable and the resulting energy is not strictly convex. Therefore, they introduce an auxiliary variable and solve a convex approximation of this energy using a numerical scheme that combines the duality principle for the TV term with a pointwise optimisation step.

4.4.1 Minimisation

Alternatively, it is possible to replace the absolute value function by the continuously differentiable and strictly convex approximation $\Psi_D(s^2) = \Psi_S(s^2) = \sqrt{s^2 + \epsilon^2}$ with a small regularisation constant $\epsilon > 0$ yielding the data term

$$D(\boldsymbol{f}, \boldsymbol{w}, u) = \sum_{i=1}^{n} w_i \, \Psi_D\big((u - f_i)^2\big) \tag{4.14}$$

and the smoothness term

$$S(\boldsymbol{\nabla} u) = \Psi_S\big(|\boldsymbol{\nabla} u|^2\big). \tag{4.15}$$

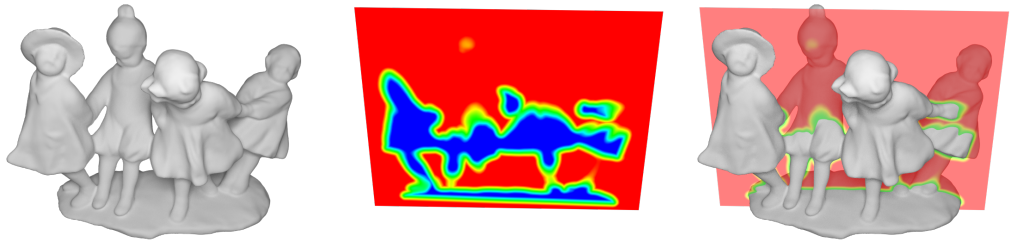


Figure 4.3: **Left:** Dancing children 3D model taken from Berger et al. [15]. **Middle:** A single xy-slice of the cumulative scaled and truncated signed distance function u. **Right:** Overlay of both.

The resulting energy approximates the TV-L^1 energy by Zach et al. and is strictly convex. Its minimiser necessarily fulfils the Euler-Lagrange equation

$$D_u(\boldsymbol{f}, \boldsymbol{w}, u) - \alpha \operatorname{div}\left(S_{\boldsymbol{\nabla} u}(\boldsymbol{\nabla} u)\right) = 0 \tag{4.16}$$

with the corresponding natural boundary condition

$$\langle \boldsymbol{n}, S_{\boldsymbol{\nabla} u}(\boldsymbol{\nabla} u) \rangle = 0, \tag{4.17}$$

where $\boldsymbol{n}$ denotes the outer normal and $S_{\boldsymbol{\nabla} u} = (S_{u_x}, S_{u_y}, S_{u_z})^\top$. When introducing the abbreviations $\Psi'_{i,D} := \Psi'_D\big((u - f_i)^2\big)$ and $\Psi'_S := \Psi'_S\big(|\boldsymbol{\nabla} u|^2\big)$, data and smoothness term derivatives are given by

$$D_u(\boldsymbol{f}, \boldsymbol{w}, u) = 2\left(u \sum_{i=1}^{n} w_i \, \Psi'_{i,D} - \sum_{i=1}^{n} w_i \, \Psi'_{i,D} \, f_i\right) \tag{4.18}$$

and

$$S_{\boldsymbol{\nabla} u}(\boldsymbol{\nabla} u) = 2 \, \Psi'_S \boldsymbol{\nabla} u. \tag{4.19}$$

4.4.2 Isotropic Regularisation

This choice of smoothness term leads to an isotropic regularisation: It is space-variant but not direction-dependent. As stated, one obtains TV regularisation for the choice of $\epsilon = 0$. Especially in the case of 3D reconstruction, this allows for a nice geometric interpretation of the regulariser. To this end, let us first define the characteristic function χ_A of a set $A \subset \Omega$ as

$$\chi_A(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in A \\ 0 & \text{else.} \end{cases} \tag{4.20}$$

This allows to define the perimeter of A by means of the total variation of the indicator function, i.e. $\mathrm{Per}(A) = \mathrm{TV}(\chi_A)$. Now the co-area formula by Federer [34] and Fleming and Rishel [37] states that

$$\mathrm{TV}(u) = \int_{-\infty}^{+\infty} \mathrm{Per}\big(A(u,\lambda)\big)\,\mathrm{d}\lambda, \tag{4.21}$$

where $A(u,\lambda) = \{\boldsymbol{x} \in \Omega \mid u(\boldsymbol{x}) < \lambda\}$ corresponds to the λ-level set of u. This shows that the TV regulariser sums up the length of all level lines. For this reason, TV regularisation leads to minimal surfaces because it penalises the perimeter of the level sets of u [24]. In the setting of 3D reconstruction, the perimeter of the zero level line of the implicit function u exactly corresponds to the surface area of the 3D object. Increasing the smoothness weight α thus results in reducing isolated small scale features and generating low-genus isosurfaces instead of an increased smoothing of u. However, the space-variant diffusivity Ψ_S' ignores the surface orientation. Incorporating an orientation dependent behaviour requires anisotropic smoothing [115], which is discussed next.

4.4.3 Anisotropic Regularisation

In order to obtain an anisotropic smoothing behaviour, we modify the diffusion term in the Euler-Lagrange equation (4.16) by replacing the smoothness term derivative with

$$S_{\boldsymbol{\nabla} u}(\boldsymbol{\nabla} u) = 2\,\Psi_S'(\boldsymbol{J}_{\rho,\sigma})\boldsymbol{\nabla} u. \tag{4.22}$$

This essentially lifts the idea of Zimmer et al. [126], who modelled an anisotropic disparity-driven stereo vision, to three dimensions. The matrix-valued function Ψ_S' is an extension of a scalar-valued function that is applied only to the eigenvalues while leaving the eigenvectors unchanged, and

$$\boldsymbol{J}_{\rho,\sigma} := K_\rho * (\boldsymbol{\nabla} u_\sigma \boldsymbol{\nabla} u_\sigma^\top) \tag{4.23}$$

is the *structure tensor* [38]. Here, $u_\sigma := K_\sigma * u$, where $*$ denotes a convolution with a Gaussian K_σ of standard deviation σ.

Apparently, the structure tensor $\boldsymbol{J}_{\rho,\sigma}$ extends the tensor product $\boldsymbol{\nabla} u \boldsymbol{\nabla} u^\top$ in two aspects: On the one hand, u is replaced by a smoothed version u_σ, which is obtained by performing a Gaussian convolution with standard deviation σ on u. Thus, σ acts like a *noise scale* because high frequencies are attenuated due to the low-pass effect of Gaussian convolution. However, also

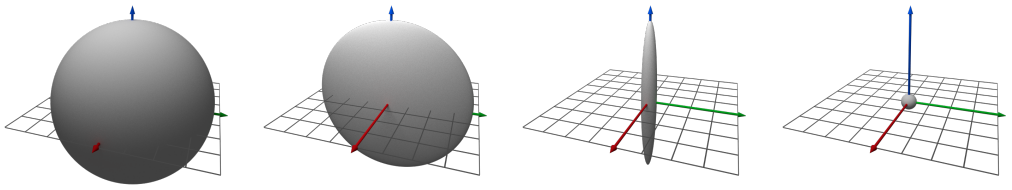


Figure 4.4: Visualisation of diffusion tensors as ellipsoids for different local structures.

$\boldsymbol{\nabla} u_\sigma$ is still sensitive to noise in case σ is chosen too small. If one selects a σ that is too large, this will result in cancellation effects which is also not desirable. As a remedy, another Gaussian convolution that acts on the tensor entries is introduced. This poses the second difference between the tensor product $\boldsymbol{\nabla} u \boldsymbol{\nabla} u^\top$ and the structure tensor. We denote the standard deviation of the convolution acting on the tensor entries by ρ. It describes the window size over which the orientations are aggregated and is commonly referred to as *integration scale*.

Let us now discuss how the anisotropic smoothing behaviour adapts to the local structure by considering the eigenvalues of the diffusion tensor $\Psi'_S(\boldsymbol{J}_{\rho,\sigma})$ for the following four cases:

(I) In homogeneous regions, all eigenvalues are equally large, which causes homogeneous smoothing in all three directions.

(II) At smooth surfaces, one eigenvalue is close to zero, which leads to anisotropic smoothing along the surface but not across.

(III) At ridges, i.e. oriented 1D structures in 3D space, only one eigenvalue is large, resulting in smoothing along the ridge.

(IV) At corners, all eigenvalues vanish, which prevents smoothing.

Figure 4.4 visualises the diffusion tensors as ellipsoids, where the eigenvectors correspond to the semi-principal axes and the eigenvalues to the respective equatorial radii.

The anisotropic regularisation can be interpreted as a generalisation of the isotropic model, and it is straightforward to show that both coincide if $\sigma = \rho = 0$. In this case, the structure tensor $\boldsymbol{J} = \boldsymbol{\nabla} u \boldsymbol{\nabla} u^\top$ has rank 1. Thus, its eigenvalues λ_2 and λ_3 vanish, and the remaining eigenvalue λ_1 is given

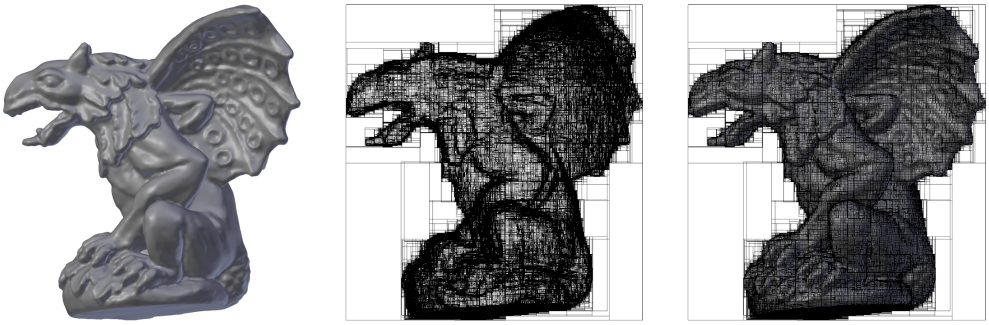


Figure 4.5: **From left to right: (a)** 3D model. **(b)** Bounding boxes of the bounding volume hierarchy. **(c)** Overlay of 3D model and bounding volume hierarchy.

by $\lambda_1 = \text{trace}(\boldsymbol{J}) = |\boldsymbol{\nabla} u|^2$. Its corresponding eigenvector is the normalised image gradient $\boldsymbol{v}_1 = \frac{\boldsymbol{\nabla} u}{|\boldsymbol{\nabla} u|}$. With this one can see that

$$\Psi_S'(\boldsymbol{J})\boldsymbol{\nabla} u = \Psi_S'(\boldsymbol{\nabla} u \boldsymbol{\nabla} u^\top)\boldsymbol{\nabla} u \tag{4.24}$$

$$= \sum_{i=1}^{3} \Psi_S'(\lambda_i)\boldsymbol{v}_i \boldsymbol{v}_i^\top \boldsymbol{\nabla} u \tag{4.25}$$

$$= \Psi_S'(|\boldsymbol{\nabla} u|^2)\frac{\boldsymbol{\nabla} u \boldsymbol{\nabla} u^\top}{|\boldsymbol{\nabla} u||\boldsymbol{\nabla} u|}\boldsymbol{\nabla} u \tag{4.26}$$

$$= \Psi_S'(|\boldsymbol{\nabla} u|^2)\boldsymbol{\nabla} u, \tag{4.27}$$

where we use the fact that $\boldsymbol{v}_2^\top \boldsymbol{\nabla} u = \boldsymbol{v}_3^\top \boldsymbol{\nabla} u = 0$, since the eigenvectors are orthonormal.

4.5 Implementation

First we explain how signed distance fields can be efficiently computed and stored. Subsequently, we focus on the numerical solution of the partial differential equation (PDE) (4.16) that allows to integrate the individual signed distance fields into a single 3D model. This also comprises the discretisation as well as important aspects for a GPU implementation.

4.5.1 Signed Distance Fields

An axis aligned bounding box that contains all range surfaces is chosen as domain of integration Ω_3. It can be discretised by choosing a number of

equidistant samples $\boldsymbol{N} = (N_1, N_2, N_3)^\top$ in each direction, resulting in the sampling distances $\boldsymbol{h} = (h_1, h_2, h_3)^\top$ and $N = N_1 \cdot N_2 \cdot N_3$ unknowns.

Efficient Computation of Signed Distance Fields. In order to set up one of the multiple signed distance fields, we have to compute the distance from a point to a triangle $N \cdot M$ times when assuming that a range surface is discretised by M triangles. This complexity of $\mathcal{O}(N \cdot M)$ causes severe problems because common resolutions of 200^3 voxels and $6 \cdot 10^5$ triangles require almost $5 \cdot 10^{12}$ computations. The resulting complexity essentially consists of two parts, N and M. Thus, we use two strategies, each dealing with one of the factors in order to greatly reduce the computational effort when setting up the signed distance fields.

First, we use the directional signed distance as a heuristic for closeness to the surface and only compute the Euclidean distance if $|\ell_i(\boldsymbol{x})| < c \cdot \delta$ for some $c \geq 1$. With this heuristic, we only need to compute the distance to all M triangles in a much smaller number of locations $\hat{N}$, where typically $\hat{N} \ll N$. Furthermore, we also expect $\hat{N}$ to grow much slower than N. Increasing the volume resolution in each dimension, we expect $\hat{N}$ to grow quadratically opposed to N, which grows cubically. Thus, we denote the resulting complexity by $\mathcal{O}(\hat{N} \cdot M)$.

In order to account for the second factor, we accelerate the computation of the Euclidean distance for a single voxel by organising the range surface in a bounding volume hierarchy (BVH) in order to bring the complexity towards $\mathcal{O}(\hat{N} \cdot \log(M))$ [84]. When choosing the shape of the bounding volume, one has to take several factors into account. On the one hand, storing them and computing distances to them should be cheap. On the other hand, we would like to fit our input data tightly so that we avoid traversing uninteresting subtrees. It hast turned out, that using axis aligned bounding boxes is often preferable to using other shapes in practice. Thus, we also opt for axis aligned bounding boxes.

For the BVH construction we employ a top-down approach. Starting with the whole object, we fit an axis aligned bounding box to all triangles. Then we split the bounding box in halves along the dimension where it has the largest extent. We recursively repeat this logic for each of the bounding boxes and stop recursing if a bounding box contains less triangles than a specified maximal number of primitives or if the bounding box becomes too small. Figure 4.5 shows a triangle mesh and the corresponding bounding volume hierarchy.

Having such a bounding volume hierarchy, it is much more efficient to compute the distance from a given point in space to the 3D mesh as the

bounding boxes allow to quickly evaluate the smallest and the largest distance possible to any triangle contained in them. This allows to prune a bounding box and all its children as soon as its smallest distance is larger than the largest distance of another bounding box. Note that a distance transform with a quadratic structuring element in principle also allows for an efficient evaluation of a distance function because it is separable. Unfortunately, this is not directly applicable in this case because the distance transform cannot directly work on the range images.

Efficient Storage of Signed Distance Fields. Storing all signed distance fields and their associated weights directly requires a huge amount of memory. This is necessary to evaluate the nonlinearities stemming from the data term

$$D(\boldsymbol{f}, \boldsymbol{w}, u) = \sum_{i=1}^{n} w_i \, \Psi_D\big((u - f_i)^2\big) \tag{4.28}$$

when Ψ_D is subquadratic. In order to reduce the memory requirement, one can either employ a voxelwise runlength encoding [124] or a coarser quantisation of the signed distances leading to a histogram based approach as in [123]. Here one uses the fact that the signed distance fields are truncated to the interval $[-1, 1]$ and samples this interval evenly to obtain m bins with centres c_j. This allows to count the occurrences of measurement values $\eta_j(\boldsymbol{x})$ in each bin and to replace the data term (4.28) by

$$D^H(\boldsymbol{\eta}, \boldsymbol{c}, u) = \sum_{j=1}^{m} \eta_j \, \Psi_D\big((u - c_j)^2\big). \tag{4.29}$$

In this approximation, the measured signed distance $f_i(\boldsymbol{x})$ is effectively replaced by its closest bin centre. Choosing the number of bins m allows to tune the accuracy of the approximation and usually a quite small number of bins, such as 8 bins, yields a good approximation. Instead of having to keep all input distance fields $\boldsymbol{f}$, one only has to maintain a histogram with m bins in each voxel and thus is independent of the number of input images used in the minimisation step. This allows to efficiently solve problems containing several hundred range images. Furthermore, the number of measurement values in each bin $\eta_j(\boldsymbol{x})$ is usually a sufficiently small integer number that is cheaper to store than a floating point number.

Another quite drastic approach to cut down on memory requirements is given by adjusting the data term to enforce similarity to the pointwise weighted median of all measurement values at a given location. This can also yield acceptable results and drastically reduce memory requirement.

As the signed distance fields are truncated to the interval $[-1, 1]$, it is possible to implicitly encode a weight of zero by using a value outside of this interval.

4.5.2 Numerical Solution

The elliptic PDE (4.16) allows to find a single signed distance function that approximates all individual input fields while adhering to a well suited smoothness assumption as discussed in Section 4.4.3. Its numerical solution is a crucial part within the implementation and may consume by far the most computation time compared to the other steps when done inefficiently. It is possible to write (4.16) as

$$p(u)\, u - \alpha \operatorname{div}\left(\boldsymbol{D}\, \boldsymbol{\nabla} u\right) \;=\; q(u)\;, \tag{4.30}$$

where $u : \Omega \to \mathbb{R}$ is the unknown solution and $p(u)$ and $q(u)$ are the real-valued functions

$$p(u) = \sum_{i=1}^{n} w_i\, \Psi'_{i,D} \quad \text{and} \quad \sum_{i=1}^{n} w_i\, \Psi'_{i,D}\, f_i \tag{4.31}$$

with the abbreviation $\Psi'_{i,D} := \Psi'_{i,D}\left((u - f_i)^2\right)$. Furthermore $\boldsymbol{D} = \Psi'_S(\boldsymbol{J})$ is the diffusion tensor, where $\boldsymbol{J} \in \mathbb{R}^{3\times 3}$ is the 3-D structure tensor, and the matrix valued function Ψ'_S yields the anisotropic behaviour.

Discretisation. After the discretisation on a regular grid, this corresponds to a nonlinear system

$$\left(\boldsymbol{P}(\boldsymbol{u}) - \alpha\, \boldsymbol{A}(\boldsymbol{u})\right) \boldsymbol{u} \;=\; \boldsymbol{q}(\boldsymbol{u}) \tag{4.32}$$

with N equations, where N is the number of voxels. The vectors $\boldsymbol{u}, \boldsymbol{q}(\boldsymbol{u}) \in \mathbb{R}^N$ are obtained by a spatial discretisation of the functions u and $q(u)$, respectively. Moreover, $\boldsymbol{P}(\boldsymbol{u}) := \operatorname{diag}(\boldsymbol{p}(\boldsymbol{u})) \in \mathbb{R}^{N\times N}$ with the discrete version $\boldsymbol{p}(\boldsymbol{u}) \in \mathbb{R}^N$ of $p(u)$. The matrix $\boldsymbol{A}(\boldsymbol{u}) \in \mathbb{R}^{N\times N}$ is the 3D discrete divergence operator. For its discretisation, we use the nonstandard approach described by Weickert et al. [118] applied to the three dimensional setting. Essentially, it uses the fact that the discrete divergence term $\boldsymbol{A}(\boldsymbol{u})$ has the continuous counterpart $\operatorname{div}(\boldsymbol{D}\boldsymbol{\nabla} u)$ which is the negative functional gradient of the energy

$$G(u) = \frac{1}{2} \int_\Omega \boldsymbol{\nabla} u^\top \boldsymbol{D} \boldsymbol{\nabla} u \, \mathrm{d}\boldsymbol{x}. \tag{4.33}$$

Thus, one can first discretise (4.33) and then obtain $\boldsymbol{A}(\boldsymbol{u})$ as the negative gradient of this discrete energy. The exact choice of discretisation is based on a private communication with Joachim Weickert and Martin Welk.

Parallel Implementation on the GPU. In order the solve (4.32), we use the fixed point iteration

$$\boldsymbol{u}^{k+1} \;=\; \big(\boldsymbol{P}(\boldsymbol{u}^k) - \alpha\, \boldsymbol{A}(\boldsymbol{u}^k)\big)^{-1}\, \boldsymbol{q}(\boldsymbol{u}^k) \tag{4.34}$$

for $k \in \mathbb{N}$ and solve each resulting linear system using the Fast Jacobi algorithm described in Section 2.4.2. It is perfectly suited for parallelisation since in one iteration the voxels can be processed independent from each other. We use CUDA and employ 3D textures and 3D surfaces for efficient processing and good caching behaviour in 3D space. The old solution is bound to a texture for efficient read operations while the new one is bound to a surface for efficient write operations. This has shown to be the favourable approach in terms of implementation effort and computational efficiency when compared to using global memory or shared memory. It is also straightforward to compute the stencil weights corresponding to the entries of $\boldsymbol{A}(\boldsymbol{u}^k)$ in parallel. On the GPU the amount of available memory is often a bottleneck. Although the stencil contains $3^3 = 27$ weights for each voxel, which indicates a large memory requirement, we know that the central weight equals the negative sum of all other weights and due to symmetry only half of the remaining 26 weights need to be stored.

Initialisation. Obviously, we can reach an accurate solution in less iterations when we start with a better initial guess of the solution. To this end, one can observe that the minimiser u is given by the pointwise weighted median of the signed distance fields when ignoring the smoothness term and setting $\varepsilon = 0$. In general, the number of iterations required for convergence can be greatly reduced when using this as an initialisation compared to an initialisation with a constant value $z \in [-1, 1]$. If a voxel has never been seen and the bounding box is chosen rather tight, it is most probable that the voxel lies inside the object such that it is reasonable to initialise it with -1. Alternatively, a coarse-to-fine approach can also be interpreted as a means of obtaining a better initialisation on the finest level with a quite low computational effort. It is well known that coarse-to-fine approaches can be employed to speed up the convergence and in this case it depends on the input data and the choice of the regularisation parameter α, which strategy yields faster convergence.

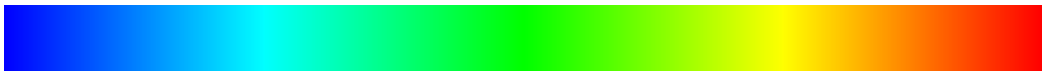


Figure 4.6: Colour code for reconstruction errors. Values of blue, green, and red denote locations in front, on, and behind the ground truth surface, respectively.

4.6 Experimental Results

First, we will cover the error measure commonly used in surface reconstruction and give some general guidelines for choosing the available parameters. Then experiments show the benefit of using the Euclidean instead of the directional distance and an anisotropic instead of an isotropic regularisation. Last but not least, we will demonstrate the advantage of a parallel GPU implementation in terms of runtime.

Error Measure

Let us assume that both the 3D reconstruction and a ground truth model are available as triangle meshes. Let us furthermore denote the reconstruction by R and the ground truth mesh by G. In order to determine the accuracy of the reconstruction, one would like to know the (signed) distances from all points on R to G. However, in practice R is simply sampled at its vertices [95]. A very insightful error visualisation is given by colour coding the reconstructed mesh according to the error values. This allows to identify locations where the reconstruction exhibits large errors and by considering the sign of the distances in the colour code, one can also judge at which locations a reconstruction tends to over- or underestimate the correct shape. Figure 4.6 depicts the colour code that we use. Blue corresponds to locations that lie behind the ground truth and red denotes locations in front of the ground truth. Green indicates that ground truth and reconstruction coincide.

In order to allow for a ranking, one usually aggregates the errors measured at each vertex in a statistic. Most commonly one reports the *accuracy* as a distance belonging to a certain percentile, i.e. a distance d such that $x\%$ of all distances are smaller than d. Although this measure of accuracy allows to rank reconstructions, it is not symmetric since it only takes into account the distances from the reconstruction to the ground truth but not the distances from the ground truth to the reconstruction. This would favour reconstructions that capture only a small part of the ground truth very accurately. Instead of adapting the accuracy measure to account for this, Seitz et al. [95] additionally measures the *completeness* of the reconstruction. To

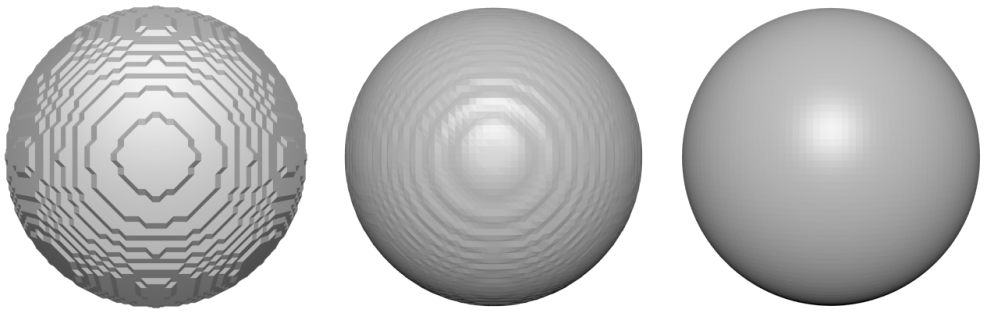


Figure 4.7: **From left to right: (a)-(c)** A sphere was reconstructed from synthetically generated range images with $\delta = 0$, $\delta = |\boldsymbol{h}|/4$, and $\delta = |\boldsymbol{h}|$ using Euclidean signed distances.

this end, they consider the fraction of vertices on the ground truth G that are within an allowable distance to the reconstruction. Points on G that are further away are considered as not covered. Thus, this distance has to be chosen to account for reasonable reconstruction errors.

Guidelines for Choosing the Parameters

The parameters δ and η denote the relevant region close to the surface and the occluded region behind the surface, respectively. When choosing $\delta < |\boldsymbol{h}|$, subvoxel accuracy that was originally present in the range image is lost in the signed distance field due to the truncation of distances. Therefore, one can see increasing staircasing artefacts when δ goes from $|\boldsymbol{h}|$ towards zero (see Figure 4.7(a)-(c)) and it is advisable to choose $\delta \geq |\boldsymbol{h}|$. Additionally, it makes sense to adapt δ in such a way that it reflects the expected measurement error in the depth maps. Choosing η involves a tradeoff: On the one hand, η should be as small as possible to avoid influencing surfaces on the other side. On the other hand, η has to be large enough to allow for sign changes in the signed distance field.

Directional Distance vs. Euclidean Distance

In Figure 4.8(b) one can see artefacts on a sphere that was reconstructed from 48 ground truth range images using directional signed distance values. Figure 4.8(a) shows a visualisation of the error values, where blue corresponds to a negative, red to a positive error value and green corresponds to an error of zero. The error for a vertex is given by its distance to the ground truth according to the error measure for accuracy which we explained in the

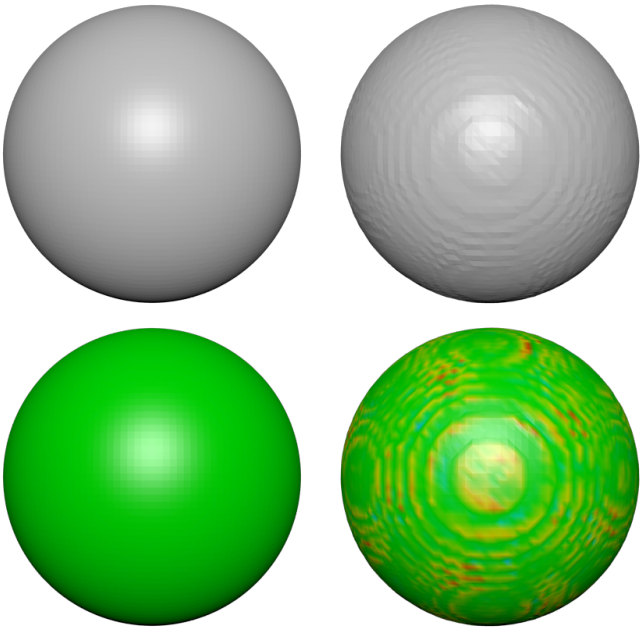


Figure 4.8: A sphere was reconstructed from synthetically generated range images with $\delta = |\boldsymbol{h}|$. **From top left to bottom right: (a)** Reconstruction using Euclidean distance **(b)** Reconstruction using directional signed distances **(c)** Error visualisation for (a) **(d)** Error visualisation for (b)

beginning of this section and which is also used in the Middlebury Benchmark [95]. The minimum and maximum error values have been mapped to blue and red, respectively. Only when using the Euclidean distance to the range surface, one can obtain an accurate reconstruction without artefacts (see Figure 4.8(c),(d)). It is important to note that even in the case of perfect range images without any noise, the directional signed distance can introduce reconstruction errors. For a given voxel, one does not obtain the true Euclidean distance for each camera but a distribution of distance values that depends on the scene geometry as well as the camera position.

Benefits of Anisotropic Regularisation

We have reconstructed two different 3D objects taken from [15] using 48 synthetically generated range images. In order to account for measurement errors as they often occur in modern depth cameras, Gaussian noise has been added along the line of sight. Figure 4.9 and Figure 4.10 show reconstructions using no regularisation, isotropic regularisation, and anisotropic

regularisation. The first two strategies correspond to the methods of Curless and Levoy [29] and Zach et al. [124], respectively. The latter one has been proposed by us.

When using no regularisation, the reconstructed surface is very rough, there is a noticeable amount of isolated clutter, and there is a large amount of locations with high errors (see red and blue regions in Figure 4.9). While the isotropic regularisation allows to get a much smoother reconstruction, wrinkles are still well visible on the surface and there is still a relatively big region containing large errors. In contrast to this, the anisotropic regulariser is able to produce a very smooth surface without any visible wrinkles. Furthermore, it also allows to greatly reduce the area of regions that contain a large error. Figure 4.10 shows that the TV smoothness term is also able to preserve the ridges but cannot achieve a comparable smoothness in the flat regions.

This also holds for the reconstructions of the full Dino dataset (cf. Figure 4.11) from the Middlebury benchmark depicted in Figure 4.13 (b) and (c). We have used the Euclidean distance when converting the range images into signed distance fields, and we have computed depth maps according to the method of Valgaerts et al. [108]. Compared to the method of Zach [123], we are able to significantly improve the accuracy from 0.55mm to 0.33mm (see also `http://vision.middlebury.edu/mview/`). In fact, only the currently leading approach of Furukawa and Ponce [40] is able to obtain a slightly higher accuracy of 0.32mm at this time for the full dataset. However, the reconstruction of Furukawa is not able to achieve a similar smoothness, such that our reconstruction is visually closer to the ground truth (see Figure 4.13 (a)). Table 4.1 additionally lists the error values for the Dino Ring and the Dino Sparse Ring datasets and Figure 4.12 shows reconstruction results. Here one can observe that the type of approach that we employ is best suited for scenarios where many input images are available. As mentioned, our approach is well suited for a parallel implementation which we will demonstrate in the next experiment.

Benefits of Parallel Implementation

In this experiment, we compare the running times of a sequential CPU and a parallel GPU implementation of the Fast-Jacobi algorithm [10] for solving the arising linear systems (4.34). The runtime comparison is shown in Table 4.2, where we have used a 3.2 GHz Intel Xeon processor and a single GPU of the NVIDIA GeForce GTX 690, respectively. The number of unknowns directly corresponds to the number of voxels which itself depends on the resolution

	Accuracy	Completeness
Dino Full	0.33	99.7%
Dino Ring	0.33	99.7%
Dino Sparse Ring	0.54	98.6%

Table 4.1: Accuracy and completeness for the Dino dataset from the Middlebury benchmark (see also http://vision.middlebury.edu/mview/).

Table 4.2: Computing times in seconds for the sequential CPU and the parallel GPU implementation of the Fast-Jacobi algorithm.

resolution	**CPU [s]**	**GPU [s]**	**speed up factor**
$\mathbf{64^3}$	30.03	0.31	96.9
$\mathbf{128^3}$	239.70	1.70	141.0
$\mathbf{256^3}$	2006.05	12.93	155.1

that was specified. Here, we have used the well known *Stanford bunny*[1] as our test object. We have measured runtimes for reconstructions with up to $256^3 \approx 16 \cdot 10^6$ unknowns. As we see in Table 4.2, the parallel implementation is up to 155 times faster than its sequential counterpart. In conclusion, this experiment illustrates that the Fast-Jacobi algorithm is very well suited for parallelisation.

4.7 Limitations and Discussion

By construction, we are required to solve a nonlinear system composed of $N = N_1 \cdot N_2 \cdot N_3$ equations. Thus, the required amount of memory grows cubically when increasing the resolution in each dimension. Therefore, it can be difficult to use methods that find an implicit representation of the 3D surface in a globally optimal way in large scale scenarios, for example in the reconstruction of whole cities. However, adaptive data structures such as octrees may be a remedy in this case or one could try to compute several smaller reconstructions and fuse these reconstructions in another step. Multi-view stereo settings where only a few images are available tend to be the most

[1] taken from the Stanford 3D scanning repository [103]

challenging ones. However, as our method scales very well in the number of range images provided a simple remedy is given by capturing more input data.

4.8 Summary

We have extended the variational range image integration method of Zach et al. [124] in several aspects. On the one hand, we have employed an anisotropic regulariser that outperforms the existing isotropic one. It can produce much smoother surfaces while preserving ridges and corners. On the other hand, the signed distance fields were generated from range images by computing the Euclidean signed distance to the range surface instead of evaluating the directional signed distance along the line of sight. In the experimental section, we could show that these modifications were able to improve the reconstruction quality. Furthermore, we have presented a parallel GPU implementation that allows for competitive runtimes.

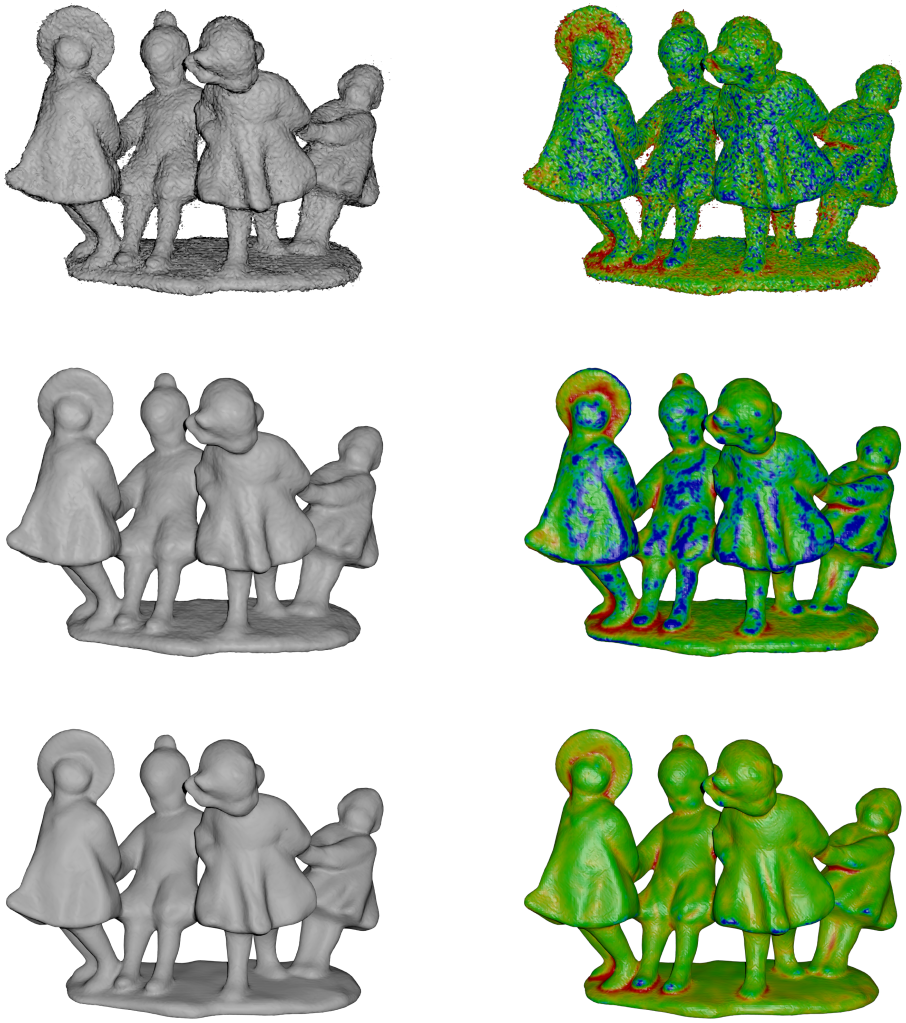


Figure 4.9: Reconstructions of the dancing children model taken from [15]. **From top to bottom**: No regularisation. Isotropic regularisation with $\alpha = 3$. Anisotropic regularisation with $\alpha = 1$, $\sigma = 0.9$, and $\rho = 1.5$. **From left to right**: Reconstructed surface. Errors w.r.t. the ground truth surface using the colour code shown in Figure 4.6.

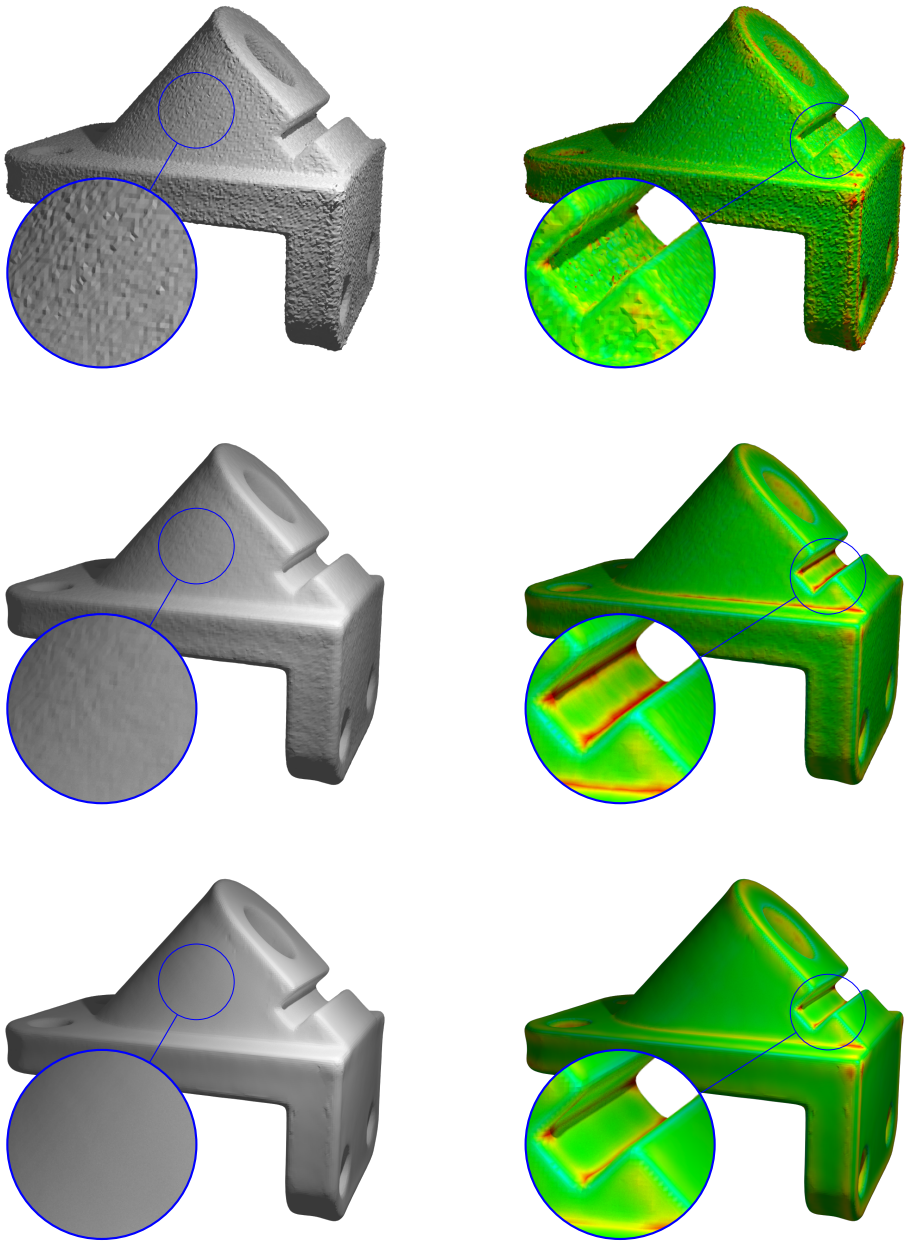


Figure 4.10: Reconstructions of the anchor model taken from [15]. **From top to bottom**: No regularisation. Isotropic regularisation with $\alpha = 3$. Anisotropic regularisation with $\alpha = 3$, $\sigma = 0.2$, and $\rho = 0.6$. **From left to right**: Reconstructed surface. Errors w.r.t. the ground truth surface using the colour code shown in Figure 4.6.

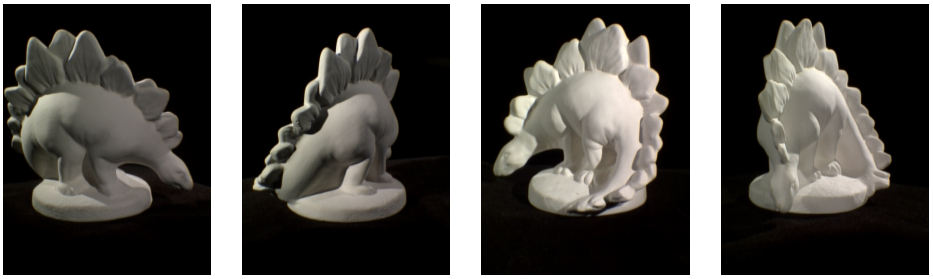


Figure 4.11: Four out of the 363 images that are available in the full dataset of the Middlebury benchmark. Each image has a resolution of 640 × 480 pixels.

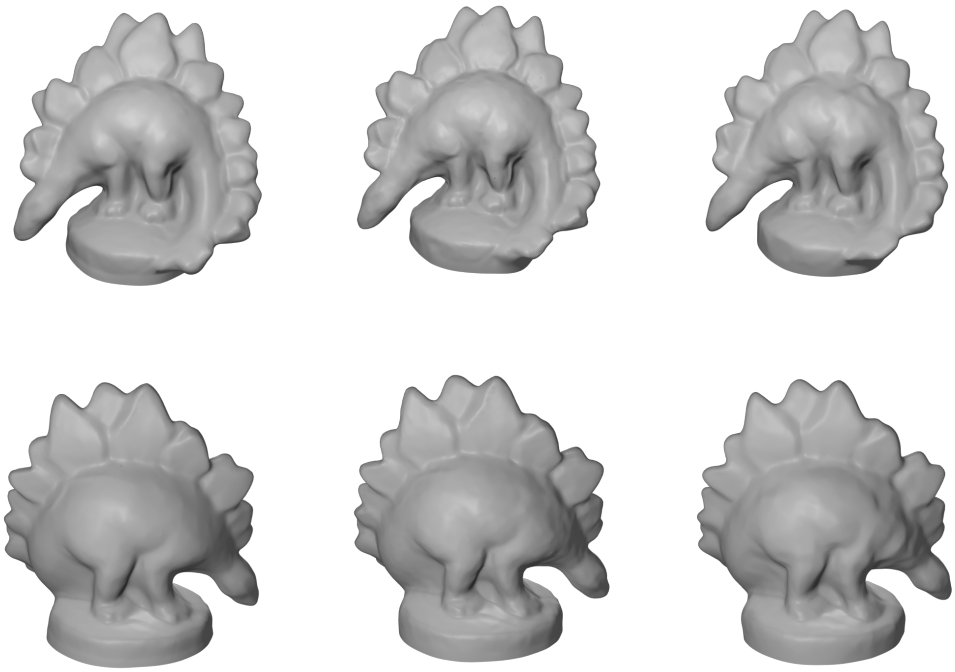
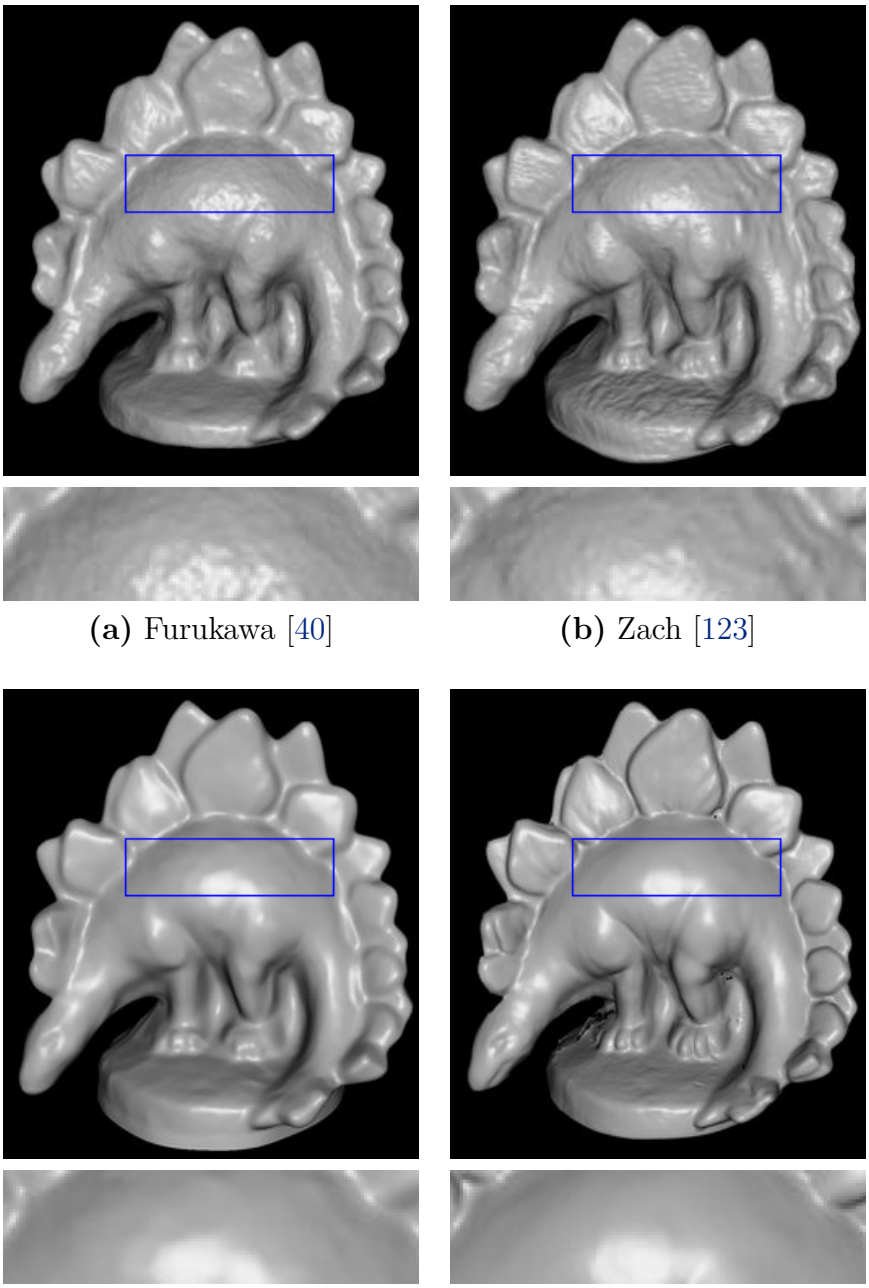


Figure 4.12: Front and back view of the reconstruction results for the Dino model from the Middlebury benchmark. **Columns from left to right**: Full dataset. Ring dataset. Sparsering dataset. The corresponding error values are shown in Table 4.1.



(a) Furukawa [40] **(b)** Zach [123]

(c) Our method **(d)** Ground truth

Figure 4.13: Compared to the reconstruction of the leading method of Furukawa and Ponce [40] as well as the method of Zach [123], our approach creates a smoother surface that better resembles the ground truth in a visual comparison. This is especially visible in the zooms that are shown below the respective reconstruction.

Chapter 5

Surface Reconstruction from Image Sequences

This chapter is concerned with computing 3D reconstructions from images of a moving camera where the camera pose is not known (see Figure 5.1). Many of the existing algorithms for this task rely on *sparse* features. Such methods have to carefully select the most appropriate data and eliminate outliers. On the other hand, *dense* methods do not have to put effort into selecting the best data but rather draw their robustness from using all data. They have made significant progress in the last decade and belong to the leading approaches for optical flow computation; see e.g. [13]. Moreover, dense strategies can also be on par with sparse methods for other problems such as the estimation of the fundamental matrix [109]. Motivated by these achievements, we construct a pipeline for 3D reconstruction that consistently relies on dense methods: It does not require sparse features at any point. The pipeline can be divided into three stages. (i) First we compute dense correspondences and the fundamental matrix for each consecutive image pair in a joint approach. Every pairwise estimate generally has its own scale. To estimate a consistent motion sequence these scales have to be unified. (ii) Thus, in the second step, we first connect a number of pairwise estimates and subsequently perform a global refinement with bundle adjustment. (iii) Finally, we refine the depth maps and merge them into a 3D model using anisotropic range image integration.

In contrast to many existing approaches for camera motion estimation that rely on feature descriptors, our novel pipeline uses state-of-the-art dense variational methods in every part of the process of reconstructing a 3D object from unregistered cameras. This is especially advantageous in sequences with less texture and many similar structures, where unordered feature matching is difficult. The main ideas of this chapter have been published in [5].

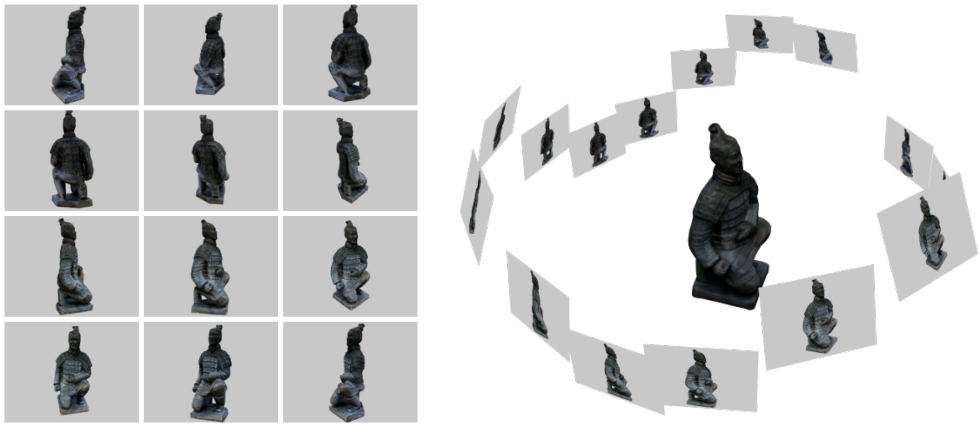


Figure 5.1: Given a number of input colour images captured by a camera following a continuous path along with its intrinsic camera parameters (left), the goal is to estimate the camera motion and a 3D reconstruction of the scene or object (right).

Organisation of the Chapter. After discussing related work and our contributions, Section 5.3 presents our dense reconstruction pipeline and explains each stage in detail. Section 5.4 then evaluates its performance before describe limitations and possible remedies (Section 5.5). Finally, we conclude with a summary in Section 5.6.

5.1 Related Work

Based on the problem statement of this chapter and the chosen solution strategy, there are two classes of methods that are closely related: On the one hand we have *structure from motion* (SfM) or *simultaneous localisation and mapping* (SLAM) approaches and on the other hand range image integration techniques. The latter ones have already been described in the previous chapter 4.1. Therefore, we will only focus on the former ones here.

There are many algorithms that simultaneously estimate camera poses along with the structure of a scene. Approaches that are tailored to work with unstructured input data often apply a robust detection and matching of feature points. These matches are first used to find initial geometric relationships between views and subsequently form the basis for a global bundle adjustment step; see e.g. [12, 11]. In this scenario, the term structure from motion is more prevalent. On the other hand, methods that are designed

for continuous image sequences or video streams often track feature points from frame to frame and perform pose optimisation iteratively. These types of approaches are mostly referred to as SLAM and they are more closely related to our scenario in the sense that we also assume a camera that is moving continuously while capturing a static scene. SLAM originally comes from the area of robotics and refers to the process of a mobile robot building a map of some environment while at the same time using this map to determine its own location [31]. However, it is also widely used for tracking a position of a camera in general.

A very prominent algorithm from this category is the parallel tracking and mapping (PTAM) algorithm by Klein and Murray [61]. It tracks sparse features of the scene and creates a map of the environment. With this map the new camera pose is estimated depending on the matched features. Newcombe et al. [80] follow a similar tracking approach, but use a dense scene model instead of a sparse feature map in their dense tracking and mapping (DTAM) method. This way, they achieve a higher robustness to rapid motion. However, they still need standard point features to initialise their tracking algorithm. In order to describe such point features one can employ the well-known SIFT descriptor by Lowe [73], or one of the many modifications such as SURF [14] or GLOH [78]. Direct SLAM methods on the other hand do not require detection and tracking of features. Engel et al. [32] recover semi-dense inverse depth maps based on photoconsistency in their large scale direct (LSD-SLAM) approach for example.

Instead of using colour images for tracking, also depth maps can be successfully employed for this task. Newcombe et al. [79] present a tracking algorithm based on dense depth frame alignment, and Izadi et al. [57] investigate a similar approach that focuses on reconstruction. Zhou et al. use points of interest to reconstruct a dense model from range images [125]. They employ a global optimisation scheme which protects parts of the scene that have been scanned already. This leads to more consistent and detailed reconstructions. The disadvantage of depth sensors is their typically lower resolution compared to RGB images. Moreover, these algorithms are only applicable if depth data is available which cannot generally be assumed.

5.2 Contributions

We present a novel 3D reconstruction pipeline that allows to obtain both the camera motion as well as a watertight 3D model of the object or scene that was observed. It solely requires a sequence of colour images captured from a camera that follows a continuous path as well as the intrinsic parameters

of the camera. Our pipeline does not rely on sparse features at any point and consistently makes use of dense methods. It consists of three main steps of which each single one minimises a suitable energy functional such that the modelling choices are very transparent throughout the whole pipeline. We experimentally present advantages compared to methods that rely on sparse features and are able to outperform them quantitatively in terms of the achieved error. The implementation for the concatenation of flow fields as well as the error computation of the camera poses has been provided by Timm Schneevoigt [93].

5.3 Dense Reconstruction Pipeline

This section describes the three steps of our dense 3D reconstruction pipeline and explains how they can be connected. First, we discuss how to obtain correspondences and epipolar geometry for all consecutive image pairs following a joint approach. Then we describe how we connect this pairwise information to compute a globally consistent camera motion with the help of bundle adjustment. Finally, we explain how this enables us to construct a high quality 3D model using anisotropic range image integration.

5.3.1 Correspondences and Epipolar Geometry

As we assume a static scene, the only scene element that moves is the camera itself. Therefore, the moving camera can equivalently be understood as multiple identical cameras that capture the scene from different positions at the same time. In this chapter, we assume that there is no knowledge about the camera movement other than that contained in the images. In order to cope with this, we use an anisotropic version of the joint method of Valgaerts et al. [109]. It is capable of estimating dense point correspondences and the associated fundamental matrix at the same time. Besides obtaining the fundamental matrix, this has the inherent advantage that the correspondence search is simplified as it is guided by the evolving epipolar constraint. The flow field $\boldsymbol{w} : \Omega \to \mathbb{R}^2$ over the image domain Ω between two images f_i, f_{i+1} and the fundamental matrix $\boldsymbol{F} \in \mathbb{R}^{3\times3}$ are found as a minimiser of a suitable energy in this joint approach [109]:

$$E(\boldsymbol{w}, \boldsymbol{F}) = E_D + \alpha E_S + \beta E_E \,. \tag{5.1}$$

Here E_D is the data term, E_S the smoothness term, and E_E denotes the epipolar term. The weights α, $\beta > 0$ balance the individual terms. In the

data term we assume brightness and gradient constancy for all image points, expressed by the first and second line of the following data term, respectively:

$$E_D(\boldsymbol{w}) = \int_\Omega \Psi\big(\, |f_{i+1}(\boldsymbol{x}+\boldsymbol{w}) - f_i(\boldsymbol{x})|^2 \\ + \gamma |\boldsymbol{\nabla} f_{i+1}(\boldsymbol{x}+\boldsymbol{w}) - \boldsymbol{\nabla} f_i(\boldsymbol{x})|^2 \,\big) \mathrm{d}\boldsymbol{x}\,, \tag{5.2}$$

with a sub-quadratic penaliser function $\Psi(s^2) = \sqrt{s^2+\epsilon^2}$ and a positive weight parameter $\gamma \in \mathbb{R}$ that balances brightness and gradient constancy assumptions.

To obtain a smooth flow field $\boldsymbol{w} = (u, v)^T$, Valgaerts et al. [109] employed an isotropic flow-driven regulariser with sub-quadratic penalisation. To obtain better performance, we use a flow-driven anisotropic regulariser [116]

$$E_S(\boldsymbol{\nabla}\boldsymbol{w}) = \int_\Omega \mathrm{tr}\,\big(\, \Psi\left(\boldsymbol{\nabla} u \boldsymbol{\nabla} u^T + \boldsymbol{\nabla} v \boldsymbol{\nabla} v^T\right) \big)\, \mathrm{d}\boldsymbol{x}\,, \tag{5.3}$$

where tr is the matrix trace operator, and Ψ is an extension of the scalar valued function that acts on the eigenvalues of the matrix. The epipolar term directly couples the flow field $\boldsymbol{w}$ with the fundamental matrix $\boldsymbol{F}$:

$$E_E(\boldsymbol{w}, \boldsymbol{F}) = \int_\Omega \Psi\left(\begin{pmatrix}\boldsymbol{x}+\boldsymbol{w}\\ 1\end{pmatrix}^T \boldsymbol{F} \begin{pmatrix}\boldsymbol{x}\\ 1\end{pmatrix}\right) \mathrm{d}\boldsymbol{x}\,. \tag{5.4}$$

In this way it softly constrains the correspondences to fulfil an epipolar constraint. Note that the trivial solution of $\boldsymbol{F} = \boldsymbol{0}$ has to be excluded by imposing a constraint on the Frobenius norm $|\boldsymbol{F}|_F^2 = 1$.

As in [109], we employ the method of Lagrange multipliers to solve the constrained optimisation problem (5.1) subject to $|\boldsymbol{F}|_F^2 = 1$. Thus, we have to find critical points for which the functional derivatives of the Lagrangian vanish. This comes down to solving a system of equations, which in this case can be achieved by iterating between optical flow computation and fundamental matrix estimation. In the first iteration, we can simply use a zero matrix as initialisation for the fundamental matrix. This corresponds to switching off the epipolar constraint such that the fundamental matrix is recovered from pure optical flow in the first iteration.

5.3.2 Constraints over Multiple Images

After processing all subsequent image pairs of the sequence as discussed in the previous subsection, we obtain fundamental matrices that describe the

epipolar geometry for each image pair along with dense point correspondences (flow field). Since we assume that the cameras are calibrated, i. e. the intrinsic parameters are known, we can extract the relative pose of a canonical camera pair from each estimated fundamental matrix as described in Section 2.2.3. The main problem at this stage is merely the remaining scale ambiguity for each pairwise estimate.

To find a mutually consistent scale we have to enforce a set of constraints that connects the whole image sequence. Each constraint has to span at least three images to contribute to a consistent scale. To build such constraints, we construct point trajectories by simply adding up flow vectors. Let us assume we would like to start a trajectory at location $\boldsymbol{x}$ in the i-th image. Then the location in the next image is given by

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{w}_i(\boldsymbol{x}_i). \tag{5.5}$$

We keep extending the trajectory in this manner and estimate displacement vectors between the flow field's grid positions by bilinear interpolation. We follow the idea of Sundaram et al. [106] and test the validity of the resulting trajectories with a forward-backward flow consistency check. If there are no occlusions and the flow was correctly estimated, we expect that forward flow $\boldsymbol{w}_i$ and backward flow $\hat{\boldsymbol{w}}_i$ add up to zero

$$\boldsymbol{w}_i(\boldsymbol{x}_i) - \hat{\boldsymbol{w}}_i(\boldsymbol{x}_{i+1}) = \mathbf{0}. \tag{5.6}$$

Thus, if the sum of forward and backward flow is too far away from zero, this is a good reason to stop extending the trajectory. To account for small estimation errors in the optic flow, we check whether

$$|\boldsymbol{w}_i - \hat{\boldsymbol{w}}_i|^2 < \sigma \cdot (|\boldsymbol{w}_i|^2 + |\hat{\boldsymbol{w}}_i|^2) + \mu. \tag{5.7}$$

Besides having a fixed threshold μ, the factor σ allows to increase the tolerance with motion magnitude. In practice, $\mu = 0.5$ and $\sigma = 0.01$ work well. Furthermore, regions of large flow gradients can originate from flow boundaries, where the estimates are less reliable. We also exclude such regions from our computation by the criterion

$$|\boldsymbol{\nabla} u_i|^2 + |\boldsymbol{\nabla} v_i|^2 > \sigma \cdot |\boldsymbol{w}_i|^2 + \mu \tag{5.8}$$

with $\mu = 0.002$ and $\sigma = 0.01$ as proposed by Sundaram et al. Even with highly accurate optical flow and despite the mentioned consistency checks, estimation and interpolation errors add up at some point. Therefore, we

propose to limit the length of trajectories to a fixed value. However, short trajectories that connect only two frames may still help to improve or preserve the initially estimated relative scene geometry, even if it does not contribute to obtaining a unified scale directly.

Unlike Sundaram et al., we do not limit the sampling of trajectory starting points to well textured regions. Since the idea of dense optical flow is to provide meaningful displacements even in less textured regions, we spare this additional filtering effort. In our experiments, we show that a simple uniform subsampling can be used to decrease the computational effort and that this can even lead to more accurate results when compared to the texture based sampling. The trajectory points should just lie on the scene object. Background regions are filtered out by given masks, which typically can be obtained by image segmentation approaches.

5.3.3 Bundle Adjustment

In order to find a consistent camera motion, we would like to adapt all camera poses to minimise the reprojection error induced by the point trajectories. This task is known as *bundle adjustment* [107]. Let us have a closer look at the formulation of the corresponding cost function, its minimisation, and its parameterisation.

Cost Function

As stated, the cost function for bundle adjustment should penalise deviations between observed locations and predicted locations in the image. To this end, let us assume that there are K cameras $\boldsymbol{P}_i$ with $i = 1, \dots, K$ and that we have created N trajectories by adding up flow vectors in the previous step. Each trajectory has to describe the observation of the same 3D point over several images. Therefore, there are also N 3D points $\boldsymbol{X}_i$ with $i = 1, \dots, N$ involved. Each trajectory can have a length of K at most. This means that every 3D point $\boldsymbol{X}_i$ was observed in all images, leading to $L = N \cdot K$ 2D observations $\boldsymbol{m}_i$ with $i = 1, \dots, L$ in the image plane. However, due to visibility constraints and due to the fact that we usually limit the maximum length of trajectories, the number of observations L will be much smaller in practice such that $L \ll N \cdot K$ holds. In order to relate measurements $\boldsymbol{m}_i$, 3D points $\boldsymbol{X}_i$ and cameras $\boldsymbol{P}_i$, we define the two maps

$$\eta : [1, \dots, L] \to [1, \dots, N] \quad \text{and} \quad \kappa : [1, \dots, L] \to [1, \dots, K]. \tag{5.9}$$

They specify to which camera and to which 3D point the i-th measurement belongs, respectively. Thus, the observation $\boldsymbol{m}_i$ should correspond to

projecting the 3D point $\boldsymbol{X}_{\kappa(i)}$ into camera $\boldsymbol{P}_{\eta(i)}$. This allows to define the reprojection error as

$$\boldsymbol{r}_i(\boldsymbol{x}) = \boldsymbol{m}_i - \boldsymbol{\pi}\left(\boldsymbol{P}_{\eta(i)}\tilde{\boldsymbol{X}}_{\kappa(i)}\right), \tag{5.10}$$

where $\boldsymbol{\pi}$ maps from homogeneous to Euclidean coordinates, and $\boldsymbol{x}$ contains all unknown camera parameters $\boldsymbol{\xi}_i \in \mathbb{R}^6$ and 3D points $\boldsymbol{X}_i \in \mathbb{R}^3$, i.e.

$$\boldsymbol{x} = \left(\boldsymbol{\xi}_1^\top, \ldots, \boldsymbol{\xi}_K^\top, \boldsymbol{X}_1^\top, \ldots, \boldsymbol{X}_N^\top\right)^\top . \tag{5.11}$$

Besides the number of cameras K and the number of 3D points N, the number of unknowns that is contained in $\boldsymbol{x}$ also depends on the parameterisation of the cameras $\boldsymbol{P}_i$. We will have $A = 3N + 6K$ unknowns because we have parameterised the camera $\boldsymbol{P}_i$ to have six degrees of freedom $\boldsymbol{\xi}_i = (\boldsymbol{\omega}_i^\top, \boldsymbol{t}_i^\top)^\top$ where $\boldsymbol{\omega}_i$ represents the rotation and $\boldsymbol{t}_i$ the translation. Note that we consider the intrinsic camera parameters as given in this scenario. The cost function then reads

$$C(\boldsymbol{x}) = \sum_{i=1}^{L} |\boldsymbol{r}_i(\boldsymbol{x})|^2 = |\boldsymbol{f}(\boldsymbol{x})|^2 \tag{5.12}$$

with $\boldsymbol{f}(\boldsymbol{x}) = (\boldsymbol{r}_1^\top(\boldsymbol{x}), \ldots, \boldsymbol{r}_L^\top(\boldsymbol{x}))^\top$. Bundle adjustment, i.e. the procedure of minimising $C(\boldsymbol{x})$ for all unknown 3D points and camera parameters $\boldsymbol{x}$, thus comes down to solving the non-linear least squares problem

$$\underset{\boldsymbol{x}\in\mathbb{R}^A}{\operatorname{argmin}} \, |\boldsymbol{f}(\boldsymbol{x})|^2 . \tag{5.13}$$

Thus, we briefly cover standard approaches for solving non-linear least squares problems and then specifically focus on the bundle adjustment problem.

Minimisation

The *Gauss-Newton* method is a very common approach to tackle nonlinear least squares problems. It turns the task of solving a nonlinear problem into the much simpler task of solving a series of linear problems. To this end, Taylor's theorem is used to rewrite the original problem (5.13) by linearising around a point $\boldsymbol{x}^k$. This gives

$$\underset{\Delta\boldsymbol{x} \, \in \, \mathbb{R}^A}{\operatorname{argmin}} \, \left|\boldsymbol{f}\left(\boldsymbol{x}^k\right) + \boldsymbol{J}\left(\boldsymbol{x}^k\right)\Delta\boldsymbol{x}\right|^2 \tag{5.14}$$

with $\Delta\boldsymbol{x} = \boldsymbol{x}^{k+1} - \boldsymbol{x}^k$ and $\boldsymbol{J}$ denoting the Jacobian of $\boldsymbol{f}$. The solution of the linear least squares problem with respect to the increment $\Delta\boldsymbol{x}$ is given by

$$\Delta\boldsymbol{x} = -\left(\boldsymbol{J}\left(\boldsymbol{x}^k\right)^\top \boldsymbol{J}\left(\boldsymbol{x}^k\right)\right)^{-1} \boldsymbol{J}\left(\boldsymbol{x}^k\right)^\top \boldsymbol{f}\left(\boldsymbol{x}^k\right). \tag{5.15}$$

With this, one computes the new solution $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \Delta\boldsymbol{x}$ and repeats this process until a stopping criterion is reached. Typical stopping criteria involve checking whether $|\Delta\boldsymbol{x}| < \epsilon$ or $|\boldsymbol{J}\left(\boldsymbol{x}^k\right)^\top \boldsymbol{f}\left(\boldsymbol{x}^k\right)| < \epsilon$, where ϵ is some small constant and $\boldsymbol{J}\left(\boldsymbol{x}^k\right)^\top \boldsymbol{f}\left(\boldsymbol{x}^k\right) = {}^1\!/_2 \boldsymbol{\nabla} C\left(\boldsymbol{x}^k\right)$ corresponds to a scaled version of the gradient of the energy (5.12) that one wants to minimise. The Gauss-Newton method can also be interpreted as performing the Newton method for $\boldsymbol{\nabla} C(\boldsymbol{x})$

$$\Delta\boldsymbol{x} = -\boldsymbol{H}^{-1} \boldsymbol{\nabla} C(\boldsymbol{x}), \tag{5.16}$$

where the Hessian $\boldsymbol{H}$ is approximated using the product $2\boldsymbol{J}^\top\boldsymbol{J}$ of the Jacobian. Levenberg [70] suggested to replace (5.15) by a damped version

$$\Delta\boldsymbol{x} = -\left(\boldsymbol{J}\left(\boldsymbol{x}^k\right)^\top \boldsymbol{J}\left(\boldsymbol{x}^k\right) + \lambda\boldsymbol{I}\right)^{-1} \boldsymbol{J}\left(\boldsymbol{x}^k\right)^\top \boldsymbol{f}\left(\boldsymbol{x}^k\right), \tag{5.17}$$

which corresponds to adding the term $\lambda|\Delta\boldsymbol{x}|^2$ to the Energy 5.14 in order to force small increments. For $\lambda = 0$, it obviously corresponds to the standard Gauss-Newton method and for very large λ it turns into gradient descent on (5.12). The non-negative damping factor λ is usually adjusted at each iteration and allows to enforce a decrease in $C(\boldsymbol{x})$ in each step. This way, it is possible to interpolate between Gauss-Newton and gradient descent which renders this approach more robust in terms of finding a solution even with a bad initialisation. Marquardt [77] replaced the identity matrix by the diagonal entries of $\boldsymbol{J}^\top\boldsymbol{J}$ in order to improve convergence yielding the iteration instruction

$$\Delta\boldsymbol{x} = -\left(\boldsymbol{J}^\top\boldsymbol{J} + \lambda \mathrm{diag}\left(\boldsymbol{J}^\top\boldsymbol{J}\right)\right)^{-1} \boldsymbol{J}^\top\boldsymbol{f}, \tag{5.18}$$

where we have omitted the dependency on $\boldsymbol{x}^k$ for better readability.

Problem Structure

Let us first inspect the structure of the Jacobian $\boldsymbol{J}$ and $\boldsymbol{J}^\top\boldsymbol{J}$ for the bundle adjustment problem before dealing with the parameterisation and the computation of individual partial derivatives. Figure 5.2 shows the sparsity pattern for a simple example with $L = 7$ observations, $K = 3$ cameras, $N = 3$

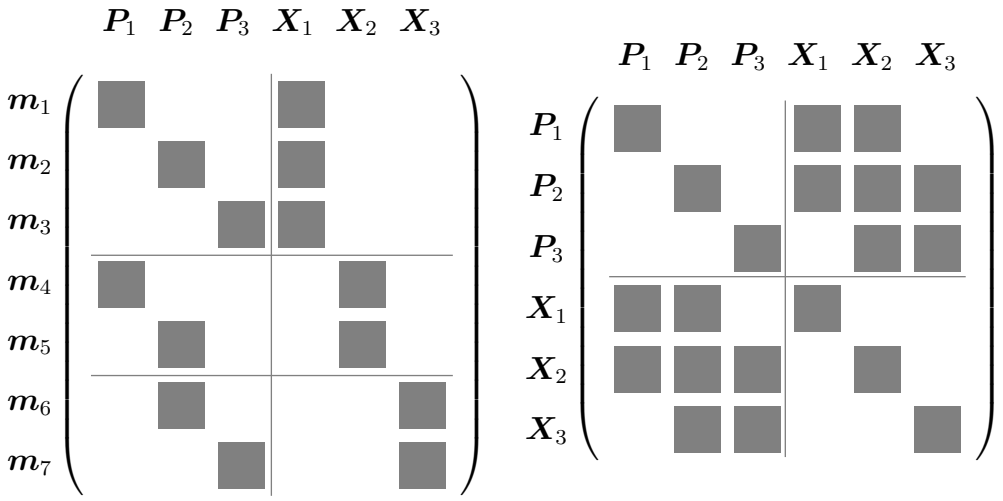


Figure 5.2: Structure of the Jacobian $\boldsymbol{J}$ (left) and $\boldsymbol{J}^\top\boldsymbol{J}$ (right) for the bundle adjustment problem in a small toy example. Grey squares denote nonzero entries in the matrices.

3D points, and the mappings

$$\eta = \{(1,1),(2,2),(3,3),(4,1),(5,2),(6,2),(7,3)\}\,, \tag{5.19}$$

$$\kappa = \{(1,1),(2,1),(3,1),(4,2),(5,2),(6,3),(7,3)\}\,, \tag{5.20}$$

that specify to which camera and to which 3D point a measurement belongs, respectively. Typically, the Jacobian has a very sparse structure, since each $\boldsymbol{r}_i$ only depends on a single camera and a single 3D point. Thus, each row of the Jacobian is zero everywhere except at the columns that belong to the corresponding unknown camera parameters $\boldsymbol{P}_{\eta(i)}$ or the unknown 3D point $\boldsymbol{X}_{\kappa(i)}$. Due to the structure of $\boldsymbol{J}$, we can express

$$\boldsymbol{J}^\top\boldsymbol{J} = \begin{pmatrix} \boldsymbol{U} & \boldsymbol{W} \\ \boldsymbol{W}^\top & \boldsymbol{V} \end{pmatrix}, \tag{5.21}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are block diagonal matrices. The individual blocks on the diagonal of $\boldsymbol{U}$ are in our case given by the 6×6 matrices

$$\boldsymbol{U}_{kk} = \sum_{\substack{i=1 \\ \eta(i)=k}}^{L} \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{P}_k}^\top \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{P}_k} \tag{5.22}$$

that aggregate the partial derivatives w.r.t. the camera parameters of all observations that are in the k-th camera for $k = 1, \ldots, K$. We will discuss the evaluation of the occurring partial derivatives in the next section where we get to the parameterisation of camera motion and focus on the problem structure in this section. The blocks on the diagonal of $\boldsymbol{V}$ are given by the 3×3 matrices

$$\boldsymbol{V}_{jj} = \sum_{\substack{i=1 \\ \kappa(i)=j}}^{L} \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{X}_j}^\top \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{X}_j} \tag{5.23}$$

that aggregate the partial derivatives w.r.t. the 3D point positions of all observations of the j-th 3D point for $j = 1, \ldots, N$. If there exists a measurement i that connects the j-th 3D point with the k-th camera, i.e. $\eta(i) = k$ and $\kappa(i) = j$, then

$$\boldsymbol{W}_{kj} = \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{P}_k}^\top \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{X}_j}. \tag{5.24}$$

Otherwise we have $\boldsymbol{W}_{kj} = \boldsymbol{0}$. Thus, the sparsity of $\boldsymbol{W}$ depends on how cameras and 3D points are connected through observations. If all 3D points were visible in all cameras, $\boldsymbol{W}$ would be dense. However, especially when the problems grow larger involving many cameras, $\boldsymbol{W}$ tends to be very sparse.

Parameterisation and Partial Derivatives

Previously, we have discussed the structure of $\boldsymbol{J}$ and $\boldsymbol{J}^\top \boldsymbol{J}$. However, we have not yet treated how the individual partial derivatives of

$$\boldsymbol{r}_i(\boldsymbol{x}) = \boldsymbol{m}_i - \boldsymbol{\pi}\left(\boldsymbol{P}_{\eta(i)} \tilde{\boldsymbol{X}}_{\kappa(i)}\right) \tag{5.25}$$

can be computed. To this end, we can w.l.o.g. drop the index i for notational convenience and simply consider a single residual that only depends on the parameters of the corresponding camera $\boldsymbol{P}_{\eta(i)}$ and the corresponding 3D point $\tilde{\boldsymbol{X}}_{\kappa(i)}$. This can be written as

$$\boldsymbol{r}(\boldsymbol{X}, \boldsymbol{\omega}, \boldsymbol{t}) = \boldsymbol{m} - \boldsymbol{\pi}\left(\boldsymbol{K}\begin{bmatrix}\exp(\hat{\boldsymbol{\omega}})\boldsymbol{R} & \boldsymbol{t}\end{bmatrix}\tilde{\boldsymbol{X}}\right), \tag{5.26}$$

where $\hat{\boldsymbol{\omega}} \in \mathfrak{so}(3)$ is an element of the Lie algebra, and $\exp : \mathfrak{so}(3) \to \mathbb{SO}(3)$ maps from the Lie algebra to the Lie group; see Section 2.1.2. Furthermore, we have used the forward compositional formulation for representing the camera rotation as in [80, 32]. This means the rotation is described as the

composition of a known rotation $\boldsymbol{R}$ and an unknown incremental rotation $\exp(\hat{\boldsymbol{\omega}})$. In our case, we initialise the known rotation based on the camera rotation extracted from the essential matrices as described in Section 2.2.3. As soon as an increment is computed, the known rotation is updated with the rule

$$\boldsymbol{R} \leftarrow \exp(\hat{\boldsymbol{\omega}})\boldsymbol{R} \tag{5.27}$$

and the next incremental rotation is computed until convergence is reached. With this strategy one always linearises around $\boldsymbol{\omega} = \mathbf{0}$ in order to obtain the incremental rotation and thus only has to evaluate the partial derivatives at $\boldsymbol{\omega} = \mathbf{0}$. More specifically, we have

$$\left.\frac{\partial \boldsymbol{r}(\boldsymbol{X}, \boldsymbol{\omega}, \boldsymbol{t})}{\partial \omega_m}\right|_{\boldsymbol{\omega}=0} = -\left.\frac{\partial \boldsymbol{\pi}(\boldsymbol{a})}{\partial \boldsymbol{a}}\right|_{\boldsymbol{a}=\boldsymbol{K}\begin{pmatrix}\boldsymbol{R} & \boldsymbol{t}\end{pmatrix}\tilde{\boldsymbol{X}}} \boldsymbol{K}\left(\left.\frac{\partial \exp(\hat{\boldsymbol{\omega}})}{\partial \omega_m}\right|_{\boldsymbol{\omega}=0} \boldsymbol{R} \quad \mathbf{0}\right)\tilde{\boldsymbol{X}} \tag{5.28}$$

for the partial derivatives w.r.t. the camera rotation with $m = 1, 2, 3$. As explained in 2.1.2, the partial derivatives

$$\left.\frac{\partial \exp(\hat{\boldsymbol{\omega}})}{\partial \omega_m}\right|_{\boldsymbol{\omega}=0} \tag{5.29}$$

are simply given by the group generators. Using this fact allows to rewrite the above expression and to conveniently express the partial derivative w.r.t. all $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)^\top$ as

$$\left.\frac{\partial \boldsymbol{r}(\boldsymbol{X}, \boldsymbol{\omega}, \boldsymbol{t})}{\partial \boldsymbol{\omega}}\right|_{\boldsymbol{\omega}=\mathbf{0}} = -\left.\frac{\partial \boldsymbol{\pi}(\boldsymbol{a})}{\partial \boldsymbol{a}}\right|_{\boldsymbol{a}=\boldsymbol{K}\begin{pmatrix}\boldsymbol{R} & \boldsymbol{t}\end{pmatrix}\tilde{\boldsymbol{X}}} \boldsymbol{K}(-[\boldsymbol{R}\boldsymbol{X}]_\times). \tag{5.30}$$

The derivative w.r.t. the translation $\boldsymbol{t}$ is given by

$$\left.\frac{\partial \boldsymbol{r}(\boldsymbol{X}, \boldsymbol{\omega}, \boldsymbol{t})}{\partial \boldsymbol{t}}\right|_{\boldsymbol{\omega}=\mathbf{0}} = -\left.\frac{\partial \boldsymbol{\pi}(\boldsymbol{a})}{\partial \boldsymbol{a}}\right|_{\boldsymbol{a}=\boldsymbol{K}\begin{pmatrix}\boldsymbol{R} & \boldsymbol{t}\end{pmatrix}\tilde{\boldsymbol{X}}} \boldsymbol{K}, \tag{5.31}$$

and the derivative w.r.t. the 3D point $\boldsymbol{X}$ is given by

$$\left.\frac{\partial \boldsymbol{r}(\boldsymbol{X}, \boldsymbol{\omega}, \boldsymbol{t})}{\partial \boldsymbol{X}}\right|_{\boldsymbol{\omega}=\mathbf{0}} = -\left.\frac{\partial \boldsymbol{\pi}(\boldsymbol{a})}{\partial \boldsymbol{a}}\right|_{\boldsymbol{a}=\boldsymbol{K}\begin{pmatrix}\boldsymbol{R} & \boldsymbol{t}\end{pmatrix}\tilde{\boldsymbol{X}}} \boldsymbol{K}\boldsymbol{R}\,. \tag{5.32}$$

Robustification and Solvers. To deal with outliers, it is a straightforward approach to robustify the Energy 5.12 by means of a subquadratic penaliser function Ψ:

$$C(\boldsymbol{x}) = \sum_{i=1}^{L} \Psi\left(|\boldsymbol{r}_i(\boldsymbol{x})|^2\right). \tag{5.33}$$

Following the same strategy as before, one obtains the modified update equation

$$\Delta \boldsymbol{x} = -\left(\boldsymbol{J}\left(\boldsymbol{x}^k\right)^\top \boldsymbol{W}\left(\boldsymbol{x}^k\right) \boldsymbol{J}\left(\boldsymbol{x}^k\right)\right)^{-1} \boldsymbol{J}\left(\boldsymbol{x}^k\right)^\top \boldsymbol{W}\left(\boldsymbol{x}^k\right) \boldsymbol{f}\left(\boldsymbol{x}^k\right), \tag{5.34}$$

when using the lagged diffusivity method [41, 25]. Compared to (5.15), it contains an additional diagonal matrix $\boldsymbol{W}(\boldsymbol{x}^k)$ that contains the weights $\Psi'\left(|\boldsymbol{r}_i(\boldsymbol{x}^k)|^2\right)$. Alternatively, this can also be interpreted as an iteratively reweighted least squares approach where in each iteration the weights are computed based on the previous solution and subsequently a weighted least squares fit is performed.

The (sparse) Cholesky factorisation is a common approach to solving the resulting linear systems [107, 49, 33, 72]. However, also iterative solvers have been successfully applied in the bundle adjustment problem [11, 20, 12, 121]. It is possible to obtain a reduced camera or structure system that only contains unknowns either concerning the camera parameters or the 3D points by using Gauss elimination. The remaining unknowns can be found by back substitution in a second step. This allows to solve two smaller linear problems instead of a single large one and is known as the *Schur complement trick* [19].

Implementation Details

After processing all subsequent image pairs of the sequence, we concatenate the pairwise poses $(\hat{\boldsymbol{R}}_k, \hat{\boldsymbol{t}}_k)$ that specify the rotation and translation relative to the previous camera one by one to form an initial scene model. The absolute camera poses in this initial scene model are thus simply given by

$$\begin{pmatrix} \boldsymbol{R}_k & \boldsymbol{t}_k \\ \boldsymbol{0} & 1 \end{pmatrix} = \prod_{l=1}^{k} \begin{pmatrix} \hat{\boldsymbol{R}}_l & \hat{\boldsymbol{t}}_l \\ \boldsymbol{0} & 1 \end{pmatrix}. \tag{5.35}$$

Furthermore, we triangulate the depth of one 3D point for each point trajectory to serve as an initialisation. This initial scene model is then refined by enforcing the constraints imposed by the point trajectories. To this end, we minimise the reprojection error at each trajectory point by adjusting the triangulated scene point coordinates and the camera poses according to (5.12). We stop when either a gradient threshold or a fixed number of maximum iterations is reached. In our experiments, less than 60 iterations were sufficient yielding average reprojection errors that were much smaller than a pixel.

To estimate initial 3D scene points, we use the first two frames containing a trajectory thus adopting their local scale. Since badly triangulated scene

points resulting from very small baselines or difficult object geometry may harm the optimisation, we propose to exclude points that are triangulated to implausible depths by using a threshold. However, it is also possible to choose a different camera pair instead, in order to perform the triangulation.

5.3.4 Range Image Integration

From the optimisation with bundle adjustment explained in the previous section, we have obtained a globally consistent estimation of camera motion. In return, we can now use this knowledge to compute fundamental matrices that are in accordance with the global estimation for each image pair. Then we minimise the energy from Equation 5.1 again, but this time we keep the fundamental matrix fixed, i.e. we only minimise w.r.t. the flow $\boldsymbol{w}$. By putting a high weight on the epipolar term, we can elegantly make use of the fact that stereo matching constitutes a 1D search problem only without requiring image rectification. This high weight allows to constrain the correspondences to closely obey the epipolar constraint. These correspondences define registered depth maps, which can be merged using variational range image integration techniques. To this end we define a 3D bounding box of interest Ω_3 and compute a signed distance field g within it for each input depth map. In the second step we find a cumulative signed distance function $u : \Omega_3 \to \mathbb{R}$ as a minimiser of the energy

$$E(u) = \int_{\Omega_3} \left(\sum_{i=1}^{K} c_i \, \Psi((u - g_i)^2) + \alpha \, S(\boldsymbol{\nabla} u) \right) \mathrm{d}\boldsymbol{x}, \tag{5.36}$$

where c_i are confidence weights. The unknown surface is then given by the zero level set of u. We employ the anisotropic range image integration algorithm described in the previous chapter. It incorporates a direction dependent smoothing behaviour that is capable of creating smooth surfaces while preserving ridges and corners.

5.4 Experimental Results

In order to evaluate our approach, we require a number of image sequences with ground truth camera poses available as well as an error measure. With this at hand, we will first conduct experiments to find preferable parameter settings for building trajectories and then compare our dense method to a sparse feature tracker. To this end, we have selected the popular Voodoo Camera Tracker (VCT) [111], which uses the Kanade-Lucas-Tomasi (KLT)

tracker [99] and monitors the tracked image windows to detect outliers. Let us now describe the datasets and the error measure in more detail before we come to the actual experiments.

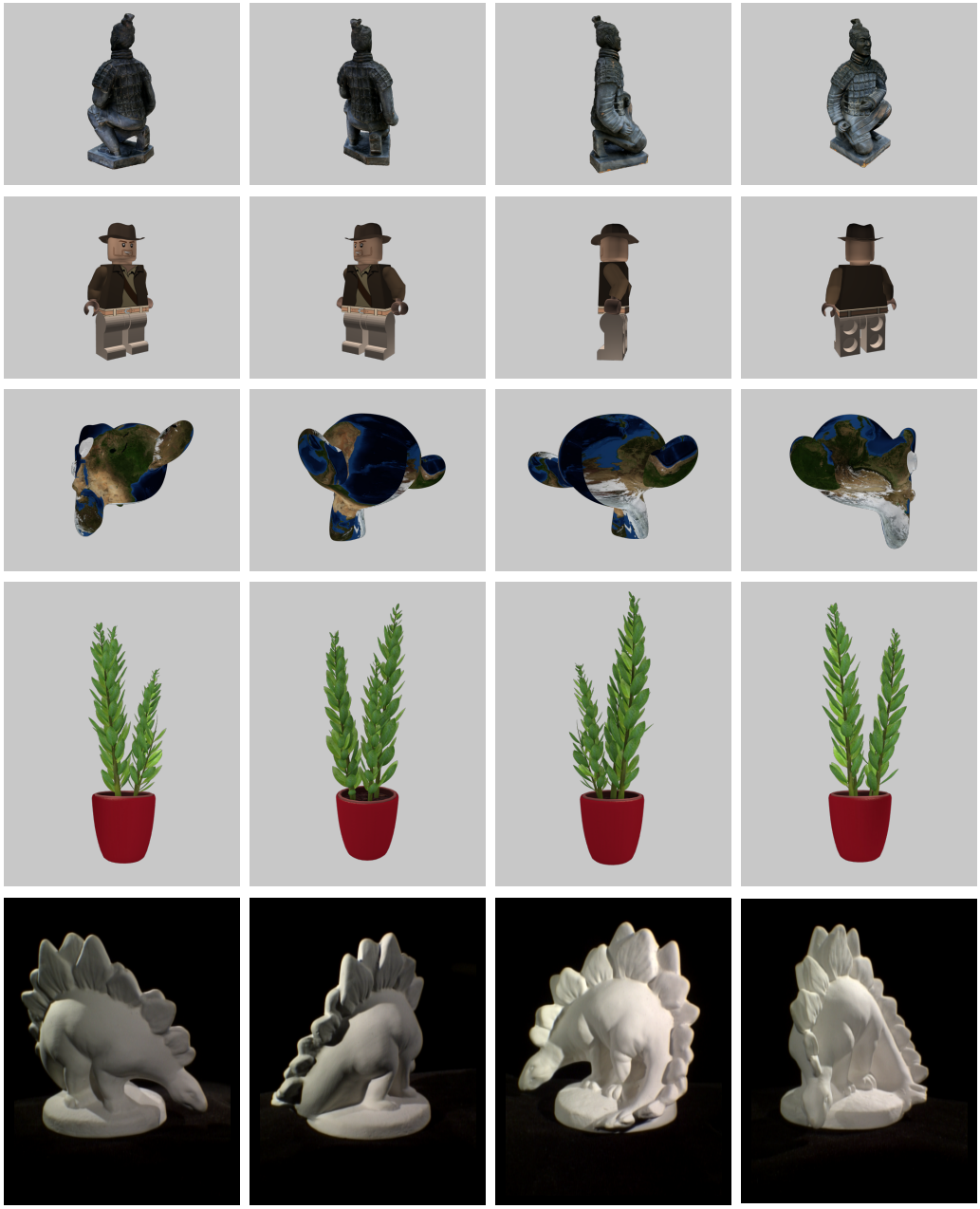


Figure 5.3: Four images out of each sequence used for the experiments. **From top to bottom**: **(a)** Terra **(b)** Indy **(c)** Monkey **(d)** Plant **(e)** Dino

Datasets

In the experimental evaluation, we will employ five image sequences which are shown in Figure 5.3. Four of these sequences are synthetic ones which have been modelled and rendered using the Blender software available at `http://www.blender.org`. The *Terra*, *Indy*, and *Plant* models are available on the *blendswap.com* platform and contain 112, 112, and 91 images, respectively. The *Monkey* model is directly available in Blender and the dataset consists of 112 images. For all synthetic datasets, the camera moves around each object while the baseline lengths between two poses vary by a factor of 2 to 3. The last sequence that we use for evaluation is the *DinoRing* dataset of the Middlebury multi-view stereo benchmark available at `http://vision.middlebury.edu/mview/`. It contains 48 images.

Error Measure

In order to quantitatively evaluate the estimated camera poses we require an error measure. However, as explained before, even after the bundle adjustment step the overall global scale is unknown and the first camera has been arbitrarily set to coincide with the world coordinate system. Therefore, the computed solution still leaves room for a similarity transform

$$\boldsymbol{T} = \begin{pmatrix} s\boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{pmatrix} \in \mathbb{R}^{4\times 4}, \tag{5.37}$$

where $\boldsymbol{R} \in \mathrm{SO}(3)$ is a rotation matrix, $\boldsymbol{t} \in \mathbb{R}^3$ a translation vector, and $s > 0$ a scale factor. To eliminate this degree of freedom, we find the transformation that optimally aligns the camera centres $\bar{\boldsymbol{c}}_i$ of the ground truth with the estimated ones $\boldsymbol{c}_i$ by minimising the least squares error

$$E(\boldsymbol{T}) = \sum_{i=1}^{K} \left| \begin{pmatrix} \bar{\boldsymbol{c}}_i \\ 1 \end{pmatrix} - \boldsymbol{T} \begin{pmatrix} \boldsymbol{c}_i \\ 1 \end{pmatrix} \right|^2 . \tag{5.38}$$

We use the closed-form solution of Horn et al. [54] to find the optimal similarity transform $\boldsymbol{T}$. This allows to align the estimated camera poses to the ground truth ones. Subsequently, we compute the pose error w.r.t. the camera position and the camera rotation. The error in camera position is simply evaluated as the average of the Euclidean distances from the estimated camera centres to their ground truth counterparts:

$$e_{\boldsymbol{c}} = \frac{1}{K} \sum_{i=1}^{K} \left| \bar{\boldsymbol{c}}_i - \boldsymbol{c}_i \right| . \tag{5.39}$$

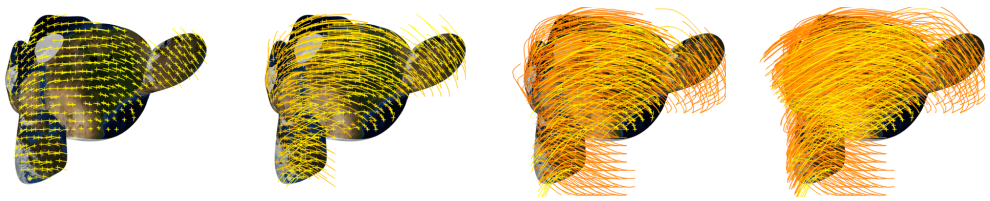


Figure 5.4: Visualisation of a subset of trajectories for the Monkey dataset. From left to right the maximum lengths are 4, 8, 16, and 32.

Measuring the error in camera rotations is not that straightforward and requires some more consideration. Obviously, building an error measure that solely relies on the viewing direction of the cameras is not sufficient because it neglects rotation around the optical axes. Instead we require a distance measure for 3D rotations. Such measures have for example been compared and analysed by Huynh [55]. In this context, it is useful to switch from rotation matrices to a quaternion representation [50] of rotation and to use the function

$$\mathrm{d}(\boldsymbol{q}_1, \boldsymbol{q}_2) = 2 \cdot \arccos\left(\left| \langle \boldsymbol{q}_1, \boldsymbol{q}_2 \rangle \right| \right) \tag{5.40}$$

to measure the distance between two unit quaternions $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$. We use the factor 2 such that we obtain values in the interval $[0, \pi]$. We then evaluate the error in rotation as the average of such distance values

$$e_{\boldsymbol{R}} = \frac{1}{K} \sum_{i=1}^{K} \mathrm{d}(\bar{\boldsymbol{q}}_i, \boldsymbol{q}_i), \tag{5.41}$$

where $\bar{\boldsymbol{q}}_i$ and $\boldsymbol{q}_i$ are unit quaternions representing the rotation of the i-th ground truth camera and the i-th estimated camera, respectively.

Maximum Length of Trajectories

As described in Section 5.3.2, we add up flow vectors of subsequent flow fields in order to obtain trajectories that span over multiple images. Figure 5.4 visualises a subset of these trajectories with different maximum lengths for the Monkey dataset. Here a single trajectory describes how the projection of a specific 3D point moves in the image sequence. Concatenating flow vectors to obtain trajectories is necessary, because otherwise we would be left with pairwise constraints between cameras. This would not allow to resolve the scale ambiguities between subsequent camera pairs. Thus, it is clear that the maximum length of trajectories should be larger than 2. However,

Max		4	8	16	32	64
Dino	$e_{\boldsymbol{c}}$	0.019	0.008	**0.007**	0.278	0.356
	$e_{\boldsymbol{R}}$	0.043	0.016	**0.014**	0.383	0.420
Monkey	$e_{\boldsymbol{c}}$	0.041	0.019	**0.014**	0.018	0.163
	$e_{\boldsymbol{R}}$	0.015	0.007	**0.004**	0.005	0.028
Indy	$e_{\boldsymbol{c}}$	1.122	0.451	0.299	**0.275**	4.098
	$e_{\boldsymbol{R}}$	0.161	0.054	0.032	**0.028**	0.527
Plant	$e_{\boldsymbol{c}}$	0.453	0.415	0.287	**0.170**	0.173
	$e_{\boldsymbol{R}}$	0.063	0.058	0.040	**0.025**	0.026
Terra	$e_{\boldsymbol{c}}$	0.083	0.058	0.051	0.058	**0.046**
	$e_{\boldsymbol{R}}$	0.013	0.009	0.006	0.007	**0.005**

Table 5.1: Errors in camera position $e_{\boldsymbol{c}}$ and camera rotation $e_{\boldsymbol{R}}$ for an increasing maximum trajectory length.

allowing trajectories that span too many images of a sequence can harm the reconstruction quality because errors in the flow fields may accumulate.

To find a good limit for trajectory lengths, we list the errors of the recovered camera poses for different maximum lengths in Table 5.1. Please note that this is only a constraint on the maximum length of trajectories. For example, we do also keep trajectories of length 2. Even though they do not contribute towards finding a unique global scale, they can still be beneficial for fixing the relative pose between two cameras.

Basically Table 5.1 shows that the additional information in longer trajectories improves the accuracy of the estimated camera poses. However, as expected the accumulated errors start to degrade the estimation quality at some point. Both 16 and 32 seem to be a reasonable limit for the maximum trajectory length for most sequences. In the remainder of the experiments we will use 16 as maximum length because it is better suited for the short Dino sequence and also computationally slightly less demanding.

Uniform Subsampling

Since we have computed dense flow fields for all subsequent camera pairs, it is possible to start trajectories for every pixel. However, this typically leads to a large memory requirement and computing time given typical image resolutions and sequence lengths. As a remedy, one can use a simple uniform subsampling to greatly reduce the computational effort and memory

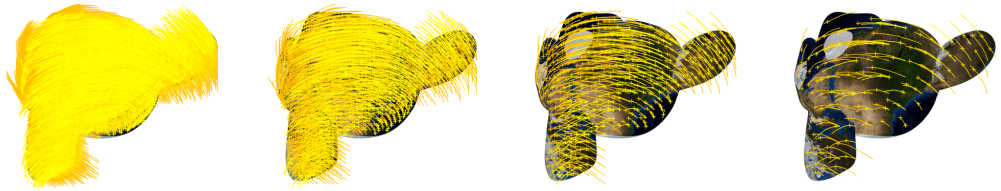


Figure 5.5: Uniform subsampling of trajectories with factors 4,8,16, and 32 from left to right.

Grid		2	4	8	16	32	64
Dino	$e_{\boldsymbol{c}}$	0.007	0.007	0.007	0.007	**0.006**	0.008
	$e_{\boldsymbol{R}}$	0.014	0.014	0.014	0.014	**0.011**	0.018
Monkey	$e_{\boldsymbol{c}}$	0.015	0.014	**0.014**	0.018	0.016	0.035
	$e_{\boldsymbol{R}}$	0.005	0.005	**0.004**	0.006	0.005	0.012
Indy	$e_{\boldsymbol{c}}$	0.269	**0.263**	0.299	0.338	0.282	0.415
	$e_{\boldsymbol{R}}$	0.031	0.029	0.032	0.035	**0.029**	0.053
Plant	$e_{\boldsymbol{c}}$	0.278	0.291	0.287	0.266	**0.178**	0.318
	$e_{\boldsymbol{R}}$	0.040	0.041	0.040	0.037	**0.026**	0.044
Terra	$e_{\boldsymbol{c}}$	**0.048**	0.051	0.051	0.054	0.083	0.285
	$e_{\boldsymbol{R}}$	**0.005**	0.005	0.006	0.006	0.009	0.036

Table 5.2: Errors in camera position $e_{\boldsymbol{c}}$ and camera rotation $e_{\boldsymbol{R}}$ for varying grid size.

requirement. Figure 5.5 shows trajectories obtained with different subsampling factors for the Monkey dataset. Even with such a subsampling, the pipeline can still be interpreted as a fully dense method that partially operates on low resolution images. Table 5.2 shows the accuracy of estimated camera poses for different subsampling factors. Varying the subsampling grid tends to affect the accuracy of the estimated camera poses only slightly and we can observe that a too coarse sampling of trajectories tends to give the worst results. Thus, we choose a subsampling factor of 8 for the remainder of the experiments because this usually gives a good compromise between computation time and memory usage as well as a good accuracy.

Structure Based Sampling

As we have mentioned previously, Sundaram et al. [106] have presented a structure based sampling strategy for determining trajectory starting points. In their approach, they consider the structure tensor [38]

$$K_\rho * \left(\nabla(K_\sigma * f_i)\nabla(K_\sigma * f_i)^\top\right) \tag{5.42}$$

of an image f_i where K_σ and K_ρ are Gaussian kernels. More specifically, they use some portion of the mean of the second eigenvalue of the structure tensor as a threshold to determine the locations where new trajectories are started.

In this experiment, we follow this idea and investigate how such a structure based sampling strategy affects the accuracy of the reconstructed camera poses. We also use the second eigenvalue of the structure tensor as a criterion to start new trajectories. However, we choose the threshold in such a way that the amount of resulting trajectories for the structure based approach is in the same order of magnitude as in the case of the uniform sampling that we compare to. Furthermore, we also examine the structure based approach in combination with subsampling to a coarser grid.

Figure 5.6 visualises trajectory starting points of the structure based approach for different grids. It shows that combining both approaches allows to obtain a compromise between respecting structural information and having an even distribution over the whole model. Again we adapt the threshold in each case such that the amount of trajectories is in the same order of magnitude.

Table 5.3 lists the resulting error values. They show that a pure structure based sampling typically leads to worse results in this scenario. When combining both ideas, the structure based sampling can in two cases slightly improve the accuracy of the camera poses. However, respecting the local texture measure does not generally increase the accuracy of the recovered camera poses. This underlines that the dense flow fields are not only reliable in highly textured regions. Therefore, we refrain from structure based sampling ideas and stick to the simple uniform approach in the remainder of the experiments.

Comparison to Voodoo Camera Tracker (VCT)

Table 5.4 shows the errors in position and orientation of the final globally aligned model estimate for the evaluation sequences. They are being compared with the errors of the VCT. Also for VCT, we supply the intrinsic camera calibration as a fixed parameter for all cameras. Our dense method

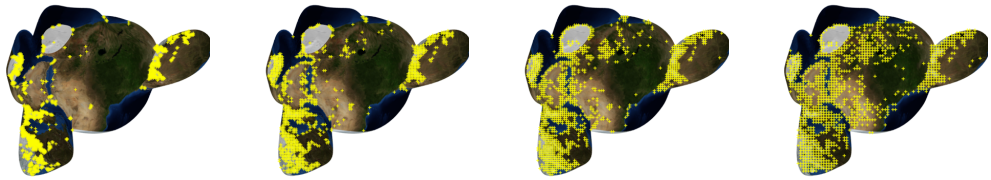


Figure 5.6: Structure based sampling. From left to right the underlying grid on which the structure based sampling is performed becomes coarser such that the trajectory starting points become more evenly distributed.

		Structure based					Uniform
Grid		**1**	**2**	**3**	**4**	**5**	**8**
Dino	$e_{\boldsymbol{c}}$	0.051	0.010	0.008	**0.006**	0.006	0.007
	$e_{\boldsymbol{R}}$	0.074	0.017	0.015	**0.012**	0.013	0.014
Monkey	$e_{\boldsymbol{c}}$	0.066	0.034	0.027	0.018	0.019	**0.014**
	$e_{\boldsymbol{R}}$	0.027	0.012	0.009	0.006	0.006	**0.004**
Indy	$e_{\boldsymbol{c}}$	0.355	0.302	**0.247**	0.272	0.267	0.299
	$e_{\boldsymbol{R}}$	0.042	0.033	**0.028**	0.030	0.030	0.032
Plant	$e_{\boldsymbol{c}}$	0.423	0.352	0.337	0.363	0.314	**0.287**
	$e_{\boldsymbol{R}}$	0.050	0.042	0.041	0.047	0.042	**0.040**
Terra	$e_{\boldsymbol{c}}$	0.141	0.102	0.061	0.070	0.055	**0.051**
	$e_{\boldsymbol{R}}$	0.014	0.009	0.005	0.006	**0.005**	0.006

Table 5.3: Structure based vs. uniform sampling of trajectory starting points. The structure based approach is performed on different grids where a grid of one corresponds to no subsampling and thus a pure structure based approach.

turns out to yield competitive results compared to VCT which relies on sparse features. Actually, it is able to outperform VCT on all models except for the Monkey dataset. Since this dataset is highly textured, it is comprehensible that sparse feature tracking achieves very good results. However, also in this case we are able to obtain a comparable accuracy. Figure 5.7 shows the reconstructed camera motion along with the 3D points estimated by VCT. As a comparison, we show the 3D points estimated within our pipeline when using a uniformly subsampled grid with factor 8. Here we can observe that our approach leads to much denser and quite uniformly distributed point clouds. Figure 5.8 shows the resulting 3D reconstructions for the used datasets with our dense pipeline. Since it is not within the scope of this thesis to estimate accurate material propertis, we resort to a very naive strategy to obtain the textured reconstructions: We simply estimate the colour of a point on the surface by projecting it into all cameras and then averaging the individual colour values from all cameras where the corresponding surface point is visible.

5.5 Limitations and Discussion

Bundle adjustment can lead to an undesired local minimum for an erroneous initialisation. Since the main source of error in our initial model is the varying scaling between scene points triangulated from different camera pairs, it might be beneficial to only optimise for the baseline scales in a first step. This can help guiding a subsequent full bundle adjustment into a preferable minimum. A similar approach is taken in the odometry method of Fraundorfer et al. [39].

Loop closure is not directly included into our pipeline since the trajectories are initially computed over neighbouring frames of an image sequence. However, after computing camera poses and the 3D reconstruction with our pipeline, it is possible to identify new frame correspondences to extend existing trajectories. This can help to solve the problem of closing loops when iterating the pipeline a second time.

Errors in the optical flow are not corrected by the bundle adjustment step, since the trajectory constraints build upon the flow field. Furthermore, the flow fields are interpolated between grid points when computing trajectories, which can introduce additional errors. However, usually this constitutes a rather small error and it is possible to limit the maximum length of trajectories. In terms of variational methods one could estimate the flow over multiple frames simultaneously. It is well-known that one can increase the flow quality by using temporal regularisation; see e.g. [117, 113].

Model		VCT (KLT)	Ours
Dino	$e_{\boldsymbol{c}}$	0.010	**0.007**
	$e_{\boldsymbol{R}}$	0.019	**0.014**
Monkey	$e_{\boldsymbol{c}}$	**0.013**	0.014
	$e_{\boldsymbol{R}}$	0.005	**0.004**
Indy	$e_{\boldsymbol{c}}$	0.437	**0.299**
	$e_{\boldsymbol{R}}$	0.054	**0.032**
Plant	$e_{\boldsymbol{c}}$	0.809	**0.287**
	$e_{\boldsymbol{R}}$	0.105	**0.040**
Terra	$e_{\boldsymbol{c}}$	0.094	**0.051**
	$e_{\boldsymbol{R}}$	0.011	**0.006**

Table 5.4: Accuracy of the sparse Voodoo Camera Tracker (VCT) compared to our dense approach.

5.6 Summary

We have presented a novel pipeline for 3D reconstruction that completely relies on dense methods. From an image sequence of a static scene, optical flow fields and stereo geometry are jointly estimated for each consecutive image pair with a variational approach. Subsequently, the pairwise estimates are connected and globally refined through bundle adjustment. After this step, we obtain a refined model of the scene camera positions. Using these, depth maps are computed and fused by anisotropic range image integration.

Even for bad initialisations our optical flow based trajectories prove to be sufficiently robust constraints, enabling the bundle adjustment to recover the correct global model. This and our comparisons to VCT show that dense approaches are an interesting alternative to sparse methods.

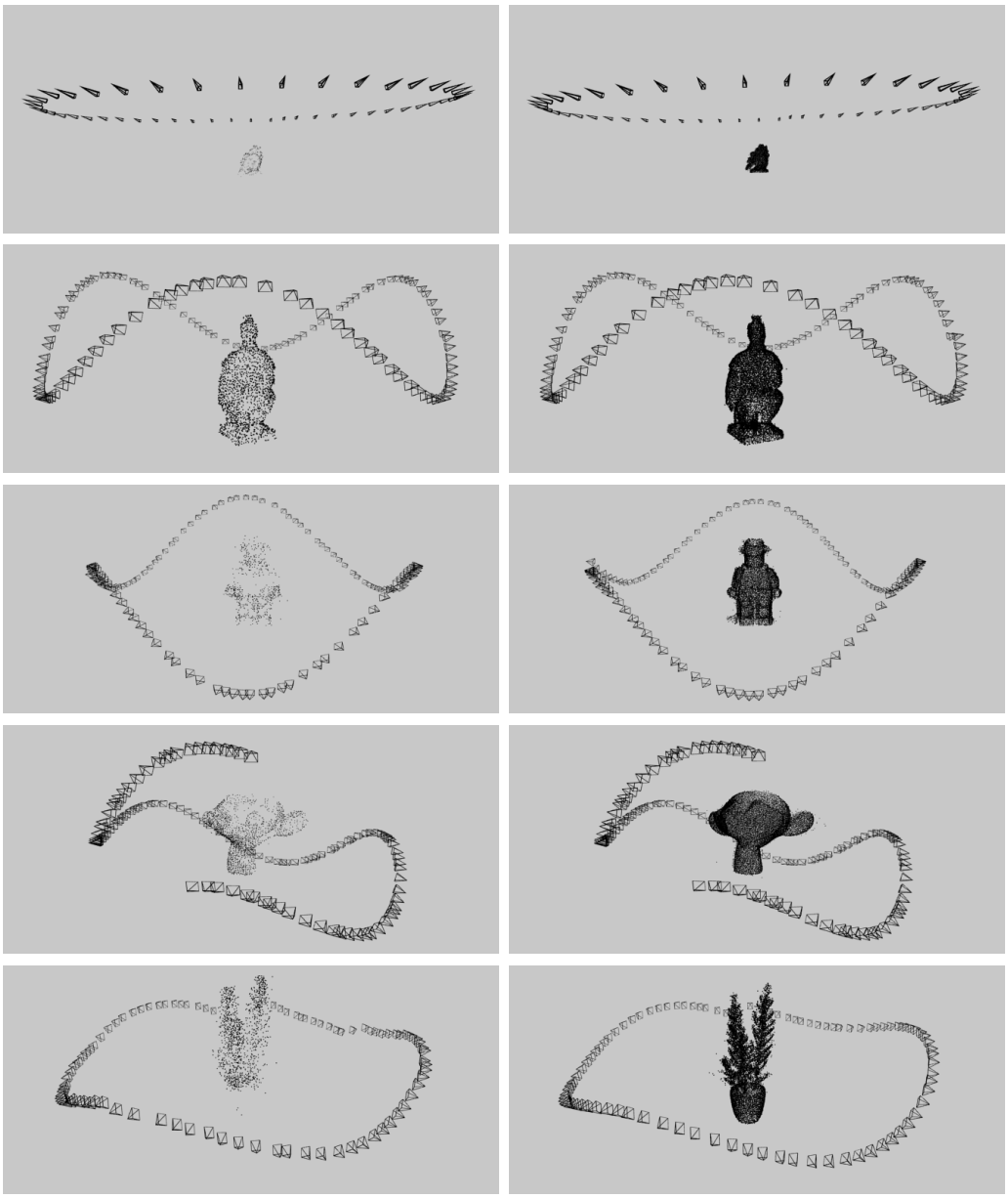


Figure 5.7: Reconstructed camera motion and 3D points for the Voodoo Camera tracker (left) and for our approach with a subsampling factor of 8 (right).

Figure 5.8: Two different views of each 3D reconstruction obtained with our dense pipeline. Each view is shown without and with texture.

Chapter 6

Surface Reconstruction from Oriented Points

An oriented point of a surface contains information about the position and the surface normal. The problem of reconstructing a watertight surface from a finite set of oriented points is visualised in Figure 6.1. Oriented point clouds arise in 3D reconstruction pipelines that use passive methods relying on sparse features. A prominent example for this is the patch-based multi-view stereo reconstruction algorithm of Furukawa and Ponce [40]. Oriented point clouds can also be obtained with active methods such as laser, structured light and time-of-flight scanning. Due to viewpoint dependence, usually many scans have to be acquired and subsequently aligned to cover the whole surface. Each surface measurement can conveniently be equipped with the direction to the source or an even better approximation of surface orientation by considering neighbouring measurements of one scan. Also many methods exist to estimate normals from point clouds, see e.g. [17].

Reconstructing an accurate surface is a difficult task. In practice it often occurs that some parts of the surface cannot be captured. Furthermore, one has to deal with uneven sampling due to overlapping scans, and the samples contain noise caused by inaccuracies of the sensor. Misalignments of the individual scans further increase the difficulty. The reconstruction problem is even more cumbersome when using multi-view stereo reconstruction algorithms on data captured from consumer grade cameras in uncontrolled environments. Due to this fact and because nowadays point clouds easily contain many millions of points, it is essential to apply robust and efficient algorithms. It is a common practice to fit the oriented points using a level set of an implicit function. Such methods can produce approximating surfaces, which is preferable if noise and outliers are present. Furthermore, they have the inherent advantage that one does not have to parameterise the surfaces.

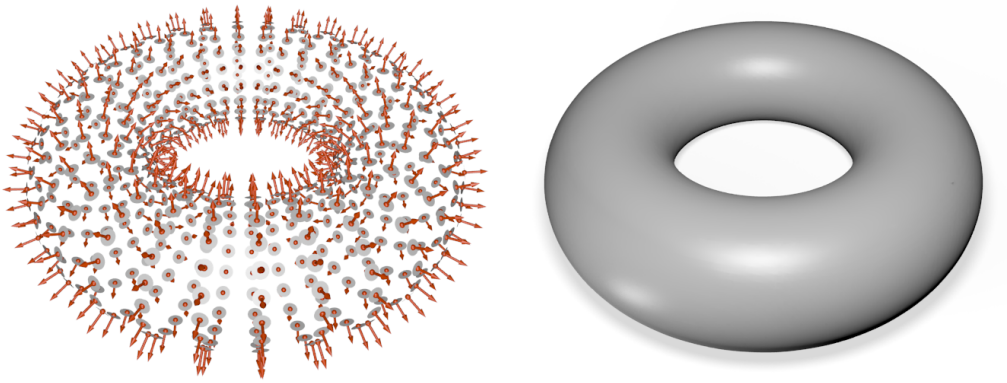


Figure 6.1: Given a finite number of oriented points (left), the goal is to estimate a watertight 3D model (right).

Many different approaches exist, and commonly the implicit function is either an approximation to the indicator function or the signed distance function of the underlying surface. As there exists a variety of different methods, it would be beneficial to have a joint platform that allows to explicitly display similarities and differences. From a didactic point of view, this gives an opportunity to explain existing methods to people new in this field. Furthermore, it offers a systematic approach for deriving novel methods. Thus, we present a general higher order framework for surface reconstruction in this chapter which we have published in [7].

Organisation of this chapter. After covering related work and listing our contributions, Section 6.3 introduces our general framework with higher order terms. Section 6.4 then classifies existing approaches within this framework and Section 6.5 describes our novel approaches for surface reconstruction. In Section 6.6, we show how colour information can be used to obtain textured reconstructions. Subsequently, we discuss our GPU implementation in Section 6.7 followed by experimental results in Section 6.8. The following discussion in Section 6.9 reveals an additional opportunity for improving the reconstruction quality. To this end, we introduce a hull constraint in Section 6.10. Finally, we summarise the most important insights in Section 6.11.

6.1 Related Work

We will focus on methods that fit the input data using a level set of an implicit function. As mentioned, the implicit function is either an approximation to the indicator function or the signed distance function of the underlying surface in many cases. Therefore, we use this as a criterion to broadly categorise prior work.

Indicator Function Approximation

Kazhdan et al. estimate the indicator function by first computing a vector field that approximates the smoothed surface normal field and then integrating it in the least squares sense [59]. This approach is known as Poisson surface reconstruction and has been extended by Kazhdan and Hoppe [58] who add an explicit point-wise constraint on the function value at the input points. Manson et al. reconstruct the indicator function using a wavelet basis [74]. As each sample point only influences a small number of coefficients, the reconstruction is very fast. Lempitsky and Boykov find a compromise between the number of collected input points and the surface area [68]. They minimise the resulting energy over binary functions using graph-cuts.

Signed Distance Function Approximation

A popular approach for estimating the signed distance function from a set of oriented points is the implicit moving least squares (IMLS) algorithm proposed by Shen et al. in [98]. Kolluri analyses a variant of this algorithm that uses constant basis functions [65]. He is able to show that it yields geometrically and topologically correct reconstructions if certain sampling conditions are fulfilled. In the presence of sharp features, it can make sense to use robust variants, see [36, 83]. Calakli and Taubin estimate an approximation of the signed distance function using a smoothing thin plate spline with additional pointwise constraints on the normals [21]. To minimise the energy, they employ a hybrid finite element / finite difference discretisation on an octree structure. Walder et al. consider the same energy but aim at expressing the solution as weighted sum of kernel functions centred at the input points [114]. If a triangle representation is desired in the end, an isosurface can be extracted from the implicit function with algorithms such as Marching Cubes [71].

6.2 Contributions

We develop a general higher order variational framework for surface reconstruction from oriented points. It is based on the idea that each oriented point allows us to construct a function that provides a good local description of an implicit representation of the unknown surface. This framework allows us to reach two goals: First, we can systematically understand and classify a number of existing methods. Second, it enables us to derive novel approaches to surface reconstruction that are fairly simple and offer state-of-the-art performance. We show with the recent reconstruction benchmark of Berger et al. [15] that one of these approaches yields favourable results when compared to the most popular and widely used methods, namely (screened) Poisson surface reconstruction and smooth signed distance surface reconstruction. Furthermore, we introduce a hull constraint that encourages the surface to stay within a given region. This improves reconstructions in difficult real world scenarios where point clouds have been estimated from colour images.

6.3 A Higher Order Framework

In this section we describe how point and normal constraints can be used in a general variational formulation with higher order terms in order to reconstruct smooth surfaces. The reconstruction is then implicitly given by the boundary of the zero level set of the minimiser. To describe our framework, we first begin with some basic definitions.

Basic Definitions

Let us assume that a set of N oriented points

$$\Big\{(\boldsymbol{p}_i, \boldsymbol{n}_i) \in \mathbb{R}^3 \times \mathbb{R}^3 \,\Big|\, i = 1, \dots, N\Big\} \tag{6.1}$$

has been sampled from a smooth surface. Here $\boldsymbol{p}_i$ and $\boldsymbol{n}_i$ denote location and normal, respectively. Then the surface can locally be approximated by a sufficiently small linear patch given by

$$\Big\{\boldsymbol{x} \in B_\sigma(\boldsymbol{p}_i) \,\Big|\, \langle \boldsymbol{x} - \boldsymbol{p}_i, \boldsymbol{n}_i \rangle = 0\Big\}, \tag{6.2}$$

i.e. a subset of the tangent plane to the surface at $\boldsymbol{p_i}$ (cf. Figure 6.2). Here $B_\sigma(\boldsymbol{p}_i)$ denotes an open ball with a small radius $\sigma > 0$ centred around $\boldsymbol{p}_i$. Accordingly, the signed distance function can also be well approximated locally

around $\boldsymbol{p}_i$ by

$$f_i(\boldsymbol{x}) = \langle \boldsymbol{x} - \boldsymbol{p}_i, \boldsymbol{n}_i \rangle. \tag{6.3}$$

This gives rise to a data fidelity term

$$D(u) = \sum_{i=1}^{N} \int_{B_\sigma(\boldsymbol{p}_i)} \Big(u(\boldsymbol{x}) - f_i(\boldsymbol{x}) \Big)^2 \, \mathrm{d}\boldsymbol{x} \tag{6.4}$$

that rewards a close fit to the given functions by penalising a locally weighted squared L^2-distance to each of the functions $f_i(\boldsymbol{x})$. In the above equation, the deviation at each location within $B_\sigma(\boldsymbol{p}_i)$ is penalised with equal weight. In order to generalise this, we rewrite Equation 6.4 as

$$D(u) = \sum_{i=1}^{N} \int_{\Omega} w_\sigma(|\boldsymbol{x} - \boldsymbol{p}_i|) \Big(u(\boldsymbol{x}) - f_i(\boldsymbol{x}) \Big)^2 \, \mathrm{d}\boldsymbol{x}, \tag{6.5}$$

where

$$w_\sigma(s) = \begin{cases} 1, & \text{if } |s| \leq \sigma \\ 0, & \text{otherwise.} \end{cases} \tag{6.6}$$

The domain $\Omega \subset \mathbb{R}^3$ is a region that contains all local approximations and $u : \Omega \to \mathbb{R}$. In this notation, it becomes apparent that one can conveniently use an arbitrary weighting function instead of a hard window. An often preferable choice is a smooth, decaying function such as a Gaussian:

$$w_\sigma(s) = \exp\left(- \left(\frac{s}{\sigma} \right)^2 \right). \tag{6.7}$$

This allows for good reconstructions even if coarser approximations with fewer linear patches are used, because the values further away from $\boldsymbol{p}_i$, that are usually less reliable, are only taken into account with a very small weight.

Higher Order Energy

So far, we have penalised deviations in function values only. In a general setting, we would now like to allow penalising the difference in all derivatives up to order K directly. To this end we first define a data term for the k-th derivative:

$$D_k(u) = \sum_{i=1}^{N} \int_{\Omega} \frac{w_{i,k}}{d_k} \Big| \mathcal{D}^{(k)} \Big(u - f_i \Big) \Big|^2 \mathrm{d}\boldsymbol{x}, \tag{6.8}$$

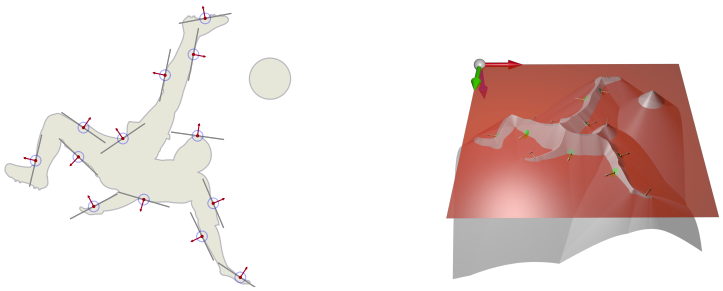


Figure 6.2: **(Left):** The oriented points define tangent planes which can be used to construct local approximations to the surface. **(Right):** They also allow to approximate the signed distance function to the surface in a small neighbourhood.

where we omit the dependence on $\boldsymbol{x}$ for better readability. We allow choosing different weighting functions $w_{i,k}$ for each order of derivative and sample. As we will see, we can create pointwise, localised or global constraints by different choices of the weighting functions. The term d_k accounts for a possible normalisation. It can either be a constant or a function that allows for a pointwise reweighting. The term $\mathcal{D}^{(k)}$ is a differential operator that results in a vector of all derivatives of order k when applied to a function. Combining these data terms with a suitable smoothness functional $S(u)$ yields the energy

$$E(u) = \sum_{k=0}^{K} \alpha_k \, D_k(u) \; + \; \alpha \, S(u). \tag{6.9}$$

The weights $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_K)^\top$ specify how strong deviations in each derivative should be penalised. This broad perspective gives a systematic way of approaching the surface reconstruction problem.

6.4 Relation of Existing Methods

In this chapter, we will consider the cases $K = 0, 1, 2$ with $\alpha = 0$, i.e. eliminating a dedicated smoothness term. This allows to systematically explain and relate several popular surface reconstruction approaches with our higher order framework.

6.4.1 Implicit Moving Least Squares ($K = 0$)

A choice of $K = 0$ means that we only consider a single data term that penalises the deviation in function values. It has the analytic solution

$$u(\boldsymbol{x}) = \frac{\sum_{i=1}^{N} w_{i,0}(\boldsymbol{x}) \, \langle \boldsymbol{x} - \boldsymbol{p}_i, \boldsymbol{n}_i \rangle}{\sum_{i=1}^{N} w_{i,0}(\boldsymbol{x})}, \tag{6.10}$$

where one can choose a weighting function $w_{i,0}(\boldsymbol{x}) = a_i \exp(-{}^{|\boldsymbol{x}-\boldsymbol{p}_i|^2}/_{\sigma_i^2})$ with a varying standard deviation σ_i modified by a normalisation factor a_i for each input point. This corresponds to the *implicit moving least squares* (IMLS) algorithm with constant basis functions, which was proposed by Shen et al. [98]. The *partition of unity* (PoU) approach of Ohtake et al. [82] is very close in spirit and can even be equivalent. The weighting functions $w_{i,0}$ are carefully adapted for each individual sample. The choice of σ_i and a_i generally depend on the sampling density, which relates to the size of features one may expect. Since there is no smoothness term, a suitable choice of σ_i and a_i has to allow for removing isolated clutter and closing gaps. However, often it is not possible to find parameters that fulfil this while preserving details. Therefore, IMLS-based reconstructions can exhibit spurious artefacts. Examples for this have also been shown in [21, 58] when comparing to the approach of Ohtake et al. [82].

6.4.2 (Screened) Poisson Surface Reconstruction ($K = 1$)

Let us start by only considering the first order data term, i.e. choosing $\boldsymbol{\alpha} = (0, 1)^\top$. Please recall that $f_i(\boldsymbol{x}) = \langle \boldsymbol{x} - \boldsymbol{p}_i, \boldsymbol{n}_i \rangle$ and thus $\boldsymbol{\nabla} f_i(\boldsymbol{x}) = \boldsymbol{n}_i$. This results in the energy

$$E(u) = \sum_{i=1}^{N} \int_{\Omega} \frac{w_{i,1}}{d_1} |\boldsymbol{\nabla} u - \boldsymbol{n}_i|^2 \, \mathrm{d}\boldsymbol{x}, \tag{6.11}$$

where $w_{i,1}$ describes a local weighting, for example via a Gaussian function. We have chosen a pointwise normalisation factor $d_1(\boldsymbol{x})$ that accounts for uneven sample placements:

$$d_1(\boldsymbol{x}) = \sum_{i=1}^{N} w_{i,1}(\boldsymbol{x}). \tag{6.12}$$

The above energy can be rewritten as

$$E(u) = \int_{\Omega} |\boldsymbol{\nabla} u - \boldsymbol{v}|^2 \, \mathrm{d}\boldsymbol{x} + R, \tag{6.13}$$

where the last expression R is a constant that contains residual terms that do not influence the minimiser, and $\boldsymbol{v}$ is a convex combination of the given normals:

$$\boldsymbol{v} = \sum_{i=1}^{N} \frac{w_{i,1}}{d_1} \boldsymbol{n}_i. \tag{6.14}$$

Interestingly, one can interpret this as solving two subsequent minimisation problems. First, a vector field is computed based on the input normals by solving a moving least squares problem similar as in the previous section but this time for the normal vectors. Then one finds the unknown function u in a second step by solving the above energy. Such an approach is described in [100].

Poisson Surface Reconstruction

Obviously the energy in Equation 6.13 leads to a Poisson equation. For a suitable choice of $\boldsymbol{v}$ it resembles *Poisson surface reconstruction* [59]. The weighting required for this is motivated by an interesting observation: The gradient of the smoothed indicator function and the smoothed surface normal field are equal. Thus, $\boldsymbol{v}$ can be understood as an approximation of the smoothed surface normal field that can be obtained by performing a numerical integration

$$\boldsymbol{v} = \sum_{i=1}^{N} |\mathcal{P}_i| w_{i,1} \boldsymbol{n_i}, \tag{6.15}$$

where the patch sizes $|\mathcal{P}_i|$ are estimated with a density estimator. High densities relate to small patches and vice versa. While this seems to be a subtle change from Equation 6.14 to Equation 6.15, the effect is rather large: With this weighting, the length of the vectors $\boldsymbol{v}$ actually decreases towards zero when going away from the input points, as required for an indicator function. With the previous choice one estimates a distance field instead.

In both cases, the solution can only be computed up to a global constant. This generally does not have to pose a problem, but sometimes it is not possible to find a satisfactory one. Let us consider a somewhat artificial but very simple example to illustrate this. If the same surface orientation is measured everywhere, this will result in a constant vector field when using Equation

6.14. Thus, the reconstructed surface will be planar for any global offset. If the locations of the normals do not happen to be on a line, the reconstruction will thus appear to drift away from the input points. In practice, such problems have also been observed for Poisson surface reconstruction, where the input points were not fitted tightly enough and the reconstructions were too smooth.

Screened Poisson Surface Reconstruction

To account for this, Kazhdan and Hoppe add point constraints and show that this results in more accurate reconstructions [58]. With an appropriate α_0, they minimise the energy

$$E(u) = \alpha_0 \sum_{i=1}^{n} u(\boldsymbol{p}_i)^2 + \int_\Omega |\boldsymbol{\nabla} u - \boldsymbol{v}|^2 \, \mathrm{d}\boldsymbol{x}, \tag{6.16}$$

over a suitable space of functions. The first term (screening term) of order zero can be obtained in our framework by choosing the Dirac distribution for each $w_{i,0}$. This effectively creates point constraints instead of constraints that are localised to some neighbourhood.

Relation to Global Optimisation for Shape Fitting

Another interesting observation is that the *global optimisation for shape fitting* approach of Lempitsky and Boykov [68] can be closely related to Poisson surface reconstruction. To see this, we rewrite Equation 6.13: We expand the scalar product and perform partial integration:

$$\begin{aligned} E(u) &= \int_\Omega |\nabla u - \boldsymbol{v}|^2 \, \mathrm{d}\boldsymbol{x} && (6.17)\\ &= \int_\Omega \left(|\nabla u|^2 - 2 \cdot \langle \nabla u, \boldsymbol{v} \rangle + |\boldsymbol{v}|^2 \right) \mathrm{d}\boldsymbol{x} && (6.18)\\ &= \int_\Omega \left(|\nabla u|^2 + 2 \cdot u \cdot \mathrm{div} \boldsymbol{v} + |\boldsymbol{v}|^2 \right) \mathrm{d}\boldsymbol{x} - 2 \cdot \int_{\partial\Omega} \langle u\boldsymbol{v}, \boldsymbol{n} \rangle \mathrm{d}s. && (6.19) \end{aligned}$$

Subsequently, we introduce the parameters α and β in the following way:

$$E(u) = \alpha \int_\Omega |\boldsymbol{\nabla} u|^{\beta+1} \mathrm{d}\boldsymbol{x} + \int_\Omega 2u \cdot \mathrm{div}\, \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} + \beta \, G(u). \tag{6.20}$$

The expression $G(u)$ is composed of three terms, where the first two of them are constant and do not have an effect in the minimisation. The last one influences the natural boundary condition. Explicitly $G(u)$ is given by

$$G(u) = R + \int_\Omega |\boldsymbol{v}|^2 \, \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} \langle u\,\boldsymbol{v}, \boldsymbol{n}\rangle \, d\boldsymbol{s}. \tag{6.21}$$

Note that we recover Equation 6.13 for $\alpha = \beta = 1$. Setting $\beta = 0$ yields the energy minimised by Lempitsky and Boykov over binary functions.

Although very differently motivated, the vector field $\boldsymbol{v}$ is in both cases computed as a weighted sum of the normals using Gaussian weighting functions. In both methods, the standard deviation is adapted based on sample spacing as it is also common for moving least squares based approaches.

Lempitsky and Boykov minimise over a space of binary functions using graph cuts. In the context of surface reconstruction, the minimisation over binary functions has the drawback that it creates aliasing problems which have to be taken care of when extracting the isosurface.

6.4.3 Smooth Signed Distance Surface Reconstruction ($K = 2$)

Let us choose $K = 2$ in (6.9) and recall that $f_i(\boldsymbol{x}) = \langle \boldsymbol{x} - \boldsymbol{p}_i, \boldsymbol{n}_i\rangle$, $\boldsymbol{\nabla} f_i(\boldsymbol{x}) = \boldsymbol{n}_i$ and $\boldsymbol{H} f_i(\boldsymbol{x}) = 0$ with $\boldsymbol{H}$ denoting the Hessian. Then, we set

$$d_k = \sum_{i=1}^{N} \int_\Omega w_{i,k}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \tag{6.22}$$

so that each data term is automatically scaled by the area of all N weight functions. We select the Dirac distribution for all $w_{i,0}$ and $w_{i,1}$ creating pointwise constraints for input locations and normals. Furthermore, we use a constant $w_{i,2} := \frac{1}{N}$ which effectively creates a global constraint. This corresponds to the energy by Calakli and Taubin:

$$\begin{aligned} E(u) \;&= \frac{\alpha_0}{N} \sum_{i=1}^{N} |u(\boldsymbol{p}_i)|^2 \\ &+ \frac{\alpha_1}{N} \sum_{i=1}^{N} |\boldsymbol{\nabla} u(\boldsymbol{p}_i) - \boldsymbol{n}_i|^2 \\ &+ \frac{\alpha_2}{|\Omega|} \int_\Omega |\mathbf{H} u(\boldsymbol{x})|_F^2 \, \mathrm{d}\boldsymbol{x}, \end{aligned} \tag{6.23}$$

Order	Weighting	Equivalent Methods	Related Methods
0	$w_{i,0} = a_i\, e^{-\frac{\lvert \boldsymbol{x}-\boldsymbol{p}_i\rvert^2}{\sigma_i^2}}$	IMLS [98]	PoU [82]
1	$w_{i,1} = a_i\, e^{-\frac{\lvert \boldsymbol{x}-\boldsymbol{p}_i\rvert^2}{\sigma_i^2}}$	VIR [100]	PSR [59] SPSR [58] GOSF [68]
2	$w_{i,0} = \delta(\lvert \boldsymbol{x}-\boldsymbol{p}_i\rvert)$ $w_{i,1} = \delta(\lvert \boldsymbol{x}-\boldsymbol{p}_i\rvert)$ $w_{i,2} = \frac{1}{N}$	SSD [21] ISM [114]	

Table 6.1: Given an order K and weighting functions w in our general framework, this table shows which existing methods are equivalent and which can be closely related. Here δ denotes the Dirac delta.

which is minimised over a suitable function space [21]. Here $|\cdot|_F$ is the Frobenius norm. The last term can be understood as a smoothness term that arose by merely selecting appropriate weighting functions and thus it is in harmony with the constraints of the other terms.

The previous sections have shown that we are able to explain a number of existing approaches when varying the order K of our model. Some approaches directly follow from simple adaptions of the weighting functions. Relating others requires slightly more involved derivations. While we have explained these relations in detail in the previous sections, Table 6.1 sketches a brief overview.

6.5 Novel Formulations

In this section we describe three novel variational formulations for the integration of point and normal constraints derived from our general framework. We will refer to them as Hessian-IMLS, TV-IMLS and HOM-IMLS.

6.5.1 Hessian-IMLS

We have discussed that for pure IMLS-based approaches it is often not possible to find a σ to recover details while removing measurement errors and

isolated clutter. Furthermore, we have argued that a Hessian smoothness term is in harmony with the point and normal constraints and it naturally arises for a suitable choice of weighting functions when discussing the case of $K = 2$. In general, this regulariser is popular for geometrical problems [114], also due to its good filling-in behaviour in unsampled regions. It corresponds to the thin plate energy of order 2, cf. [30]. Thus, we propose to minimise an energy combining both terms:

$$E(u) = \sum_{i=1}^{N} \int_{\Omega} w_i \big(u - f_i\big)^2 \,\mathrm{d}\boldsymbol{x} + \alpha \int_{\Omega} |\boldsymbol{H}u|_F^2 \,\mathrm{d}\boldsymbol{x} \tag{6.24}$$

with

$$w_i = w_\sigma(|\boldsymbol{x} - \boldsymbol{p}_i|), \tag{6.25}$$

using a small constant $\sigma > 0$. We advocate that our model comprised of only two terms is the simplest choice possible within the higher order framework that incorporates the benefits of IMLS and an appropriate regularisation. Affine functions are not penalised: Thus, if only a single oriented point is present, the computed surface is given by the plane defined by it. Increasing the smoothness weight α leads to solutions with less curvature. Therefore, the regulariser can be understood as a low curvature prior. The corresponding Euler-Lagrange equation can be written as

$$u \sum_{i=1}^{N} w_i \;+\; \alpha \,\mathrm{div}^2(\boldsymbol{H}u) = \sum_{i=1}^{N} w_i \, f_i. \tag{6.26}$$

Operators of the form div^k with $k \in \mathbb{N}$ are commonly used when working with tensor fields as for example in [18]. Accordingly, div^2 applied to a 3-dimensional second order tensor field $\boldsymbol{M}$ results in a scalar field, i.e. a tensor field of two orders less, given by

$$\mathrm{div}^2(\boldsymbol{M}) = \sum_{i,j=1}^{3} \partial x_i \partial x_j m_{ij}. \tag{6.27}$$

The natural boundary conditions are given by:

$$\boldsymbol{n}^\top \begin{pmatrix} u_{xxx} + u_{xyy} + u_{xzz} & u_{xx} & u_{yx} & u_{zx} \\ u_{yxx} + u_{yyy} + u_{yzz} & u_{xy} & u_{yy} & u_{zy} \\ u_{zxx} + u_{zyy} + u_{zzz} & u_{xz} & u_{yz} & u_{zz} \end{pmatrix} = \boldsymbol{0}. \tag{6.28}$$

Here $\boldsymbol{n}$ is a vector normal to the boundary of the domain Ω. In the SSD approach, the Dirac distribution as weighting function effectively removes

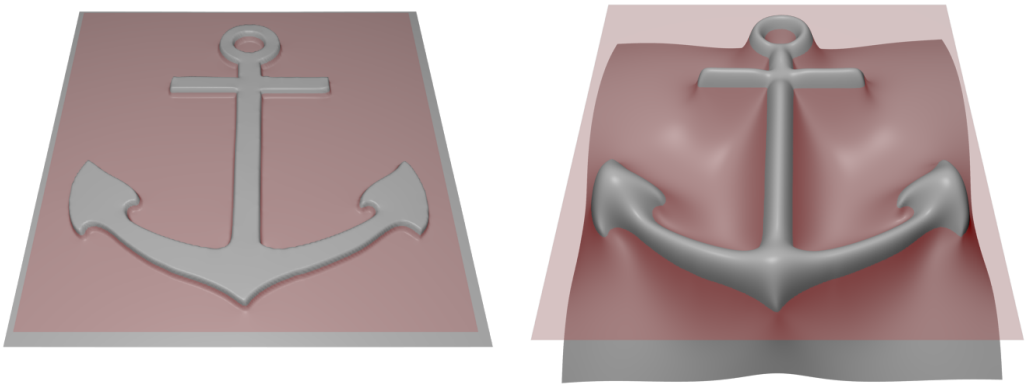


Figure 6.3: Reconstruction with our TV-IMLS **(left)** and Hessian-IMLS **(right)** approaches from two dimensional oriented points sampled from an anchor shape. The 2D implicit functions are visualised as height fields.

the normal constraint from the first term in (6.23) in contrast to our second order model (6.24). We only require one data term that has the benefit of allowing to smooth out smaller measurement errors by considering local information with a small $\sigma > 0$. Additionally this term is easier to discretise as will be seen in Section 6.7. We do not require a hybrid finite element / finite difference discretisation as in [21], and our data term only contributes to the diagonal entries of the discrete system matrix. Furthermore, the use of weighting functions offers a natural way of treating varying patch sizes. They can arise e.g. when computing oriented points with the patch based multi-view stereo approach of Furukawa and Ponce [40].

6.5.2 TV-IMLS and HOM-IMLS

Instead of approximating the signed distance function, several other approaches approximate the indicator function of the unknown surface. This poses an interesting alternative. To achieve this, we reconsider the basic idea of understanding oriented points as local approximations within some small neighbourhood. As previously discussed, the oriented points allow us to construct a local approximation of the signed distance function

$$f_i(\boldsymbol{x}) = \langle \boldsymbol{x} - \boldsymbol{p}_i, \boldsymbol{n}_i \rangle. \tag{6.29}$$

However, this also allows to locally approximate the indicator function around $\boldsymbol{p}_i$ by $\Theta(f_i(\boldsymbol{x}))$, where Θ is a continuous approximation of the Heaviside func-

tion. Thus, we propose the data fidelity term

$$D_\Theta(u) = \sum_{i=1}^{N} \int_{B_\sigma(\boldsymbol{p}_i)} \Big(u(\boldsymbol{x}) - \Theta\big(f_i(\boldsymbol{x})\big)\Big)^2 \,\mathrm{d}\boldsymbol{x} \tag{6.30}$$

that rewards a close fit to the given local approximations. Also here, one can replace the hard window $B_\sigma(\boldsymbol{p}_i)$ by a Gaussian weighting function for example. A simple way to obtain a continuous approximation Θ is given by convolving the Heaviside function with a suitable kernel. If one uses a box function as convolution kernel, the resulting Θ can as well be interpreted as a scaled and truncated signed distance function as it is used in [29].

The indicator function is fundamentally different from the signed distance function as displayed in a simplified 2D example in Figure 6.3. This has to be considered in the choice of the smoothness term. As the gradient of the indicator function is zero almost everywhere, we propose to minimise a suitable energy with a first order smoothness term:

$$E_\Theta(u) = D_\Theta(u) + \alpha \int_\Omega \Psi(|\boldsymbol{\nabla} u(\boldsymbol{x})|^2) \,\mathrm{d}\boldsymbol{x}, \tag{6.31}$$

where the parameter $\alpha > 0$ controls the degree of smoothness. This alternative is interesting as it only requires computing first order derivatives.

More specifically, we will consider two choices of Ψ, namely $\Psi(s^2) = s^2$ and $\Psi(s^2) = \sqrt{s^2 + \epsilon^2}$ with a small constant $\epsilon > 0$. We refer to these models as HOM-IMLS and TV-IMLS, respectively. The former choice will lead to a penalisation of the squared gradient magnitude. The latter choice is motivated by the fact that it yields total variation regularisation (TV) for $\epsilon = 0$, which is well-suited for the reconstruction of piecewise constant functions such as an indicator function. Furthermore, it is known that TV penalises the perimeter of the level sets, which in this case corresponds to surface area. This is a favourable property, which usually leads to removal of small isolated clutter and reconstructions of low genus. The minimiser of Equation 6.31 must necessarily fulfil the corresponding Euler-Lagrange equation

$$u \sum_{i=1}^{N} w_i \;-\; \alpha \operatorname{div}\big(\Psi'(|\boldsymbol{\nabla} u|^2)\boldsymbol{\nabla} u\big) = \sum_{i=1}^{N} w_i \,\Theta(f_i), \tag{6.32}$$

with natural boundary condition $\langle \boldsymbol{n}, \boldsymbol{\nabla} u \rangle = 0$.

6.6 Surfaces with Texture

If a set of oriented points is provided along with colour information

$$\left\{(\boldsymbol{p}_i, \boldsymbol{n}_i, \boldsymbol{c}_i) \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \,\middle|\, i = 1, \ldots, N\right\}, \tag{6.33}$$

then it is straightforward to obtain textured reconstructions. After reconstructing a surface from the oriented points, one can simply follow the previous strategy and compute a unified colour map $\boldsymbol{g} : \Omega \to \mathbb{R}^3$ as the minimiser of

$$E(\boldsymbol{g}) = \sum_{i=1}^{N} \int_{\Omega} w_i |\boldsymbol{g} - \boldsymbol{c}_i|^2 \, \mathrm{d}\boldsymbol{x} \;+\; \alpha \int_{\Omega} |\boldsymbol{J}\boldsymbol{g}|_F^2 \, \mathrm{d}\boldsymbol{x}. \tag{6.34}$$

Here $\boldsymbol{J}$ is the Jacobi Matrix. This generalises the approach of Calakli and Taubin [22] who used the Dirac distribution as weighting function. Although this constitutes a simple way of obtaining reconstructions with texture, it might be advisable to use recorded colour images directly if they are available.

6.7 GPU Implementation

Our GPU implementation uses the NVIDIA CUDA framework. It can essentially be divided into two stages of computation. First, setting up coefficients and right hand side for a system of equations and subsequently solving it. Let us now first discuss required discretisations before describing how both of these steps can efficiently be computed on parallel graphics hardware.

Either we set the domain Ω as a rectangular axis aligned bounding box $[a_1, b_1] \times [a_2, b_2] \times [a_3, b_3] \subset \mathbb{R}^3$ that contains all oriented points or we manually specify it. The domain Ω is then discretised by choosing $(m_1, m_2, m_3)^\top$ equidistant samples in each direction resulting in $M = m_1 \cdot m_2 \cdot m_3$ unknowns.

Minimising Hessian-IMLS

Let us denote by $\boldsymbol{u}$ and $\boldsymbol{f}_i \in \mathbb{R}^M$ the discrete versions of $u(\boldsymbol{x})$ and $f_i(\boldsymbol{x})$. We have rearranged the unknowns in a vector using column major ordering. Let $\boldsymbol{W}_i$ be a diagonal matrix that contains all weights $w_i(\boldsymbol{x})$ and $V := \{x, y, z\}$. Then we can discretise Equation 6.24 as follows:

$$E(\boldsymbol{u}) = \sum_{i=1}^{N} |\boldsymbol{W}_i^{\frac{1}{2}}(\boldsymbol{u} - \boldsymbol{f}_i)|^2 + \alpha \sum_{j=1}^{M} \sum_{\boldsymbol{\gamma} \in V^2} (\boldsymbol{D}_{\boldsymbol{\gamma}} \boldsymbol{u})_j^2 \,, \tag{6.35}$$

where $\boldsymbol{D}_{\boldsymbol{\gamma}}$ realises the corresponding second order derivative at each location where it can be estimated using a given stencil. The necessary and sufficient condition for a minimiser can be obtained in this case by setting the gradient to zero:

$$\left(\sum_{i=1}^{N} \boldsymbol{W}_i + \alpha \sum_{\boldsymbol{\gamma} \in V^2} \boldsymbol{D}_{\boldsymbol{\gamma}}^{\top} \boldsymbol{D}_{\boldsymbol{\gamma}} \right) \boldsymbol{u} = \sum_{i=1}^{N} \boldsymbol{W}_i \boldsymbol{f}_i. \tag{6.36}$$

Compared to the smooth signed distance surface reconstruction [21], we only require one data term instead of two. Moreover, we do not require a hybrid finite element / finite difference discretisation. This is because we effectively enforce similarity to a small oriented patch instead of a pointwise function value and normal constraint. Therefore, the data term only contributes to the diagonal of the system matrix and not to any off-diagonal entries as in [21].

Discretisation. Let us now cover the discretisation in more detail. Specifically we will point out our choice of the different $\boldsymbol{D}_{\boldsymbol{\gamma}}$. To this end, it is more convenient to use the spatial coordinates (i, j, k) instead of considering $\boldsymbol{u}$ to be a column vector with a single index. This is because the spatial coordinates better express that $\boldsymbol{u}$ is a discrete version of a 3D function. Remember that we have rearranged the unknowns in $\boldsymbol{u}$ by first sampling the 3D function $u(\boldsymbol{x})$ and then using column major ordering to obtain the column vector $\boldsymbol{u}$. Therefore, there is a bijective mapping between the single index and the spatial coordinates such that it is straightforward to go back and forth between both different representations. For the second derivatives w.r.t. x, we use the discretisation

$$[\boldsymbol{D}_{xx}\boldsymbol{u}]_{i,j,k} = \frac{u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}}{h_x^2} \,. \tag{6.37}$$

The corresponding contribution $\boldsymbol{D}_{xx}^{\top}\boldsymbol{D}_{xx}$ to the discrete gradient from Equation 6.36 can be represented by the weights in the following stencil:

$$\frac{1}{h_x^4} \cdot \begin{array}{|c|c|c|c|c|} \hline \mathcal{X}_{i-1} & \begin{array}{c} -2\,\mathcal{X}_i \\ -2\,\mathcal{X}_{i-1} \end{array} & \begin{array}{c} \mathcal{X}_{i+1} \\ +4\,\mathcal{X}_i \\ +\mathcal{X}_{i-1} \end{array} & \begin{array}{c} -2\,\mathcal{X}_i \\ -2\,\mathcal{X}_{i+1} \end{array} & \mathcal{X}_{i+1} \\ \hline \end{array} \tag{6.38}$$

with the function $\mathcal{X}$ defined as

$$\mathcal{X}_i = \begin{cases} 1, & 1 < i < m_1 \\ 0, & \text{else.} \end{cases} \tag{6.39}$$

It assumes that we use the indexing $1, \ldots, m_1$ for the unknowns in x-direction and ensures correct stencil weights also at the boundaries. The discretisation of the second derivatives w.r.t. y and z is chosen analogously. Therefore, the corresponding stencils can be represented in a similar way using the functions $\mathcal{Y}$ and $\mathcal{Z}$ defined as

$$\mathcal{Y}_j = \begin{cases} 1, & 1 < j < m_2 \\ 0, & \text{else} \end{cases} \quad \text{and} \quad \mathcal{Z}_k = \begin{cases} 1, & 1 < k < m_3 \\ 0, & \text{else} \end{cases} \tag{6.40}$$

to ensure correct treatment of the boundaries. For the mixed derivatives, we choose the discretisation

$$[\boldsymbol{D}_{xy}\boldsymbol{u}]_{i,j,k} = \frac{u_{i+1,j+1,k} - u_{i+1,j-1,k} - u_{i-1,j+1,k} + u_{i-1,j-1,k}}{4\, h_x\, h_y}. \tag{6.41}$$

Then the corresponding contribution $\boldsymbol{D}_{xy}^\top \boldsymbol{D}_{xy}$ to the discrete gradient from Equation 6.36 can be represented by the weights in the following stencil:

$$\frac{1}{16\, h_x^2\, h_y^2} \cdot \begin{array}{|c|c|c|c|c|} \hline \mathcal{X}_{i-1}\mathcal{Y}_{j+1} & 0 & \begin{array}{c} -\mathcal{X}_{i+1}\mathcal{Y}_{j+1} \\ -\mathcal{X}_{i-1}\mathcal{Y}_{j+1} \end{array} & 0 & \mathcal{X}_{i+1}\mathcal{Y}_{j+1} \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \begin{array}{c} -\mathcal{X}_{i-1}\mathcal{Y}_{j+1} \\ -\mathcal{X}_{i-1}\mathcal{Y}_{j-1} \end{array} & 0 & \begin{array}{c} \mathcal{X}_{i+1}\mathcal{Y}_{j+1} + \\ \mathcal{X}_{i+1}\mathcal{Y}_{j-1} + \\ \mathcal{X}_{i-1}\mathcal{Y}_{j+1} + \\ \mathcal{X}_{i-1}\mathcal{Y}_{j-1} \end{array} & 0 & \begin{array}{c} -\mathcal{X}_{i+1}\mathcal{Y}_{j+1} \\ -\mathcal{X}_{i+1}\mathcal{Y}_{j-1} \end{array} \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \mathcal{X}_{i-1}\mathcal{Y}_{j-1} & 0 & \begin{array}{c} -\mathcal{X}_{i+1}\mathcal{Y}_{j-1} \\ -\mathcal{X}_{i-1}\mathcal{Y}_{j-1} \end{array} & 0 & \mathcal{X}_{i+1}\mathcal{Y}_{j-1} \\ \hline \end{array}$$

The other mixed derivatives $\boldsymbol{D}_{xz}$ and $\boldsymbol{D}_{yz}$ are chosen analogously. When assembling all the contributions of the six different stencils by adding up all entries corresponding to the same cell, one obtains a $(5 \times 5 \times 5)$-stencil with a sparsity pattern as shown in Figure 6.4. Note that the functions $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ can efficiently be computed on the fly on the GPU and do not have to be precomputed and stored.

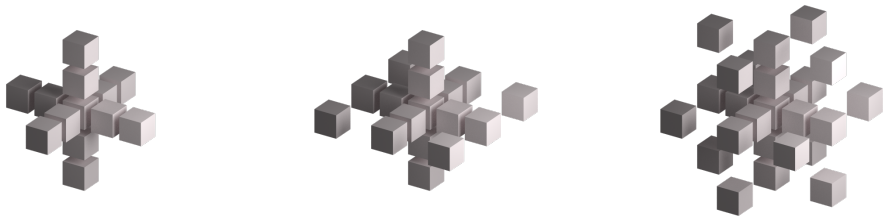


Figure 6.4: Sparsity pattern of the $(5 \times 5 \times 5)$-stencil of the discretisation arising from penalising the Frobenius norm of the Hessian. Locations with a nonzero entry are depicted by a grey box. **Left:** Nonzero entries arising from the terms u_{xx}^2, u_{yy}^2, and u_{zz}^2. **Middle:** Nonzero entries when considering the previous terms and u_{xz}^2. **Right:** Nonzero entries when considering all terms. Although there are only 25 nonzero entries out of the 125 locations in the $(5 \times 5 \times 5)$-neighbourhood, the stencil is still irreducible.

Minimising TV-IMLS and HOM-IMLS

Similar as in the previous paragraph, we discretise (6.31) as follows:

$$E(\boldsymbol{u}) = \sum_{i=1}^{N} |\boldsymbol{W}_i^{\frac{1}{2}}(\boldsymbol{u} - \boldsymbol{\Theta}(\boldsymbol{f}_i))|^2 + \alpha \sum_{j=1}^{M} \Psi\left(\sum_{\gamma \in V} (\boldsymbol{D}_\gamma \boldsymbol{u})_j^2\right). \tag{6.42}$$

Here $\Theta(\boldsymbol{f}_i) \in \mathbb{R}^M$ is the discrete version of $\Theta(f_i(\boldsymbol{x}))$, and $\boldsymbol{D}_\gamma$ realises forward differences in direction of γ for each location where forward differences can be evaluated. By computing the gradient, we obtain the necessary condition a minimiser u must fulfil:

$$\left(\sum_{i=1}^{N} \boldsymbol{W}_i + \alpha \sum_{\gamma \in V} \boldsymbol{D}_\gamma^\top \boldsymbol{\Phi}(\boldsymbol{u}) \boldsymbol{D}_\gamma\right) \boldsymbol{u} = \sum_{i=1}^{N} \boldsymbol{W}_i \, \boldsymbol{\Theta}(\boldsymbol{f}_i), \tag{6.43}$$

where $\boldsymbol{\Phi}(\boldsymbol{u})$ is a diagonal matrix with

$$(\boldsymbol{\Phi}(\boldsymbol{u}))_{jj} = \Psi'\left(\sum_{\gamma \in V} (\boldsymbol{D}_\gamma \boldsymbol{u})_j^2\right). \tag{6.44}$$

Since the energy is strictly convex, the minimiser is unique and the necessary condition is also sufficient. More compactly, we express the nonlinear system as

$$(\boldsymbol{P} + \alpha \, \boldsymbol{A}(\boldsymbol{u})) \, \boldsymbol{u} = \boldsymbol{q}, \tag{6.45}$$

with the abbreviations

$$\boldsymbol{P} = \sum_{i=1}^{N} \boldsymbol{W}_i, \quad \boldsymbol{A}(\boldsymbol{u}) = \sum_{\gamma \in V} \boldsymbol{D}_\gamma^\top \boldsymbol{\Phi}(\boldsymbol{u}) \boldsymbol{D}_\gamma, \quad \boldsymbol{q} = \sum_{i=1}^{N} \boldsymbol{W}_i \boldsymbol{\Theta}(\boldsymbol{f}_i). \tag{6.46}$$

The nonlinear system of equations can be solved by the fixed point iteration

$$\left(\boldsymbol{P} + \alpha\, \boldsymbol{A}(\boldsymbol{u}^{\boldsymbol{k}})\right)\, \boldsymbol{u}^{k+1} = \, \boldsymbol{q} \quad (k \geq 0), \tag{6.47}$$

i.e. by solving a sequence of linear systems. In the case of HOM-IMLS, the squared gradient magnitude is penalised, and the resulting system of equations is linear.

Discretisation. Let us now cover the discretisation in more detail. As when discussing the Hessian-IMLS approach, we switch to the spatial location (i, j, k) at this point instead of the single index notation where $\boldsymbol{u}$ is considered to be a column vector. We use forward differences such that

$$[\boldsymbol{D}_x \boldsymbol{u}]_{i,j,k} = \frac{u_{i+1,j,k} - u_{i,j,k}}{h_x}. \tag{6.48}$$

Accordingly, the coefficients of the matrix $\boldsymbol{D}_x^\top \boldsymbol{\Phi}(\boldsymbol{u}) \boldsymbol{D}_x$ can be described by the weights of the stencil

$$\frac{1}{h_x^2} \cdot \begin{array}{|c|c|c|} \hline -\Psi'_{i-1,j,k} & \Psi'_{i-1,j,k} + \Psi'_{i,j,k} & -\Psi'_{i,j,k} \\ \hline \end{array}, \tag{6.49}$$

where

$$\Psi'_{i,j,k} := \Psi' \left(\sum_{\gamma \in V} (\boldsymbol{D}_\gamma \boldsymbol{u})_{i,j,k}^2 \right). \tag{6.50}$$

The contributions of the other terms $\boldsymbol{D}_y^\top \boldsymbol{\Phi}(\boldsymbol{u}) \boldsymbol{D}_y$ and $\boldsymbol{D}_z^\top \boldsymbol{\Phi}(\boldsymbol{u}) \boldsymbol{D}_z$ are found in an analogous way. By adding up all entries corresponding to the same cell, one obtains the complete $(3 \times 3 \times 3)$-stencil for $\boldsymbol{A}(\boldsymbol{u})$ which contains seven nonzero entries. We have also tested other common discretisations for the PDE (6.32). However, we could not find a noticeable improvement in this context.

Implementation Details

For setting up the system matrix and the right hand side, we carry out the summations required to compute $\boldsymbol{P}$ and $\boldsymbol{q}$ in parallel using atomic operations.

Due to the fact that we use a small constant σ this operation is extremely fast. For solving the linear system of equations that either arise directly or within the fixed point iteration for solving a nonlinear system, we use a cascadic version of the Fast Jacobi (FJ) solver [10] explained in Chapter 2. The FJ solver is essentially a modified Jacobi over-relaxation (JOR) method, where the relaxation parameter is not fixed but varied in a cyclic way. Due to this, FJ is much more efficient than JOR but still as simple to implement; see Section 2.4.2. In particular, it is perfectly suited for parallelisation as it merely requires knowing values from the last iteration to compute the new iterations result. We use 3D CUDA arrays bound to textures or surfaces, which is well-suited in this scenario. It allows for fast read and write operations required in each iteration and makes use of efficient 3D caching. In the linear case, $4 \cdot M^3$ variables have to be stored. This corresponds to a memory usage of 2 GiB when using a volumetric grid of 512^3 voxels and 32-bit floating point accuracy. For the nonlinear case, the nonlinearities $\boldsymbol{\Phi}(\boldsymbol{u}^k)$ are stored as well. The reconstruction with 400^3 voxels shown in Figure 6.5 took 7.4 seconds on a GeForce GTX690. This illustrates the good performance of FJ on modern GPUs.

6.8 Experimental Results

For the examples taken from the Stanford scanning repository [103], raw data in range grid format is available. We estimate normals using the neighbouring pixels within each range scan. We have used cubic voxels and Gaussian weighting functions with a constant σ equal to the voxel size for each input point in all experiments.

Dragon

In Figure 6.5, we have reconstructed the Dragon model from the Stanford scanning repository [103] using our HOM-IMLS approach. One can see that choosing a too small smoothness weight ($\alpha = 0.1$) results in isolated clutter. This is reasonable because a smoothness weight of zero simply yields an implicit moving least squares solution which has been shown to produce reconstructions with spurious artefacts in many cases [58, 21]. By selecting a suitable smoothness weight ($\alpha = 10$), it is possible to remove the clutter while preserving the details even with the quadratic first order model. This illustrates the benefit of combining the strengths of the IMLS approach with the advantages of regularisation.

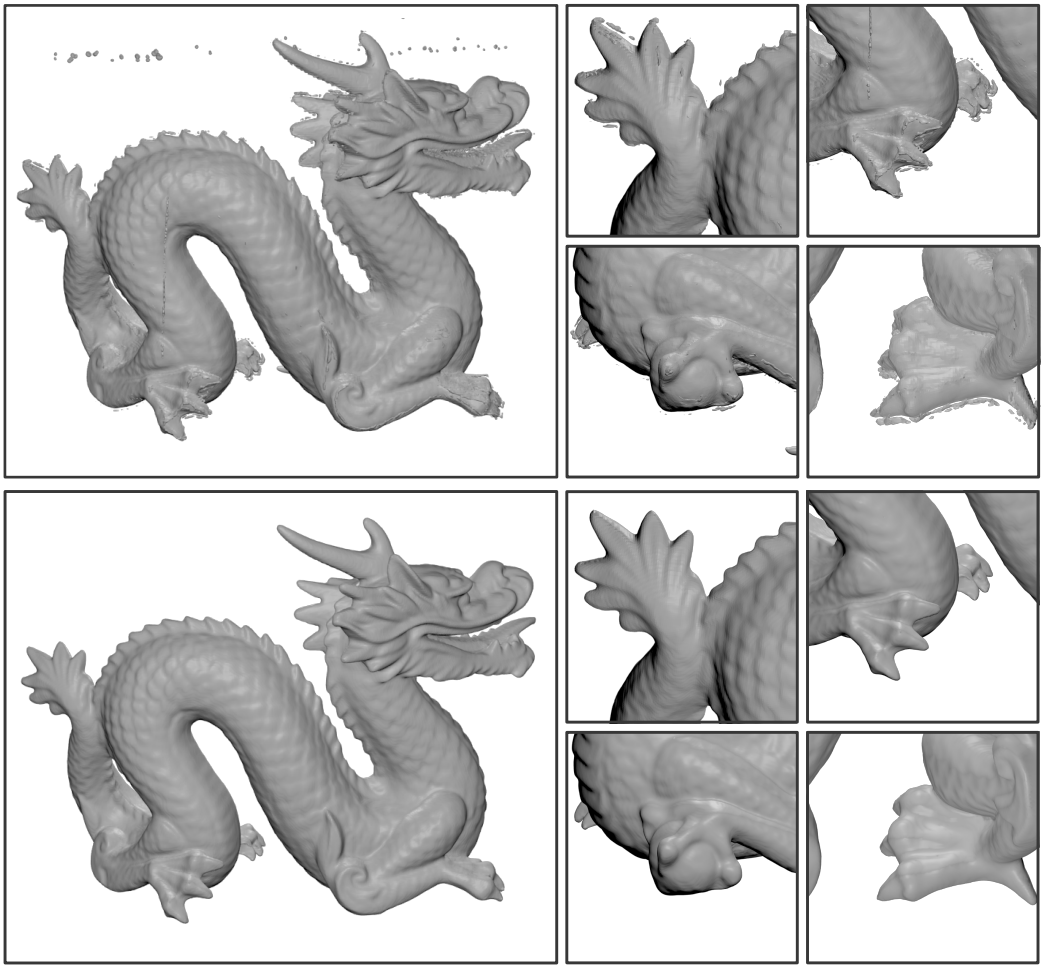


Figure 6.5: Reconstructions with our HOM-IMLS approach (6.31) at 400^3 voxels. **Top:** Choosing a smoothness weight close to zero ($\alpha = 0.1$) approximates a moving least squares solution, which results in isolated clutter. **Bottom:** By choosing a suitable smoothness weight ($\alpha = 10$), the isolated clutter is removed while small details are kept.

Drill

In Figure 6.6, we use use the drill dataset taken from the Stanford scanning repository [103] to compare wavelet surface reconstruction [74], (screened) Poisson surface reconstruction (PSR) [58], and smooth signed distance surface reconstruction (SSD) [21] to our novel approaches. We have always used the implementations provided by the respective authors and a tree depth of 9. Let us now consider the reconstructions in this order.

The wavelet approach **(b)** is very fast (0.4 seconds) and can also deal

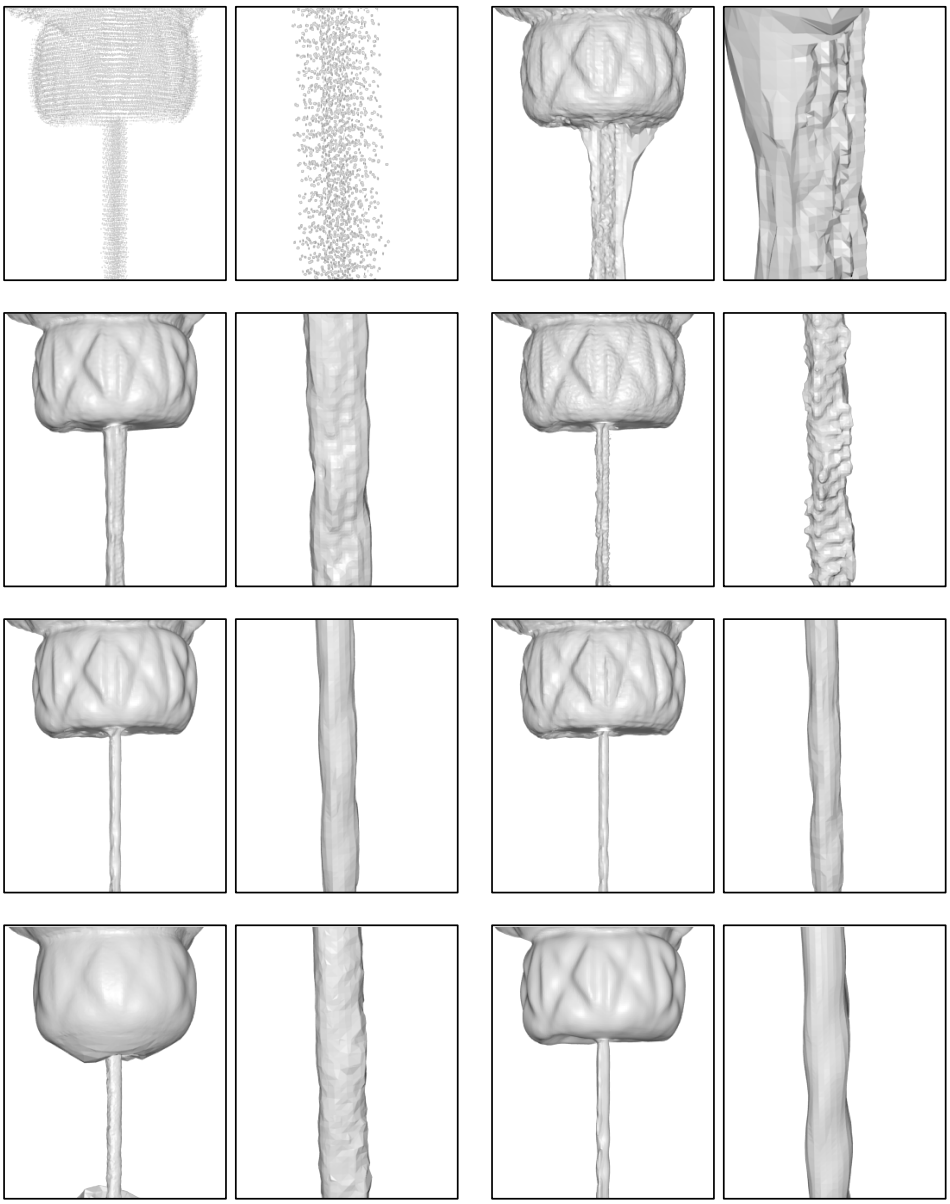


Figure 6.6: **In reading order**: **(a)** Oriented points **(b)** Wavelet surface reconstruction **(c)** Poisson surface reconstruction **(d)** Screened Poisson surface reconstruction **(e)** Our HOM-IMLS approach (6.31) **(f)** Our TV-IMLS approach (6.31) **(g)** SSD **(h)** Our Hessian-IMLS approach (6.24)

with noise and outliers to a certain extent. However, in this case it is not able to produce a faithful reconstruction. PSR **(c)** delivers a reasonable reconstruction, but the drill bit itself is rather unsmooth. Adding a screening

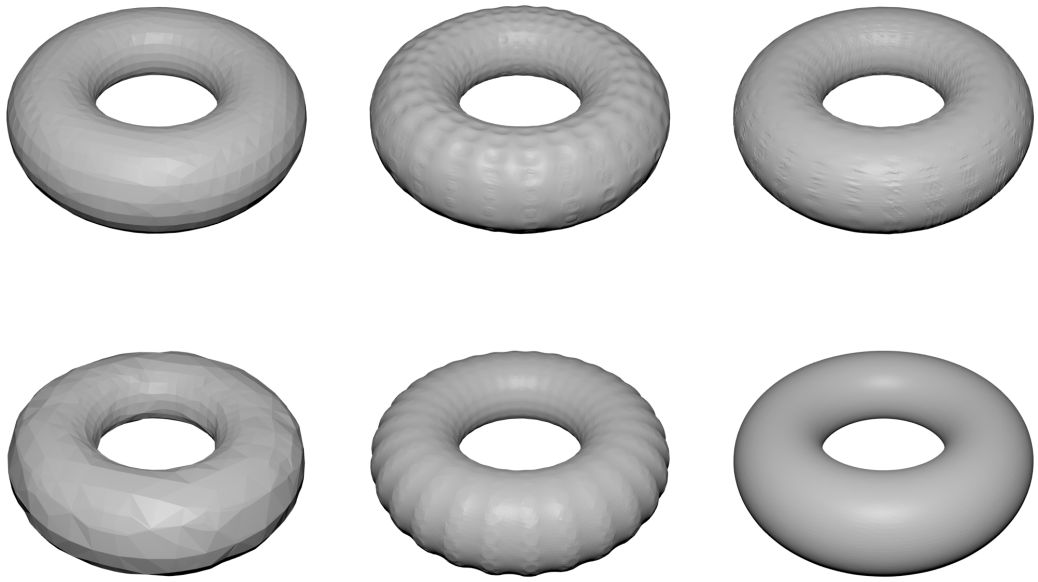


Figure 6.7: **In reading order**: **(a)** Screened PSR with adaptive octree **(b)** Screened PSR on fine regular grid **(c)** Our HOM-IMLS approach **(d)** SSD **(e)** Our TV-IMLS approach (6.31) **(f)** Our Hessian-IMLS approach (6.24)

weight ($\alpha_0 = 1$) in this case leads to overfitting noise **(d)**. In the SSD reconstruction **(g)**, the drill bit is also noisy although we have already chosen a large smoothness weight ($\alpha_2 = 25$) and one can clearly see that the top part is oversmoothed. By choosing a smaller weight for the smoothness term, it is possible to obtain a good reconstruction of the top part at the cost of more noise on the drill bit itself. All of our novel reconstructions **(e)**, **(f)** and **(h)** are able to convey the shape of the drill bit. Moreover, in the magnifications one can even slightly recognise the windings carved in towards the bottom of the drill bit. The running times of our HOM-IMLS, TV-IMLS, and Hessian-IMLS approaches are 2.9, 3.2, and 3.6 seconds, respectively. PSR requires 12.8 in the standard and 14.3 in the screened version, whereas SSD finishes in 4.8 seconds.

Torus

In Figure 6.7, we examine the different models and implementations for a simple shape defined by only few oriented points. PSR computes the solution on an octree that adapts to the input points. Doing so it is possible to

obtain a reasonable but coarse reconstruction, see Figure 6.7 **(a)**. When switching off the adaptivity of the octree using a maximal tree depth of 8, artefacts occur around the input points, cf. Figure 6.7 **(b)**. Similar but less prominent artefacts occur in our HOM-IMLS approach **(c)**. The SSD approach does not allow to switch off the adaptivity of the octree in the given implementation. Therefore, it can only compute a coarse reconstruction **(d)**. In our reconstruction with TV-IMLS **(e)**, one can see how the surface area is minimised. However, here this property is not beneficial. Our Hessian-IMLS approach **(f)** achieves a reconstruction without artefacts and is in our eyes the most promising approach. Thus, we will focus on this approach in the remainder of the experiments. The coarse reconstructions with SSD (0.3 s) and PSR (0.4 s) are quite fast. However, two minutes are required when switching off the adaptivity of the octree in PSR. Our methods HOM-IMLS, TV-IMLS, and Hessian-IMLS finish in 0.9, 1.1, and 1.4 seconds, respectively.

Reconstruction Benchmark

We use the reconstruction benchmark of Berger et al. [15] for evaluating reconstruction accuracy of our Hessian-IMLS approach compared to (screened) PSR [58] and SSD [21]. The benchmark simulates scanner error as nonuniform sampling, noise and misalignment and covers many virtual scans of five different implicit surfaces. We use the most recently available author implementations for PSR and SSD, which are Versions 5.5 and 3.0, respectively. For PSR, we select the settings recommended by the authors for this benchmark, i.e. we use a screening weight of 4 for screened PSR and the same implementation with a screening weight of 0 to compute the unscreened version [58]. For SSD we have found the weights $\boldsymbol{\alpha} = (1, 1, 1)^\top$ to produce the best error values when considering both distance and angular errors. We use a resolution of 300^3 voxels, which is appropriate for covering present details in the models and accordingly choose an octree depth of 9 for the other approaches. For our approach we select a smoothness weight $\alpha = 1$.

When comparing the errors in distance values, one can see in Figure 6.8 that all methods tend to yield lower errors compared to PSR. In general, the error values are close together, though. However, when considering the angular error values, our approach manages to also obtain the lowest error in most cases.

Globe

We have recorded several colour images of an ordinary globe and used VisualSFM [120] to estimate the camera poses. Subsequently we used the

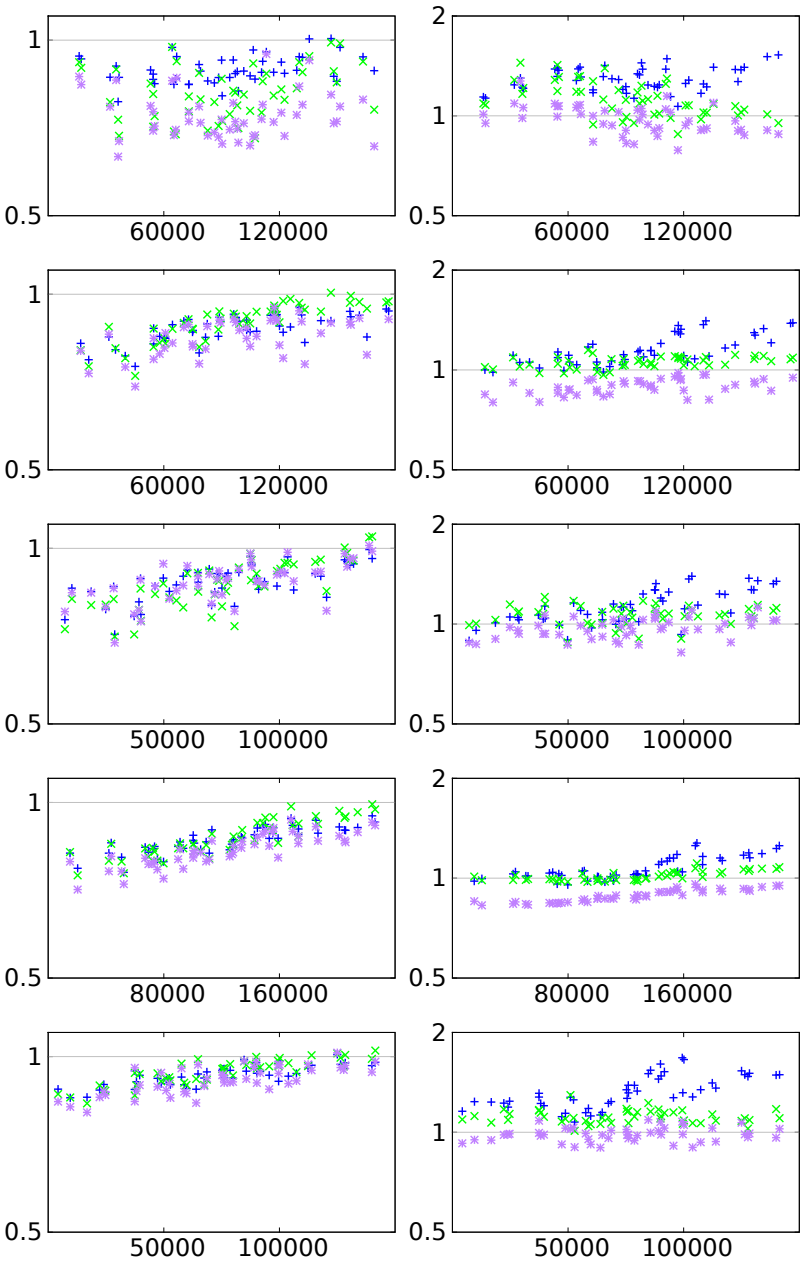


Figure 6.8: Reconstruction accuracy compared with the benchmark of Berger et al. [15]. For each of the five datasets, Anchor, Dancing, Daratech, Gargoyle and Quasimodo (rows from top to bottom), the two plots show the ratios of the mean distance (left) and mean normal errors (right) of screened PSR (blue), SSD (green) and our approach Hessian-IMLS (purple), relative to the original PSR algorithm. Each symbol corresponds to one benchmark test, where the horizontal axis denotes the amount of oriented points available in that test.

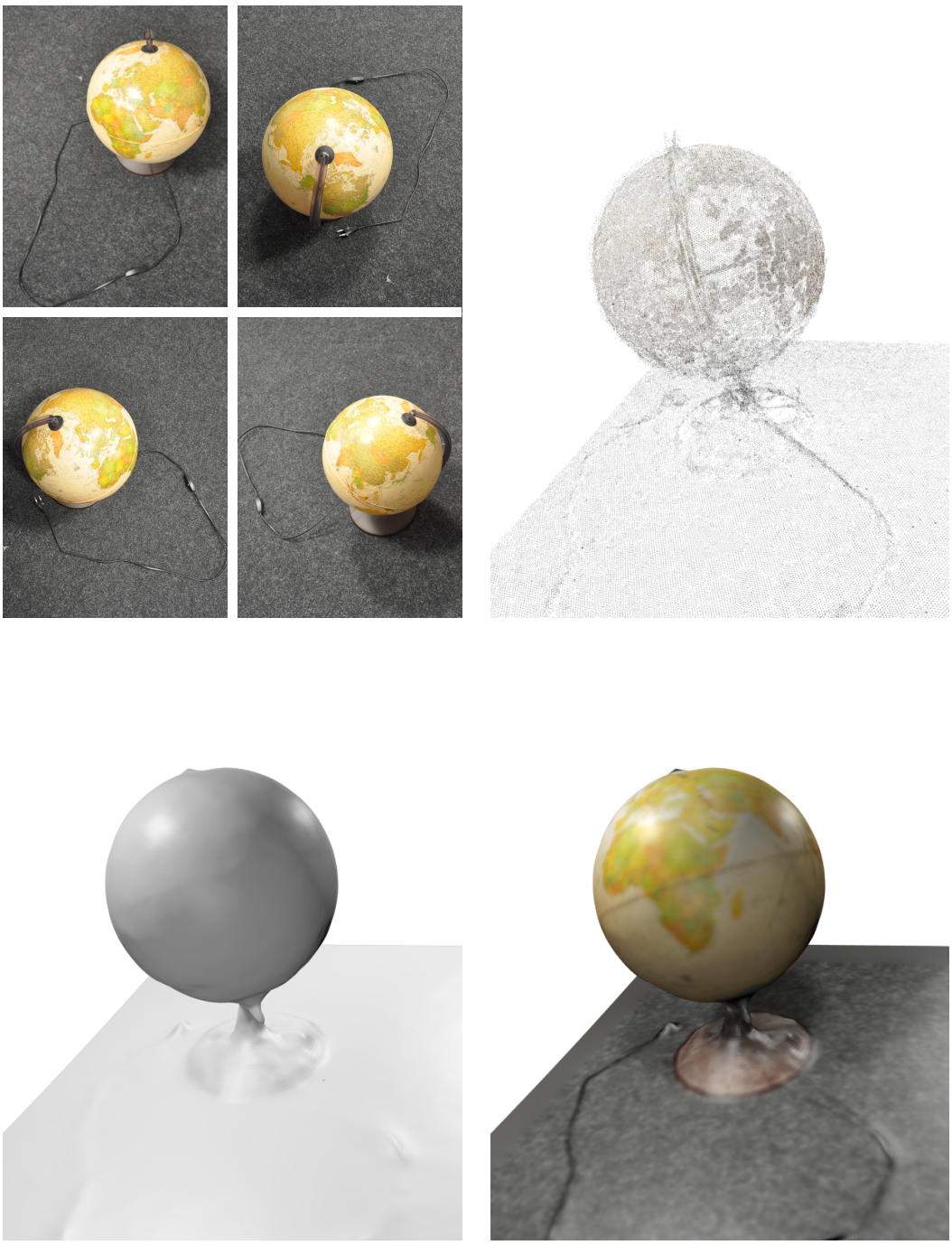


Figure 6.9: **In reading order**: **(a)** Subset of the input images **(b)** Coloured oriented point cloud produced by PMVS **(c)** Dense geometry reconstructed with our Hessian-IMLS approach (6.24) **(d)** Textured reconstruction

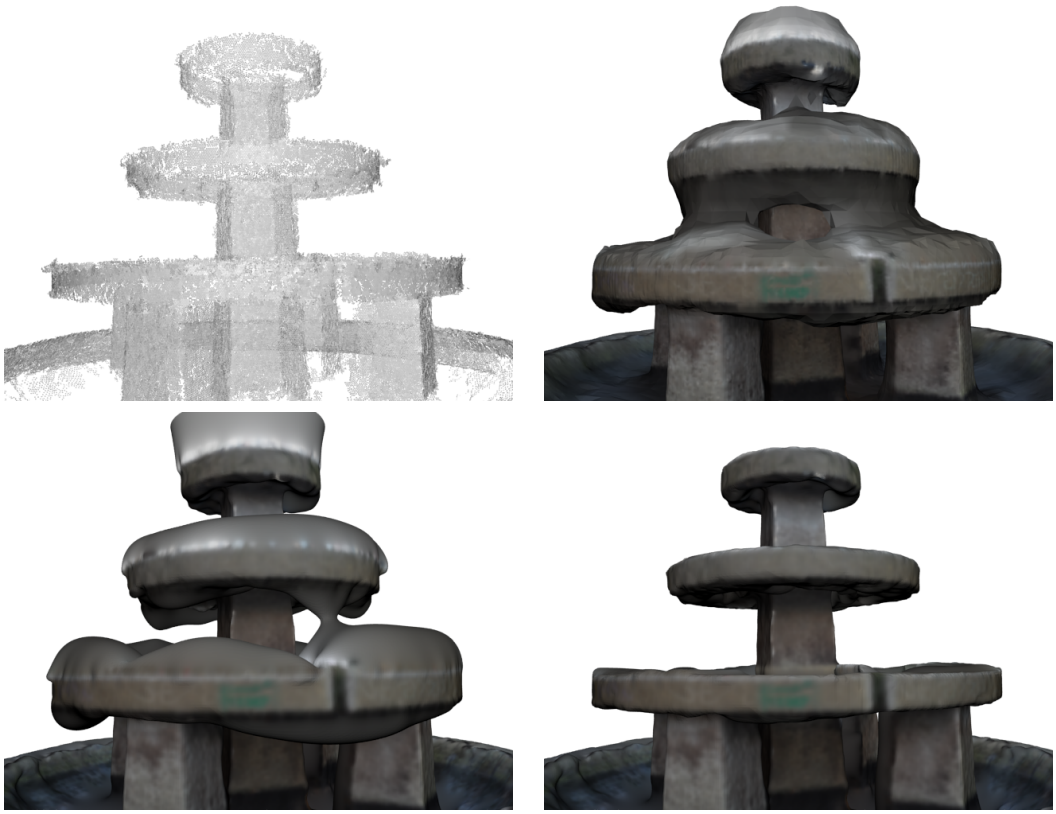


Figure 6.10: **In reading order**: **(a)** Input point cloud **(b)** SSD **(c)** Hessian-IMLS **(d)** Hessian-IMLS with hull constraint

patch-based multi-view stereo (PMVS) [40] algorithm to compute a coloured oriented point cloud. Figure 6.9 shows that our method is capable of producing a reasonable reconstruction from this point cloud despite of uncovered areas on the globe and noise. The texture has been computed by extending the idea of [22] to fit into our framework.

6.9 Limitations and Discussion

An octree as in [21, 58] allows for a better scaling in the unknowns and it is also possible to solve our model on an octree instead of a regular grid. However, in both previously mentioned implementations, the octree only adapts to the input data and not to the evolving solution, i.e. the unknown surface. Our fast GPU implementation allows to compute reconstructions of resolutions as required for the recent reconstruction benchmark of Berger et al. [15] in a competitive runtime of a couple of seconds.

We usually choose a small constant σ in order to account for the fact that an oriented point only defines a small linear surface patch locally. This is possible because the smoothness term will take care of regularising and filling in missing information. For standard implicit moving least squares approaches, the parameter σ has to be used to both regularise and fill in missing information. As previously mentioned, this can be problematic since filling large missing regions requires a large σ and thus smoothing away small scale features. When considering our Hessian-IMLS approach, the smoothness term does not penalise linear functions. It is interesting to observe that this is in accordance with the data term. To this end, we assume that all oriented points have been sampled from a planar surface. Then one can easily observe that the data term will preserve this planar surface independent of the choice of σ.

Concerning measurement errors, we basically can differentiate between two kinds of problems in the oriented point cloud: First, some oriented points may have been measured at inaccurate locations. Second, there may be large regions of the object that are not covered by oriented points at all. In the first case, regularisation can be a remedy. However, in the second case the regularisation may fill in the missing regions in an undesirable way. To avoid unwanted surface sheets, it can thus be beneficial to incorporate a further constraint that preserves free space. We explain how such a constraint can be added into the Hessian-IMLS model, which is our best performing one.

6.10 Hull Constraints

Often, it is possible to estimate a hull that should contain the object to be reconstructed. In real world scenarios where oriented points are recovered from images, a prominent example of such a hull is the visual hull [67]. Thus, we propose to augment our Hessian-IMLS formulation with an additional term that allows to encourage the surface to stay within a specified hull. This is especially helpful to steer the surface reconstruction when larger parts of the object are not sufficiently covered with oriented points.

Let $\mathcal{H}$ denote the set of all points within a specified hull. Then we know that for any point $\boldsymbol{x}$ outside $\mathcal{H}$, its distance value $u(\boldsymbol{x})$ should be larger or equal to the Euclidean distance from the hull $d(\mathcal{H}, \boldsymbol{x})$. However, inside the hull all values should be allowed without further penalisation. Thus, we propose to add the following term to our model (6.24):

$$\text{Hull}(u) = \beta \int_{\Omega \setminus \mathcal{H}} \max\{0, d(\mathcal{H}, \boldsymbol{x}) - u(\boldsymbol{x})\}^2 \mathrm{d}\boldsymbol{x}. \tag{6.51}$$

This hull constraint can be regarded as an optional additional level of control. Its importance can be specified by choosing a suitable $\beta > 0$ and it can be switched off by setting $\beta = 0$ if a hull should not be used. In the presence of a hull constraint, we need to consider the gradient of 6.51 additionally. The j-th component of it can be written as

$$2\,\beta\,H(d_j)\,H(d_j - u_j)\,(u_j - d_j), \tag{6.52}$$

where H denotes the Heaviside function and $\boldsymbol{d}$ the discrete version of $d(\mathcal{H}, \boldsymbol{x})$. Although the Heaviside function itself is discontinuous, the above expression is continuous.

With the same pipeline as in the globe experiment, we have computed an oriented point cloud of a fountain, see Figure 6.10 **(a)**. In this case, the lack of oriented points in some locations, for example at the very top, causes unwanted filling-in effects as in **(b)** and **(c)**. Our hull constraint according to Equation 6.51 allows for a better control of the surface **(d)**. In this case, we used a visual hull estimated from silhouettes made with the approach in [47].

As we have stated in [7], incorporating a hull constraint is also possible for the first order approaches. It can be realised by adding a term to the energy that prefers a suitable constant value outside the given hull. Such an approach has simultaneously been published by Shan et al. [97] who integrate such a constraint into the Poisson surface reconstruction method.

6.11 Summary

We have proposed a general higher order framework for the implicit reconstruction of watertight surfaces from a finite set of oriented points and showed the benefits of this systematisation: It makes specific features of popular existing approaches explicit. Moreover, it helps to identify gaps within the systematisation allowing to derive hitherto unexplored approaches. While all these approaches can yield competitive results, one of them showed to be especially promising. Our Hessian-IMLS formulation combines the benefits of implicit moving least squares based approaches and thin plate spline regularisation. In difficult real world scenarios, unwanted filling-in effects that produce surfaces in regions that should be unoccupied can frequently appear for all approaches. To deal with such effects, we have proposed to incorporate a hull constraint. We implemented our framework on the GPU using a novel cyclic scheme named Fast Jacobi for solving the resulting systems of equations.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we have considered the problem of reconstructing the surface of a static object or scene using variational methods. Typically, such a task is split into a number of independent steps which altogether can be referred to as 3D reconstruction pipeline. We have investigated and contributed to several important steps that occur within popular 3D reconstruction pipelines and presented a novel pipeline that consistently relies on dense methods.

In Chapter 3, we analysed variational methods that estimate a single depth map from multiple views. While previous work is mostly formulated in terms of the unknown depth, we generalised it by introducing a parameterisation of depth. This allowed us to find that parameterisations along the line of sight are not suitable for such a scenario and that parameterisations along the optical axis are much more reasonable. For them, we presented a detailed analysis of a direct depth and an inverse depth parameterisation and are able to theoretically and practically point out several advantages of an inverse depth parameterisation. Concerning smoothness assumptions, it is compatible with second order regularisation because piecewise affine inverse depth leads to piecewise planar 3D surfaces. On the contrary, this is not the case for a direct depth parameterisation. It introduces a bias which we quantify both theoretically by means of the shape operator as well as by experiments. Furthermore, we were able to show that an inverse depth parameterisation is not only advantageous for the smoothness term. It is also preferable for the linearisation required in the data term. This is due to the fact that for common camera setups, the inverse depth parameterisation does not increase the deviation from linearity of the data term. Based on our findings, we thus recommend the inverse depth parameterisation along

the optical axis as the parameterisation of choice for variational multi-view depth estimation.

Having computed depth maps, Chapter 4 dealt with the problem of merging multiple depth maps into a single watertight 3D model. However, it is important to note that it does not matter how the depth maps have been acquired at this stage. They can be computed from multiple colour images as in a multi-view stereo setting but they can also be directly captured with active sensors. We based our analysis on the variational range image integration method of Zach et al. [124] and were able to improve the reconstruction quality by extending it in several aspects. On the one hand, we have employed an anisotropic regulariser that outperforms the existing isotropic one because it can produce much smoother surfaces while preserving ridges and corners. On the other hand we use the Euclidean signed distance instead of the distance along the line of sight when converting the depth maps into signed distance fields. Furthermore, we have presented a parallel GPU implementation using a nonstandard discretisation and the Fast Jacobi algorithm that allows for competitive runtimes.

In Chapter 5, we have relaxed the requirement that the camera poses have to be known in advance and have presented a novel 3D reconstruction pipeline that solely relies on dense methods. It only requires an image sequence of a static scene captured while following a continuous path along with the intrinsic camera parameters. We first estimate optical flow and stereo geometry in a joint variational approach for each consecutive image pair. Then we concatenate the flow fields and perform a global refinement via bundle adjustment. This yields globally consistent camera poses which allows to evaluate depth maps that can be merged as previously described. Our optical flow based constraints prove to be sufficiently robust such that the bundle adjustment can recover an accurate global model. Furthermore, our comparisons to a popular approach working with sparse features show that dense methods can be an interesting alternative also in this scenario.

When building a 3D reconstruction pipeline that relies on sparse features, this usually results in an oriented point cloud instead of a mesh. Thus, this point cloud has to be converted into a mesh in the last processing step of the pipeline. To this end, we have proposed a general higher order framework for the implicit reconstruction of watertight surfaces from a finite set of oriented points in Chapter 6. We were able to show the benefits of such a systematisation: It made specific features of popular existing approaches explicit. Furthermore, it allowed to identify gaps within the systematisation allowing to derive several new formulations. One of them combines the benefits of implicit moving least squares based approaches and thin plate spline

regularisation. This approach, which we refer to as Hessian-IMLS, was able to outperform very popular and widely used state-of-the-art methods on a publicly available benchmark. Last but not least, we have proposed to incorporate a hull constraint when computing a mesh from an oriented point cloud. This allows to avoid unwanted filling in effects that produce surfaces in regions that should be free space. We have implemented our framework on the GPU using a novel cyclic scheme named Fast Jacobi for solving the resulting systems of equations.

In summary, this thesis has contributed to several important steps that occur within common 3D reconstruction pipelines. Since these pipelines are designed in such a way that each step can be processed sequentially, the methods presented are effectively individual approaches that are able to reconstruct surfaces from different input data such as colour images, depth maps, or oriented points. Each method relies on a variational formulation which allows for a clean and transparent modelling without any hidden assumptions. The recently introduced Fast Jacobi solver is employed in the minimisation of all models presented in this thesis and thus proves its versatility. It is perfectly suited for parallelisation and leads to highly efficient GPU implementations. Furthermore, we have seen that anisotropic ideas can allow to improve the reconstruction quality by incorporating directional information also in the case of 3D reconstruction. Last but not least, we show that dense approaches can offer an interesting alternative to sparse ones also in the setting of 3D reconstruction.

7.2 Future Work

Although we were able to provide a number of advances and insights within this thesis, capturing accurate 3D models of real world objects or scenes still remains a challenging task. Thus, we will discuss some possibilities for future work in the following.

In Chapter 3, we have shown that an inverse depth parameterisation is preferable compared to a direct parameterisation of depth. To demonstrate this, we have used very basic assumptions in both the data term and the smoothness term. Thus, in future work, reconstruction results can be improved by using more sophisticated photoconsistency measures than a simple brightness constancy assumption in the data term. Also the smoothness term can benefit from more sophisticated regularisation strategies that incorporate directional information or non-local ideas. Furthermore, the estimation of a depth map from multiple views usually assumes that the camera poses are

known in advance such that they have to be computed in a preprocessing step if they are not captured with a calibrated camera setup. Therefore, it could be rewarding to analyse the challenges and benefits of strategies that jointly estimate the camera poses and the depth map.

The range image integration approach presented in Chapter 4 scales very well in terms of runtime with the number of input images and allows for detailed and accurate reconstructions. However, if a very high resolution is needed to capture very large objects or scenes, both memory requirements and runtime can become prohibitive for practical purposes. To this end, it could be very beneficial to investigate discretisations of the anisotropic regularisation term on octrees or general unstructured grids. This can allow to compute implicit functions that only have a very coarse resolution away from the surface but are highly resolved in regions close to the surface. Most probably, this will allow to compute reconstructions of a given quality with a greatly reduced memory footprint and computation time compared to using a Cartesian grid. The usage of multiple GPUs can be another more hardware demanding remedy. Computer architectures that support multiple GPUs have become quite popular and also the speed of data transfer between GPUs increases steadily. Therefore, algorithms that are parallelised across many GPUs also become more attractive. In this case, it would be interesting to analyse the data transfer required between GPUs as well as the runtime for different decomposition strategies of the domain. Also in the setting of surface reconstruction from oriented points, unstructured grids and multiple GPUs are interesting to consider.

The dense pipeline for 3D reconstruction from image sequences presented in Chapter 5 proves to be an interesting alternative to sparse approaches and the jointly estimated flows and camera geometries yield a sufficiently good initialisation for the bundle adjustment step. Future work could investigate to what extent incremental ideas as well as spatio-temporal regularisation can increase the robustness and accuracy of the results.

On a more general scale, it would be interesting to transfer the novel ideas and insights from the context of 3D reconstruction from static scenes to a dynamic setting where the scene geometry can change over time. Furthermore, the textured reconstructions shown in this thesis are simply computed by averaging colour values from the input images. To this end, it would be interesting to estimate material properties in a physically plausible way in order to allow for rendering the appearance more realistically.

Own Publications

[1] D. Hafner, C. Schroers, and J. Weickert. Introducing maximal anisotropy into second order coupling models. In J. Gall, P. Gehler, and B. Leibe, editors, *Pattern Recognition*, volume 9358 of *Lecture Notes in Computer Science*, pages 79–90, Berlin, 2015. Springer.

[2] V. Kramarev, O. Demetz, C. Schroers, and J. Weickert. Cross anisotropic cost volume filtering for segmentation. In K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, editors, *Computer Vision – ACCV 2012*, Lecture Notes in Computer Science, pages 803–814, Berlin, 2013. Springer.

[3] N. Persch, C. Schroers, S. Setzer, and J. Weickert. Introducing more physics into variational depth-from-defocus. In X. Jiang, J. Hornegger, and R. Koch, editors, *Pattern Recognition*, volume 8753 of *Lecture Notes in Computer Science*, pages 15–27, Berlin, September 2014. Springer.

[4] N. Persch, C. Schroers, S. Setzer, and J. Weickert. Physically inspired depth-from-defocus. Technical Report 355, Saarland University, Saarbrücken, Germany - Dept. of Math., February 2015.

[5] T. Schneevoigt, C. Schroers, and J. Weickert. A dense pipeline for 3D reconstruction from image sequences. In X. Jiang, J. Hornegger, and R. Koch, editors, *Pattern Recognition*, volume 8753 of *Lecture Notes in Computer Science*, pages 629–640, Berlin, September 2014. Springer.

[6] C. Schroers, H. Zimmer, L. Valgaerts, A. Bruhn, O. Demetz, and J. Weickert. Anisotropic range image integration. In A. Pinz, T. Pock, H. Bischof, and F. Leberl, editors, *Pattern Recognition*, volume 7476 of *Lecture Notes in Computer Science*, pages 73–82. Springer, Berlin, 2012.

[7] C. Schroers, S. Setzer, and J. Weickert. A variational taxonomy for surface reconstruction from oriented points. *Computer Graphics Forum*, 33(5):195–204, August 2014.

[8] C. Schroers, D. Hafner, and J. Weickert. Multiview depth parameterisation with second order regularisation. In J. Aujol, M. Nikolova, and N. Papadakis, editors, *Scale Space and Variational Methods in Computer Vision*, volume 9087 of *Lecture Notes in Computer Science*, pages 551–562, Berlin, 2015. Springer.

[9] O. Wang, C. Schroers, H. Zimmer, M. Gross, and A. Sorkine-Hornung. VideoSnapping: Interactive synchronization of multiple videos. *ACM Transactions on Graphics*, 33(4):77:1–77:10, July 2014.

[10] J. Weickert, S. Grewenig, C. Schroers, and A. Bruhn. Cyclic schemes for PDE-based image analysis. *International Journal of Computer Vision*, 118:275–299, July 2016.

Bibliography

[11] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, volume 6312 of *Lecture Notes in Computer Science*, pages 29–42. Springer, Berlin, 2010.

[12] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, Oct. 2011.

[13] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, Mar. 2011.

[14] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision – ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer, Berlin, 2006.

[15] M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, and C. T. Silva. A benchmark for surface reconstruction. *ACM Transactions on Graphics*, 32(2):20:1–20:17, Apr. 2013.

[16] F. Bernadini and H. Rushmeier. The $3D$ model acquisition pipeline. *Computer Graphics Forum*, 21(2):149–172, 2002.

[17] A. Boulch and R. Marlet. Fast and robust normal estimation for point clouds with sharp features. *Computer Graphics Forum*, 31(5):1765–1774, Aug. 2012.

[18] K. Bredies, K. Kunisch, and T. Pock. Total generalized variation. *SIAM Journal on Imaging Sciences*, 3(3):492–526, Sept. 2010.

[19] D. C. Brown. A solution to the general problem of multiple station analytical stereo triangulation. Technical Report 43, Patrick Airforce Base, Florida, 1958.

[20] M. Byröd and K. Åström. Conjugate gradient bundle adjustment. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, volume 6312 of *Lecture Notes in Computer Science*, pages 114–127. Springer, Berlin, 2010.

[21] F. Calakli and G. Taubin. SSD: Smooth signed distance surface reconstruction. *Computer Graphics Forum*, 30(7):1993–2002, 2011.

[22] F. Calakli and G. Taubin. SSD-C: Smooth signed distance colored surface reconstruction. In J. Dill, R. Earnshaw, D. Kasik, J. Vince, and P. C. Wong, editors, *Expanding the Frontiers of Visual Analytics and Visualization*, pages 323–338, London, 2012. Springer.

[23] A. Chambolle and P.-L. Lions. Image recovery via total variation minimization and related problems. *Numerische Mathematik*, 76(2):167–188, Apr. 1997.

[24] T. F. Chan and S. Esedoglu. Aspects of total variation regularized L^1 function approximation. *SIAM Journal on Applied Mathematics*, 65 (5):1817–1837, 2004.

[25] T. F. Chan and P. Mulet. On the convergence of the lagged diffusivity fixed point method in total variation image restoration. *SIAM Journal on Numerical Analysis*, 36(2):354–367, 1999.

[26] T. F. Chan, S. Esedoglu, and M. Nikolova. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM Journal on Applied Mathematics*, 66(5):1632–1648, 2006.

[27] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics*, 32(4):113:1–113:16, 2013.

[28] J. Civera, A. J. Davison, and J. Montiel. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, Oct. 2008.

[29] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. SIGGRAPH 96*, volume 3, pages 303–312, New York, NY, USA, Aug. 1996. ACM.

[30] J. Duchon. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *RAIRO Analyse Numérique*, 10:5–12, 1976.

[31] H. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, June 2006.

[32] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular slam. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8690 of *Lecture Notes in Computer Science*, pages 834–849. Springer, Berlin, Sept. 2014.

[33] C. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. In *In Photogrammetric Computer Vision*, pages 124–131, 2006.

[34] H. Federer. Curvature measures. *Transactions of the American Mathematical Society*, 93(3):418–491, 1959.

[35] D. Ferstl, C. Reinbacher, R. Ranftl, M. Ruether, and H. Bischof. Image guided depth upsampling using anisotropic total generalized variation. In *Proc. IEEE International Conference on Computer Vision*, pages 993–1000, Sydney, Australia, Dec. 2013.

[36] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics*, pages 544–552, 2005.

[37] W. H. Fleming and R. Rishel. An integral formula for total gradient variation. *Archiv der Mathematik*, 11(1):218–222, 1960.

[38] 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.

[39] F. Fraundorfer, D. Scaramuzza, and M. Pollefeys. A constricted bundle adjustment parameterization for relative scale estimation in visual odometry. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1899–1904, Anchorage, Alaska, 2010.

[40] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[41] S. Fučìk, A. Kratochvìl, and J. Nečas. Kačanov-Galerkin method. *Commentationes Mathematicae Universitatis Carolinae*, 14(4):651–659, 1973.

[42] I. M. Gelfand and S. V. Fomin. *Calculus of Variations*. Dover Publications, Mineola, NY, USA, 2000.

[43] M. Goesele, B. Curless, and S. M. Seitz. Multi-view stereo revisited. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2402–2409, New York, NY, USA, June 2006.

[44] G. Graber, J. Balzer, S. Soatto, and T. Pock. Efficient minimal-surface regularization of perspective depth maps in variational stereo. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, June 2015.

[45] A. Gray, E. Abbena, and S. Salamon. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. Chapman & Hall/CRC, 3rd edition, 2006.

[46] S. Grewenig. *Fast Explicit Methods for PDE-Based Image Analysis*. PhD thesis, Dept. of Mathematics and Computer Science, Saarland University, Saarbrücken, Germany, 2013.

[47] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. Geodesic star convexity for interactive image segmentation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 3129–3136, San Francisco, CA, June 2010.

[48] B. C. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. Springer, New York, 2003.

[49] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2003.

[50] W. B. Heard. *Rigid Body Mechanics: Mathematics, Physics and Applications*. Wiley-VCH, Weinheim, 2006.

[51] C. Hernandez, G. Vogiatzis, and R. Cipolla. Probabilistic visibility for multi-view stereo. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, MN, USA, June 2007.

[52] A. Hewer, J. Weickert, H. Seibert, T. Scheffer, and S. Diebels. Lagrangian strain tensor computation with higher order variational models. In *Proc. British Machine Vision Conference*, pages 129.1–129.10, Bristol, UK, Sept. 2013. BMVA Press.

[53] A. Hilton, A. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In B. Buxton and R. Cipolla, editors, *Computer Vision – ECCV '96*, volume 1064 of *Lecture Notes in Computer Science*, pages 117–126. Springer Berlin, 1996.

[54] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Societ of America A*, 5(7):1127–1135, 1988.

[55] D. Q. Huynh. Metrics for 3D rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, Oct. 2009.

[56] H. Ishikawa. Total absolute Gaussian curvature for stereo prior. In Y. Yagi, S. Kang, I. Kweon, and H. Zha, editors, *Computer Vision – ACCV 2007*, volume 4844 of *Lecture Notes in Computer Science*, pages 537–548. Springer Berlin, 2007.

[57] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proc. 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568, New York, NY, 2011. ACM.

[58] M. Kazhdan and H. Hoppe. Screened Poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3):29:1–29:13, July 2013.

[59] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 61–70. Eurographics Association, 2006.

[60] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, Philadelphia, 1995.

[61] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, pages 1–10, Washington, DC, 2007. IEEE Computer Society.

[62] K. Kolev, M. Klodt, T. Brox, and D. Cremers. Propagated photoconsistency and convexity in variational multiview 3D reconstruction.

In *Workshop on Photometric Analysis for Computer Vision*, Rio de Janeiro, Brazil, Oct. 2007.

[63] K. Kolev, M. Klodt, T. Brox, S. Esedoglu, and D. Cremers. Continuous global optimization in multiview 3d reconstruction. In A. L. Yuille, S.-C. Zhu, D. Cremers, and Y. Wang, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, volume 4679 of *Lecture Notes in Computer Science*, pages 441–452. Springer, Berlin, 2007.

[64] K. Kolev, T. Pock, and D. Cremers. Anisotropic minimal surfaces integrating photoconsistency and normal information for multiview stereo. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, volume 6313 of *Lecture Notes in Computer Science*, pages 538–551. Springer, Berlin, 2010.

[65] R. Kolluri. Provably good moving least squares. *ACM Transactions on Algorithms*, 4(2):18:1–18:25, May 2008. ISSN 1549-6325.

[66] S.-H. Lai and B. C. Vemuri. Reliable and efficient computation of optical flow. *International Journal of Computer Vision*, 29(2):87–105, 1998.

[67] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, Feb. 1994.

[68] V. Lempitsky and Y. Boykov. Global optimization for shape fitting. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, MN, USA, June 2007.

[69] V. Lempitsky, Y. Boykov, and D. Ivanov. Oriented visibility for multiview reconstruction. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision – ECCV 2006*, volume 3953 of *Lecture Notes in Computer Science*, pages 226–238. Springer, Berlin, 2006.

[70] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathmatics*, 2(2):164–168, 1944.

[71] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proc. SIGGRAPH 87*, volume 21, pages 163–169, July 1987.

[72] M. A. Lourakis and A. Argyros. SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software*, 36(1):1–30, 2009.

[73] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[74] J. Manson, G. Petrova, and S. Schaefer. Streaming surface reconstruction using wavelets. *Computer Graphics Forum*, 27(5):1411–1420, 2008.

[75] A.-R. Mansouri, A. Mitiche, and J. Konrad. Selective image diffusion: Application to disparity estimation. In *Proc. IEEE International Conference on Image Processing*, pages 284–288, Chicago, IL, Oct. 1998.

[76] R. March. Computation of stereo disparity using regularization. *Pattern Recognition Letters*, 8(3):181 – 187, 1988.

[77] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[78] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[79] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc. Tenth IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 127–136, Washington, DC, 2011. IEEE Computer Society.

[80] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proc. IEEE International Conference on Computer Vision*, pages 2320–2327, Barcelona, Spain, Nov. 2011.

[81] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics*, 32(6):169:1–169:11, Nov. 2013.

[82] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics*, pages 463–470, 2003.

[83] A. C. Öztireli, G. Guennebaud, and M. Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum*, 28(2):493–501, 2009.

[84] B. Payne and A. Toga. Distance field manipulation of surface models. *Computer Graphics and Applications, IEEE*, 12(1):65–71, Jan 1992.

[85] K. Pulli, T. Duchamp, H. Hoppe, J. Mcdonald, L. Shapiro, and W. Stuetzle. Robust meshes from multiple range maps. In *Proc. International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 205–211, 1997.

[86] R. Ranftl, S. Gehrig, T. Pock, and H. Bischof. Pushing the limits of stereo using variational stereo estimation. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 401–407, Alcala de Henares, Spain, June 2012.

[87] R. Ranftl, K. Bredies, and T. Pock. Non-local total generalized variation for optical flow estimation. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8689 of *Lecture Notes in Computer Science*, pages 439–454. Springer, Berlin, 2014.

[88] L. Robert and R. Deriche. Dense depth map reconstruction: A minimization and regularization approach which preserves discontinuities. In B. Buxton and R. Cipolla, editors, *Computer Vision – ECCV 96*, volume 1064 of *Lecture Notes in Computer Science*, pages 439–451. Springer, Berlin, 1996.

[89] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[90] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2nd edition, 2003.

[91] D. Scharstein and R. Szeliski. Stereo matching with non-linear diffusion. *International Journal of Computer Vision*, 28:155–174, June 1998.

[92] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[93] T. Schneevoigt. Camera motion estimation based on optic flow. Master's thesis, Dept. of Mathematics and Computer Science, Saarland University, Saarbrücken, Germany, 2013.

[94] C. Schroers. Variational range image integration. Master's thesis, Dept. of Mathematics and Computer Science, Saarland University, Saarbrücken, Germany, 2011.

[95] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 519–528, New York, NY, USA, June 2006.

[96] B. Semerjian. A new variational framework for multiview surface reconstruction. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8694 of *Lecture Notes in Computer Science*, pages 719–734. Springer, Berlin, Sept. 2014.

[97] Q. Shan, B. Curless, Y. Furukawa, C. Hernandez, and S. Seitz. Occluding contours for multi-view stereo. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 4002–4009, Columbus, OH, June 2014.

[98] C. Shen, J. F. O'Brien, and J. R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. *ACM Transactions on Graphics*, pages 896–904, 2004.

[99] J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, Seattle, WA, June 1994.

[100] P. G. Sibley and G. Taubin. Vectorfield isosurface-based reconstruction from oriented points. In *ACM SIGGRAPH 2005 Sketches*, New York, 2005.

[101] E. P. Simoncelli, E. H. Adelson, and D. J. Heeger. Probability distributions of optical flow. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 310–315, Lahaina, Maui, HI, June 1991.

[102] N. Slesareva, A. Bruhn, and J. Weickert. Optic flow goes stereo: A variational method for estimating discontinuity-preserving dense disparity maps. In W. G. Kropatsch, R. Sablatnig, and A. Hanbury, editors, *Pattern Recongnition*, volume 3663 of *Lecture Notes in Computer Science*, pages 33–40, Berlin, 2005. Springer.

[103] Stanford 3D Scanning Repository. Stanford 3D Scanning Repository. http://graphics.stanford.edu/data/3Dscanrep/, 2014.

[104] C. Strecha and L. V. Gool. PDE-based multi-view depth estimation. In *First International Symposium on 3D Data Processing Visualization and Transmission (3DPVT)*, pages 416–425, June 2002.

[105] J. Stühmer, S. Gumhold, and D. Cremers. Real-time dense geometry from a handheld camera. In M. Goesele, S. Roth, A. Kuijper, B. Schiele, and K. Schindler, editors, *Pattern Recognition*, volume 6376 of *Lecture Notes in Computer Science*, pages 11–20. Springer, Berlin, Sept. 2010.

[106] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, volume 6311 of *Lecture Notes in Computer Science*, pages 438–451. Springer, Berlin, 2010.

[107] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, pages 298–372, London, UK, 2000. Springer-Verlag.

[108] L. Valgaerts, A. Bruhn, and J. Weickert. A variational model for the joint recovery of the fundamental matrix and the optical flow. In G. Rigoll, editor, *Pattern Recognition*, volume 5096 of *Lecture Notes in Computer Science*, pages 314–324. Springer, Berlin, 2008.

[109] L. Valgaerts, A. Bruhn, M. Mainberger, and J. Weickert. Dense versus sparse approaches for estimating the fundamental matrix. *International Journal of Computer Vision*, 96(2):212–234, 2012.

[110] R. S. Varga. *Matrix Iterative Analysis*. Prentice Hall, Englewood Cliffs, NJ, USA, 1962.

[111] Viscoda. Voodoo camera tracker 1.2.0, 2015. http://www.viscoda.com/index.php/en/products/non-commercial/voodoo-camera-tracker.

[112] G. Vogiatzis, C. Hernández Esteban, P. H. S. Torr, and R. Cipolla. Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2241–2246, Dec. 2007.

[113] S. Volz, A. Bruhn, L. Valgaerts, and H. Zimmer. Modeling temporal coherence for optical flow. In D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, editors, *Proc. IEEE International Conference on Computer Vision*, pages 1116–1123, Barcelona, Spain, Nov. 2011.

[114] C. Walder, B. Schölkopf, and O. Chapelle. Implicit surface modelling with a globally regularised basis of compact support. *Computer Graphics Forum*, 25(3):635–644, 2006.

[115] J. Weickert. *Anisotropic Diffusion in Image Processing*. Teubner, Stuttgart, 1998.

[116] J. Weickert and C. Schnörr. A theoretical framework for convex regularizers in PDE-based computation of image motion. *International Journal of Computer Vision*, 45(3):245–264, Dec. 2001.

[117] J. Weickert and C. Schnörr. Variational optic flow computation with a spatio-temporal smoothness constraint. *Journal of Mathematical Imaging and Vision*, 14(3):245–255, May 2001.

[118] J. Weickert, M. Welk, and M. Wickert. l^2-stable nonstandard finite differences for anisotropic diffusion. In A. Kuijper, K. Bredies, T. Pock, and H. Bischof, editors, *Scale Space and Variational Methods in Computer Vision*, volume 7893 of *Lecture Notes in Computer Science*, pages 380–391. Springer Berlin, 2013.

[119] M. D. Wheeler, Y. Sato, and K. Ikeuchi. Consensus surfaces for modeling 3D objects from multiple range images. In *Proc. Sixth International Conference on Computer Vision*, pages 917–924, Bombay, India, Jan. 1998.

[120] C. Wu. VisualSFM : A Visual Structure from Motion System. `http://homes.cs.washington.edu/~ccwu/vsfm/`, 2014.

[121] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 3057–3064, Colorado Springs, CO, June 2011.

[122] D. M. Young. *Iterative Solution of Large Linear Systems*. Academic Press, New York, 1971.

[123] C. Zach. Fast and high quality fusion of depth maps. In *Proc. Fourth International Symposium on 3D Data Processing, Visualization and Transmission*, pages 1–8, Atlanta, GA, June 2008.

[124] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust TV-L^1 range image integration. In *Proc. Ninth International Conference on Computer Vision*, pages 1–8, Rio de Janeiro, Brazil, Oct. 2007.

[125] Q.-Y. Zhou and V. Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics*, 32(4):112:1–112:8, 2013.

[126] H. Zimmer, A. Bruhn, L. Valgaerts, M. Breuß, J. Weickert, B. Rosenhahn, and H.-P. Seidel. PDE-based anisotropic disparity-driven stereo vision. In *Proc. Vision, Modeling, and Visualization*, pages 263–272, Konstanz, Germany, Oct. 2008. Akademische Verlagsgesellschaft Aka.

[127] H. Zimmer, A. Bruhn, J. Weickert, L. Valgaerts, A. Salgado, B. Rosenhahn, and H.-P. Seidel. Complementary optic flow. In D. Cremers, Y. Boykov, A. Blake, and F. R. Schmidt, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, Lecture Notes in Computer Science, pages 207–220. Springer, Berlin, 2009.