Universität des Saarlandes
Faculty of Natural Sciences and Technology I
Department of Computer Science

Master's Thesis

Graph Cuts in Computer Vision

Oliver Demetz

submitted February 19, 2009

Supervisor Prof. Dr. Joachim Weickert
Advisor Dr. Andrés Bruhn
Reviewers Prof. Dr. Joachim Weickert
Prof. Dr. Matthias Hein

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement under Oath

I confirm under oath that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Oliver Demetz
Saarbrücken, February 19, 2009

Contents

1 Introduction

This thesis addresses a powerful method for discrete optimisation problems, called *graph cuts*. These methods have become popular in the last years [12, 13, 38] although their general capabilities are well known already since the 1960s.

Graph cuts are applicable to many computer vision problems. Besides classical image regularisation and denoising [12, 13], there are plenty of publications on segmentation methods using graph cuts [58, 9, 51, 45, 11]. Also the estimation of optical flow [13, 42, 5, 38] and stereo vision [12, 13, 38, 65, 5] are numerously treated problems. Moreover, there also exist several successful graph cut based algorithms for scene reconstruction from multiple views [39, 52]. But graph cut methods are not limited to the field of image processing and computer vision. In principle, they are a well suited method for minimising a wide class of discrete functions and could be applied to any problem, that fulfils certain properties [40]. As an example, in the field of machine learning massive research is being done on graph gut ideas. One example is semi-supervised learning, where one tries to expand the small set of labelled input data with such methods [6, 7]. Another field of application where graphs play an important role are spectral clustering methods [61]. There, one tries to cluster a graph whose vertices represent data measurement points and whose edges represent the similarity of such data points. The interesting speciality of spectral clustering methods is that they try to solve this problem by analysing the eigenvalues and eigenvectors of matrices that can be computed from the graphs instead of employing cuts on this graph. But besides all these wide spread application possibilities, we want to focus on image regularisation and stereo vision in this thesis.

The general procedure of all graph cut optimisation methods we consider is essentially always the same: For the given problem we formulate an appropriate discrete cost function. Then a graph consisting of nodes and arcs is constructed. This graph then separated – or cut – into several subgraphs – usually two subgraphs. Finally, due to the special construction of our graph, we can identify a solution of our cost function with this partition. Provided we found a minimal cut, we can then be sure to also have a solution of our objective function with minimal energy.

Our first tries on graph cuts will be made for signal regularisation. There we will consider the problem of filtering discrete 1-D signals, which are assumed to be deteriorated by noise. This filtering can be expressed as an energy function, whose minimum is the filtered signal. We will also compare

the minimisation results obtained via graph cuts with the output of related variational ideas.

The main topic of this thesis shall be stereo vision. In this task, we are given a pair of images, and the goal is to recover the depth of the scene. The main problem that has to be solved for this task is to establish a correspondence relation between pixels of the two images. We will consider the simplified case of the ortho-parallel setting in this thesis and concentrate on the minimisation of a a suitable cost function, that we will have to formulate. As a first step towards this problem we will discuss a successful publication by Boykov, Veksler and Zabih [13] in detail.

A big problem in the context of stereo vision is the naturally occurring effect of occlusions, where due to the geometrical setting certain parts of the scene are only visible from one camera. The problem here is that simple cost functions cannot resemble this phenomenon correctly and thus introduce a systematic error. To overcome this problem we will discuss a very successful publication by Kolmogorov and Zabih [38] in section 7, which completely reformulates the stereo problem and imposes model assumptions that are able to handle occlusions adequately.

Inspired by a variational method [3] and based on [13], we will finally formulate a cross-checking based stereo algorithm that also incorporates occlusions.

Organisation This thesis is organised as follows: We will start discussing general graph theoretic modalities in section 2 that explain why all the following considerations are valid. In this section we will prove the famous Min-Cut-Max-Flow theorem and introduce algorithms to determine maximal flows. The following section will then introduce one-dimensional signal regularisation ideas and methods. We will introduce one of the most classical and basic graph constructions there and compare the obtained results with the results from other well known methods. After these first experiences with graph cut methods, we will discuss the important role of smoothness penalty functions in section 4. Starting from section 5, we will concentrate on issues directly related to stereo vision. After a first introduction and explaination of the problem, we will use two different approaches to solve the correspondence problem in sections 6 and 7. Finally we will present own extensions to the methods from the previous sections in section 8 and conclude this thesis in last section.

2 Graph Theoretical Aspects

In order to be able to apply graph cuts on real imagery later on, we will try in this section first to introduce all necessary basic graph theoretical concepts. We will start defining formally what a graph and related important notions are, then we will prove the result which is for us most important, the so-called Min-Cut-Max-Flow theorem. Finally we will discuss practical algorithms to determine maximal flows and minimal cuts.

2.1 Flow Network Definition

In this section, we want to clarify and define the notation needed to describe what we will call a *graph* throughout this document.

Before giving precise definitions, one should remark that the entity that we mean when speaking of a *graph*, is also known under the name *flow network* in the majority of the graph theoretical literature (e.g. [64, 23]). Within this document we will use both notions side-by-side.

A *graph*, or *flow network*,

$$\mathcal{G} = < \mathcal{V}, \mathcal{E} >$$

consists of a set of *nodes*, or *vertices*, $v \in \mathcal{V}$ and a set of *edges*, or *arcs*, $e \in \mathcal{E}$. We will mainly consider undirected graphs for our applications.

We will now define properties and invariants of graphs, but before giving these formalisms, one should get an intuition on what a real world flow network might be. One simple example (among many many others) is the sewage water system of a city. It consists of a large network of water pipes, which are obviously the counterparts to the edges of a graph. These pipes often join, or one pipe ends in another pipe. All such junctions correspond to the nodes of a graph, because only there two or more pipes can be connected. In the following, we will define two important functions on graphs: the *capacity* and the *flow* between two adjacent nodes. For the sewage system example, one could identify the diameter of a pipe with the capacity of the corresponding edge. The larger its diameter is, the more water can flow through it. And by that already the meaning of a flow on an edge becomes clear. It can be understood as the amount of water flowing through it per time unit.

Further, there are some invariants a graph has to fulfil. One is, that for each normal node, there cannot flow more water into it, than leaves it. Physicists might also know this rule as *Kirchhoff's law*. There are some

special nodes, for which Kirchhoff's law may be violated. The so-called *terminal* nodes $\mathcal{T} \subset \mathcal{V}$ are by definition allowed to produce or consume flow. Their necessity is obvious, because the water must come from somewhere and go somewhere. So, all the connected houses and the drain hole covers could be identified as *source* nodes, and a river might be a *sink* node for our example. But for the whole system, the sum of all flows produced in sources must be equal to the sum of all consumed flows in sink nodes. Another invariant is that the flow on a pipe may not exceed its capacity. Otherwise it might happen that the pipe would burst.

To become more precise, the capacity is a function

$$c : \mathcal{E} \to \mathbb{R}_0^+$$

which assigns each edge a non-negative value. Sometimes, one also speaks of the weight of an edge instead of its capacity. To emphasise the direction of flows and capacities, we will use a slightly different version of the capacity function taking the two nodes that are connected by this edge as argument:

$$c : \mathcal{V} \times \mathcal{V} \to \mathbb{R}_0^+$$

Using this notion taking two nodes as argument, we can also define the capacity of a directed edge formally. Besides the capacity, also the flow is a function

$$f : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$$

which assigns a real value to each edge. Note that flows are directed quantities. In figure 1, an example of a flow network can be seen.

It is important to note that the flow function as well as the capacity function are defined as quantities of *one* edge. This means that the versions taking two nodes as argument resemble the capacity and the flow of a *direct* edge between these two nodes, and not of a path consisting of several edges. If there is no edge between two nodes, then the flow and capacity between them is zero. For example, flow and capacity between node s and v_3 in figure 1 are zero.

An important property of a flow function is its skew symmetry, and can be expressed as

$$f(u, v) \;=\; -\, f(v, u) \tag{2.1}$$

This rule also implies, that even if there existed more than one edge between two nodes u and v, the flow function resembles the accumulated flow over

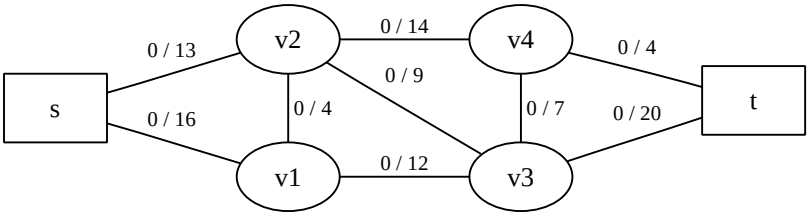


Figure 1: Undirected example graph. Slightly different version of [23, Chapter 26]. Edges are labelled $f(e)/c(e)$. Terminal nodes are box-shaped.

all these edges. In practice, one can neglect this problem by constructing graphs that only have one single edge to connect two nodes. For two subsets of the node set $X, Y \subset \mathcal{V}$, we can define the *net flow* from X to Y as

$$f(X, Y) := \sum_{u \in X} \sum_{v \in Y} f(u, v) \tag{2.2}$$

The previously mentioned Kirchhoff's law can then be expressed as

$$\sum_{v \in \mathcal{V}} f(u, v) = 0 \qquad \forall u \in \mathcal{V} \setminus \mathcal{T} \tag{2.3}$$

where $\mathcal{T} \subset \mathcal{V}$ is the set of all terminal nodes. In our applications, we will usually have two terminal nodes, one source node denoted by s, and one sink node t. Further following the previous example, it must hold that

$$f(u, v) \leq c(u, v) \qquad \forall u, v \in \mathcal{V} \tag{2.4}$$

Besides the notion of the capacity of an edge, i.e. how much water the pipe can transport, another important quantity for flow networks is the so-called *residual capacity*. It is a function

$$c_f : \mathcal{V} \times \mathcal{V} \to \mathbb{R}_0^+$$

that returns the amount of flow, that, given a flow f, an edge still could carry, and can be computed by

$$c_f(u, v) = c(u, v) - f(u, v) \tag{2.5}$$

Note, that the residual network of an undirected graph is a directed graph, because if an edge has a current flow, then the residual capacity in the

Figure 2: Example graph $\mathcal{G} = < \{u, v\}, \{e\} >$ consisting of two nodes u, v and one edge e. e has (undirected) capacity 10 and the flow on it is $f(u, v) = 3$. **Left:** Flow network. The diamond shaped arrow head of the edge indicates flow direction. **Right:** Corresponding residual flow network. In the residual network, the edge from u to v has residual capacity $10-3 = 7$, and $c_f(v, u) = 10 - (-3) = 13$.

direction of the flow is smaller than in the opposite direction. This effect is illustrated in figure 2 with one edge between two nodes.

A similar and at first glance unintuitive effect concerns directed edges: Let a directed edge from u to v have capacity C. If no flow is on that edge, we have $c(u, v) = C$ and in the opposite direction $c(v, u) = 0$. Also for the residual capacities, we get $c_f(u, v) = C$ and $c_f(v, u) = 0$. Now assume there is a flow $F \leq C$ on that edge from u to v. The residual capacity in direction of the flow changes as expected: $c_F(u, v) = C - F \geq 0$. But, the opposite direction which had zero residual capacity beforehand, now has

$$c_f(v, u) = c(v, u) - f(v, u) = 0 - (-F) = F \tag{2.6}$$

This means that by sending a flow along (u, v), a previously not existing edge has been opened in the residual network.

2.2 Maximal Flows and Minimal Cuts on Graphs

As already the title of this thesis promises, this document will deal not only with graphs but especially also with cuts of such graphs.

Let a graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ be given. For now, let us assume that $\mathcal{G}$ has two terminal nodes, a source s and a sink t, $\{s, t\} = \mathcal{T} \subset \mathcal{V}$.

A *cut* $\mathcal{C} \subset \mathcal{E}$ on a graph $\mathcal{G}$ is a subset of the edge set having the property that in the *induced graph*

$$\mathcal{G}^{\mathcal{C}} := < \mathcal{V} \, , \, \mathcal{E} \setminus \mathcal{C} >$$

the terminal nodes are separated from each other. This means that the set of nodes $\mathcal{V}$ is partitioned into two sets $\mathcal{V}_s$ and $\mathcal{V}_t = \mathcal{V} \setminus \mathcal{V}_s$. Each of the arising parts has exactly one terminal node as element, i.e. $s \in \mathcal{V}_s$ and

$t \in \mathcal{V}_t$. Additionally we require for a cut, that it shall not contain more edges than necessary. Cuts as we will consider them will not have a proper subset that would still separate the terminals. Otherwise, the edge set of a graph itself would be a cut, which obviously does not make sense.

We will restrict our considerations on cuts in this section to the case where $|\mathcal{T}| = 2$ mainly, since the graphs we construct and cut in practice will all have two terminal nodes only.

With this notion of a cut at hand, we can now define the capacity as well as the flow of a cut. The capacity of a cut $c(\mathcal{C})$ is the sum of the capacities of all its edges.

$$c(\mathcal{C}) = \sum_{e \in \mathcal{C}} c(e) = \sum_{u \in \mathcal{V}_s} \sum_{v \in \mathcal{V}_t} c(u, v) \tag{2.7}$$

Similarly, the flow of a cut $\mathcal{C}$ is defined as the sum of the flow on the edges of $\mathcal{C}$

$$f(\mathcal{C}) = \sum_{e \in \mathcal{C}} f(e) = f(\mathcal{V}_s, \mathcal{V}_t) \tag{2.8}$$

The edges of $\mathcal{C}$ are said to cross the cut, which means for an (undirected) edge e from u to v that either $u \in \mathcal{V}_s$ and $v \in \mathcal{V}_t$, or $u \in \mathcal{V}_t$ and $v \in \mathcal{V}_s$.

For directed graphs, cuts are defined slightly differently. There, an edge is only part of the cut, if it is directed from $\mathcal{V}_s$ to $\mathcal{V}_t$. Then, the capacity and flow of that edge can be calculated as for undirected flow networks.

The flow of a graph is defined to be the sum of flow leaving the source, or, equivalently the sum of flow entering the sink of the graph.

$$|f(\mathcal{G})| = \sum_{u \in \mathcal{V}} f(s, u) = \sum_{u \in \mathcal{V}} f(u, t) \tag{2.9}$$

See figure 3 for an example of a graph transporting flow.

As the sewage system of a city has a certain maximal amount of water, that can be transported, so has every flow network a uniquely determinable *maximal flow* value. If a graph is at maximal flow, then it is not possible any more to increase the flow, because there exists no path with residual capacity that could be augmented any more. Note that while the value of the maximal flow can be uniquely determined, there still may exist many different flow functions for a given graph being a maximal flow.

2.2.1 Min-Cut-Max-Flow Theorem

Before stating the famous Min-Cut-Max-Flow theorem itself, we need to state and prove a few corollaries in order to prove the theorem.

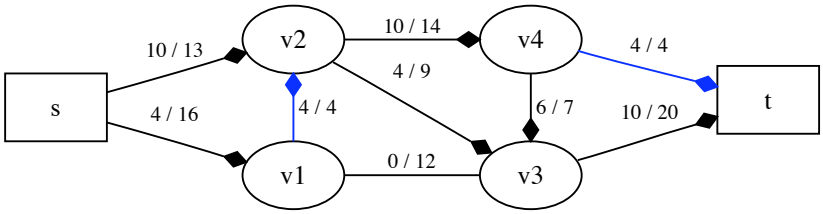


Figure 3: Undirected example graph transporting a flow. Slightly different version of [23, Chapter 26]. The diamond shaped edge endings indicate the direction of the flow. Note that the flow is not a maximum flow, since e.g. the path $s \rightarrow v_1 \rightarrow v_3 \rightarrow t$ could still be augmented.

There, let us assume a graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ with two terminals $\{s, t\} = \mathcal{T} \subset \mathcal{V}$ be given. Further, let $\mathcal{C} \subset \mathcal{E}$ be a cut partitioning the node set $\mathcal{V} = \mathcal{V}_s \cup \mathcal{V}_t$, $\mathcal{V}_s \cap \mathcal{V}_t = \emptyset$. Then we can state the following corollaries (see also [23]).

Corollary 1 For a flow f on $\mathcal{G}$, the following equalities are valid:

1. For any $X \subset \mathcal{V}$ holds: $f(X, X) = 0$
2. For all $X, Y \subset \mathcal{V}$ we have $f(X, Y) = -f(Y, X)$
3. For all $X, Y, Z \subset \mathcal{V}$ with $X \cap Y = \emptyset$ it holds that

$$f(Z, X \cup Y) = f(Z, X) + f(Z, Y)$$

Proof

1. By equation (2.2) we have

$$f(X, X) = \sum_{u \in X} \sum_{v \in X} f(u, v)$$

In this summation, each pair of elements of X appears twice, once as (u, v) and once as (v, u). Together with the skew symmetry (2.1) we can prove that the elements of the summation cancel out in pairs.

2. With equations (2.2) and (2.1) we can show

$$\begin{aligned} f(X,Y) &\overset{(2.2)}{=} \sum_{u\in X}\sum_{v\in Y} f(u,v) \\ &\overset{(2.1)}{=} \sum_{u\in X}\sum_{v\in Y} -f(v,u) \\ &= -\sum_{u\in X}\sum_{v\in Y} f(v,u) \\ &= -\sum_{v\in Y}\sum_{u\in X} f(v,u) \\ &= -f(Y,X) \end{aligned}$$

3. Writing out

$$\begin{aligned} f(Z,X\cup Y) &\overset{(2.2)}{=} \sum_{u\in Z}\sum_{v\in X\cup Y} f(u,v) \\ &\overset{X\cap Y=\emptyset}{=} \sum_{u\in Z}\sum_{v\in X} f(u,v) + \sum_{u\in Z}\sum_{v\in Y} f(u,v) \\ &\overset{(2.2)}{=} f(Z,X) + f(Z,Y) \end{aligned}$$

□

Corollary 2 The flow of a cut is equal to the flow of the graph.

$$f(\mathcal{V}_s,\mathcal{V}_t) = |f(\mathcal{G})| \tag{2.10}$$

Proof By corollary 1 we know that

$$\begin{aligned} f(\mathcal{V}_s,\mathcal{V}) &= f(\mathcal{V}_s,\mathcal{V}_s\cup\mathcal{V}_t) \\ &= f(\mathcal{V}_s,\mathcal{V}_s) + f(\mathcal{V}_s,\mathcal{V}_t) \end{aligned}$$

Further, we can write for the flow of the cut

$$\begin{aligned} f(\mathcal{V}_s,\mathcal{V}_t) &= f(\mathcal{V}_s,\mathcal{V}) - f(\mathcal{V}_s,\mathcal{V}_s) \\ &= f(\mathcal{V}_s,\mathcal{V}) && \text{by Corollary 1.1} \\ &= f(s\cup(\mathcal{V}_s\setminus s),\mathcal{V}) && \text{since } s\in\mathcal{V}_s \\ &= f(s,\mathcal{V}) + f(\mathcal{V}_s\setminus s,\mathcal{V}) && \text{by Corollary 1.3} \\ &= f(s,\mathcal{V}) && \text{because } f(\mathcal{V}_s\setminus s,\mathcal{V})=0 \text{ by (2.1)} \\ &= |f(\mathcal{G})| \end{aligned}$$

□

Corollary 3 The value of a flow in a graph cannot be larger than the capacity of any cut of that graph.

Proof We have

$$\begin{aligned} |f(\mathcal{G})| &= f(\mathcal{V}_s, \mathcal{V}_t) \\ &= \sum_{u \in \mathcal{V}_s} \sum_{v \in \mathcal{V}_t} f(u, v) \\ &\leq \sum_{u \in \mathcal{V}_s} \sum_{v \in \mathcal{V}_t} c(u, v) \\ &= c(\mathcal{V}_s, \mathcal{V}_t) \end{aligned} \tag{2.11}$$

□

Minimal cuts and maximal flows are closely related problems. The following theorem states their equivalence.

Min-Cut-Max-Flow Theorem Let $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ be a graph according to section 2, and let $s, t \in \mathcal{V}$ be the two terminal nodes. Then the three statements are equivalent:

1. f is a maximal flow.
2. There is no path in the residual graph $\mathcal{G}_f$ from s to t that could be augmented.
3. $|f(\mathcal{G})| = c(\mathcal{C})$ for some cut $\mathcal{C}$

Proof The proof for this theorem was given in 1956 by Elias et al. [27] and independently by Ford and Fulkerson [29] in the same year.

(**1**) ⇒ (**2**) Assume f was a maximum flow on $\mathcal{G}$. Now, assume there still existed a path $P \subset \mathcal{E}$ through $\mathcal{G}$ from s to t. If this path is an augmenting path, it fulfils

$$m := \min_{e \in P} c_f(e) \; > \; 0$$

This means that it must be possible to augment this path with flow m. The resulting new network flow f' then definitely fulfils $|f'(\mathcal{G})| > |f(\mathcal{G})|$ which contradicts the assumption that f is a maximum flow.

(**2**) ⇒ (**3**) Assume there was no path in $\mathcal{G}_f$ from s to t that could be augmented. This means that t is not reachable from s in the residual network. But still, it is straightforward to determine the subset $\mathcal{V}_s$ of $\mathcal{V}$ as

$$\mathcal{V}_s := \{v \in \mathcal{V} \mid v \text{ is reachable from } s \text{ in } \mathcal{G}_f\}$$

and

$$\mathcal{V}_t := \mathcal{V} \setminus \mathcal{V}_s$$

Obviously is $(\mathcal{V}_s, \mathcal{V}_t)$ a valid cut, since by construction the two parts are disjoint, and $s \in \mathcal{V}_s$ and $t \notin \mathcal{V}_s$. Further, and most important, for every edge crossing the cut, i.e. for each edge from $u \in \mathcal{V}_s$ to $v \in \mathcal{V}_t$, it must hold that $c(u, v) = f(u, v)$. If this was not the case, then v was reachable from s, which contradicts the previous construction. Now if all edges that cross the cut are saturated, we have

$$|f(\mathcal{G})| = f(\mathcal{V}_s, \mathcal{V}_t) = c\,(\mathcal{V}_s, \mathcal{V}_t)$$

(**3**) ⇒ (**1**) By corollary 3, we have that any flow on $\mathcal{G}$ can at most be equal to the capacity of any cut,

$$|f(\mathcal{G})| \;\leq\; c\,(\mathcal{V}_s, \mathcal{V}_t)$$

and by assumption (3) it even holds that

$$|f(\mathcal{G})| = c\,(\mathcal{V}_s, \mathcal{V}_t)$$

thus $|f(\mathcal{G})|$ cannot be larger and is a maximum flow.

□

2.3 Minimal Cut Algorithm

The existence of the min-cut-max-flow theorem is of central importance for our later applications, since this theorem offers a relatively straight forward algorithm to determine minimal cuts. We want to mention here, that there are other algorithms that compute minimal cuts in flow networks without incorporating maximal flows (see e.g. [54] and references therein), but the vast majority of minimal cut algorithms relies on maximum flow computations.

Throughout this thesis, we will consider two general classes of maximum flow algorithms. First, we will consider the most classical and straightforward methods for this problem, which are based on ideas first published by

Step 1: Determine maximal flow in $\mathcal{G}$

Step 2: Determine partition of $\mathcal{V}$ into $\mathcal{V}_s$ and $\mathcal{V}_t$

Step 3: $\mathcal{C} = \{(u, v) = e \in \mathcal{E} | u \in \mathcal{V}_s \wedge v \in \mathcal{V}_t\}$

Figure 4: Min-Cut algorithm overview.

Ford and Fulkerson [29]. After that, we will consider more advanced and faster methods for the problem, so-called push-relabel algorithms due to Goldberg et. al [33]. One should already here note that even though push-relabel methods at least bring the computation times for our applications with images into the range of minutes, still there is room for enhancements. But as said, we will see that the push-relabel algorithm provides enough efficiency for this thesis to fulfil its main objective of a proof of concept.

In the following, we will as a first step always determine the maximum flow in the given graph. Then, we will determine the two sets $\mathcal{V}_s$ and $\mathcal{V}_t$ corresponding to the minimal cut. Since the cut separates the two previously extracted sets, the cut edges are in a last step easily identifiable: All edges going from $\mathcal{V}_s$ into $\mathcal{V}_t$ are cut edges:

$$\mathcal{C} = \{(u, v) = e \in \mathcal{E} \mid u \in \mathcal{V}_s \wedge v \in \mathcal{V}_t\} \tag{2.12}$$

An overview over the algorithm is again printed in figure 4.

Concerning the first step, the max flow calculation, we will detail on this topic extensively in the upcoming section 2.4.

After a maximal flow is present in the graph, the next step is to identify $\mathcal{V}_s$ and $\mathcal{V}_t$. To understand how this can be achieved, we have to realise that $\mathcal{V}_s$ and $\mathcal{V}_t$ must be disjoint sets and that there does not exist a path in the residual graph $\mathcal{G}_f$ from s to t. Therefore it is sufficient to determine $\mathcal{V}_s$, the subset of $\mathcal{V}$ being reachable from the source node s in the residual network. This subset fulfils all requirements on the partition of $\mathcal{V}$ into $\mathcal{V}_s$ and $\mathcal{V}_t = \mathcal{V} \setminus \mathcal{V}_s$.

The last step is to identify the cut edges. It can be easily done by iterating once over all edges and evaluating their inclusion in $\mathcal{C}$ accordingly to (2.12).

We want to conclude this paragraph with figure 5 which shows the example graph from figures 1 and 3 with a maximum flow. Besides this, also its residual network together with the minimal cut is illustrated.

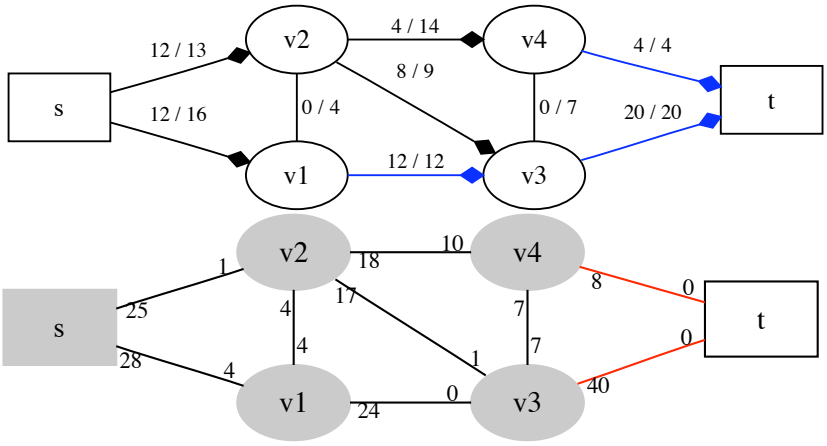


Figure 5: Undirected example graph from figure 1 transporting a maximal flow. **Top:** Flow network. Blue edges are saturated. **Bottom:** Residual flow network. Filled nodes are reachable from the source node s. Red edges belong to the minimal cut.

2.4 Maximal Flow Calculation

The generic minimal cut algorithm from section 2.3 employs a maximum flow calculation in its first step. This calculation shall be the topic of this section.

It turns out, that just this computation of the maximal flow is also in terms of computational effort by far the most costly part of the whole algorithm. The two remaining steps for the minimal cut determination are not expensive any more once the maximal flow is found: the exploration of the connected graph should not cost more than $O(|\mathcal{V}|)$ operations, and the following decision for each edge, whether it is a cut edge or not is one iteration over all edges, $O(|\mathcal{E}|)$.

Throughout the work on this thesis, we implemented two types of maximum flow algorithms: First we implemented the algorithm by Ford and Fulkerson [29] and several variants of it. While the latter algorithmic idea is very intuitive and straightforward, it turned out that the efficiency of Ford-Fulkerson type algorithms is too low in order to be applicable to computer vision problems.

A competing class of algorithms are so-called push-relabel algorithms, which will be discussed after introducing the algorithm by Ford and Fulkerson.

2.4.1 Ford-Fulkerson Algorithm

The algorithm by Ford and Fulkerson, first published in 1952, was the first so-called *augmenting path algorithm*. Algorithms of this class always work by repeatedly searching for a path from the source s to the sink t in the residual network. Once such a path is found, it is *augmented*. This augmentation stage will increment the flow on all edges lying on this path. If no path with free capacity can be found any more, the algorithm terminates, since the actual flow is a maximal flow. A listing of the steps of Ford-Fulkerson style methods can be found in listing 1. One concrete instance of such augment-

```
while ( p = find_path ( s, t ) ) do
  for each edge e in p do
    increment flow on e
  end
end
```

Listing 1: Ford-Fulkerson algorithm overview.

ing path algorithms is the algorithm by Dinic [26] from 1970, which was independently also found by Edmonds and Karp in 1972 (but developed at the same time [63]). The algorithm by Edmonds and Karp specified to use a breadth first search (BFS) algorithm for searching the paths from s to t in the residual network. By using BFS and defining the *length* of edges to be 1, the algorithm ensures to always find a shortest path. In listing 2 one can see the structure of the Edmonds-Karp algorithm. Please note that the amount of augmentation for each path is specified there to be the minimal (residual) capacity that all edges of the found path still have.

Concerning the worst-case running time of the Edmonds-Karp algorithm, we make the following important observations: In the worst case, searching a path in a flow network via breadth first search will cost $O(|\mathcal{E}|)$, since BFS visits each edge of a graph at most once. In each augmentation step the amount of augmentation is chosen such, that at least one edge on the actual path becomes saturated. We will call such an edge *critical*. One can show (see e.g. [23]) that each edge of the graph can become critical at most $|\mathcal{V}|/2 - 1$ times. In total, we get then a worst case complexity of $O(|\mathcal{V}| \cdot |\mathcal{E}|^2)$. To conclude this part on the Edmonds-Karp and

```
while ( p = BFS ( s , t ) ) do
  m = ∞
  for each edge e in p do
    m = min(m, c_f(e))
  end

  for each edge e in p do
    f(e) = f(e) + m
  end
end
```

Listing 2: Edmonds-Karp algorithm overview. The path search routine BFS performs a breadth first search and outputs a path $p \subset \mathcal{E}$.

Ford-Fulkerson methods, we want to give the steps and intermediate flow networks that one execution of the Edmonds-Karp algorithm may produce in figure 6. Note that these steps are not unique, since the order in which the nodes are visited is not uniquely defined.

It is of course possible to use other graph search strategies than a breadth first search. Depending on the graph structure, it might e.g. be more efficient to use a depth first search [23], or other strategies. But in general one should remark that all Ford-Fulkerson type max flow algorithms have poor performance, especially for graph structures as they will occur in our constructions. The reason for this is the general procedure of searching always a new augmentable path that is to be saturated then. Each new path search has to start from scratch and cannot reuse graph information from previous executions. This problem is the starting point of the idea of [10] of reusing this information if possible.

2.4.2 Push-Relabel Algorithm

We now want to investigate on a competing class of algorithms for the maximum flow problem. The first representative of the class of Push-Relabel algorithms was published in 1986 by Goldberg and Tarjan ([32, 33]).

There is one major difference between the previously discussed augmenting path algorithms and the push-relabel methods: While augmenting path algorithms always maintain a valid flow throughout their execution, push-

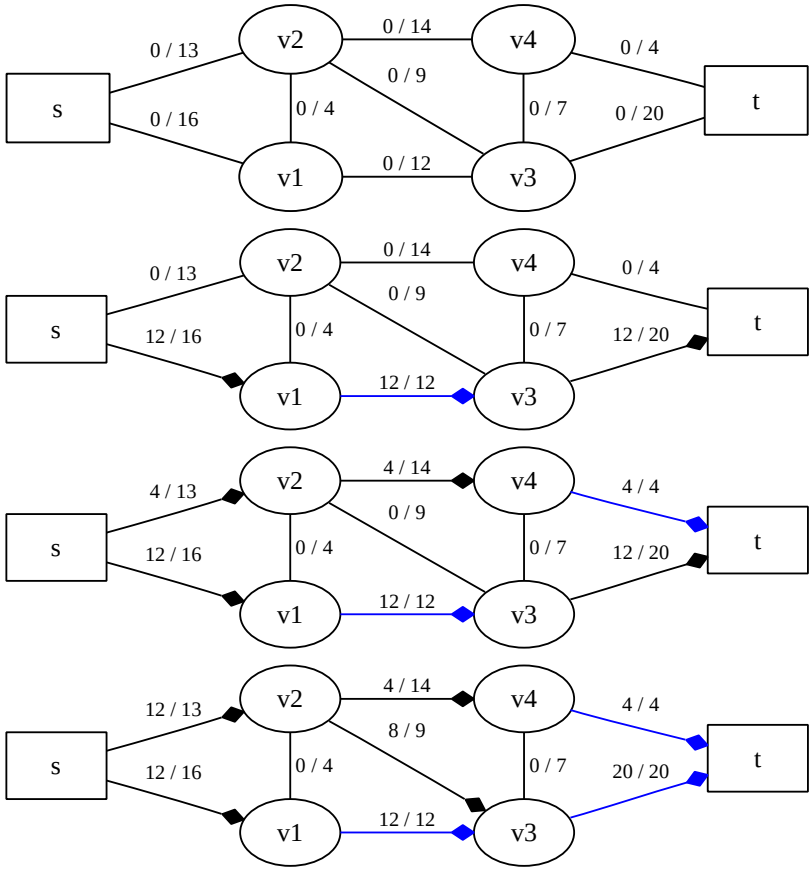


Figure 6: Example run of the Edmonds-Karp algorithm on the example graph from figure 1. **First Graph:** Input flow network with zero flow. **Second Graph:** Flow network after first augmentation. The path that was augmented was $s \to v_1 \to v_3 \to t$. Note that the edge between v_1 and v_3 is critical. **Third Graph:** Network after augmentation of path $s \to v_2 \to v_4 \to t$. **Fourth Graph:** Flow network after last augmentation $s \to v_2 \to v_3 \to t$. The flow is maximal.

relabel methods omit this property and maintain a so-called *preflow* with weakened validity constraints.

In order to explain the intuition behind this algorithm, we need to define two additional attributes of a vertex. The *excess* of a node is defined as

$$e(u) = \sum_{v \in \mathcal{V}} f(v, u) \tag{2.13}$$

A valid flow has to respect Kirchhoff's law, which constraints the excess to be equal to zero always. We will call a node *active*, if its excess is larger than zero. Further, we will assign a positive integer valued number to each node which we will call its *height*.

Intuitively speaking, we can understand the push-relabel method as follows: The vertices of the network are regarded as a landscape, where the height of each vertex might change over time. The height of a vertex should ideally correspond somehow to its distance to the sink node. By this the landscape descends towards the sink.

Initially, the height of all nodes is set to zero, except for the source, whose height is set to $|\mathcal{V}|$ fixed. Secondly, all outgoing edges of the source node are saturated. By this all direct children of the source node get a positive excess and become active by that.

The next step, which is applied to all active nodes until the algorithm terminates is the following: The actual node will try to *push* as much flow as possible to its neighbours. The only restriction is to only push flow *downhill*. By this latter constraint, we give the push operations a preferred direction towards the sink. If an active node has no neighbour whose height is smaller than his, a relabel operation is applied to this node: His height is increased to one plus the height of his lowest neighbour.

The temporarily weakened constraints on the flow function for the time of execution of the algorithms are summarised in the notion of a *preflow*. A flow is a preflow, if for all non-terminal vertices it holds that

$$e(u) \geq 0 \quad \forall u \in \mathcal{V} \setminus \mathcal{T}. \tag{2.14}$$

Intuitively, each vertex may have a reservoir to store flow. If the push-relabel algorithm terminates, the preflow is converted into a valid flow.

The core algorithm consists of three main procedures: push, relabel and discharge. Further, it maintains list of active nodes. As long as this list is not empty, it takes one node of this list and applies the procedure discharge to it. If the node is still active after this application, it is added to the list again.

A pseudo-code formulation of the three mentioned procedures is given in listing 3.

The initialisation of the algorithm is straight-forward. The height of each node are set to zero, except for the source vertex, whose height will be kept fixed at $|\mathcal{V}|$. Similarly the sink, which will always represent the deepest spot in the landscape, will stay at fixed height 0. As described, all

```
procedure push (vertex u, vertex v)
  f(u,v) = f(u,v) + min(c_f(u,v), e(u));
end

procedure relabel (vertex u)
  h = ∞;
  for all neighbours v of u do
    h = min(h,v.height);
  end
  u.height = h + 1;
end

procedure discharge (vertex u)
  do
    // n is the actual neighbour to consider
    n = u.neighbour_list(u.actual_neighbour_id);

    if((u.height == n.height+1) ∧ (c_f(u,n) > 0))
      push(u,n);
      continue;
    end

    // goto the next neighbour
    u.actual_neighbour_id += 1;
    if (u.actual_neighbour_id > u.neighbour_list.length)
      u.actual_neighbour_id = 1;
      relabel(u);
      return;
    end
  while (u.excess > 0)
end
```

Listing 3: Push-Relabel algorithm overview. Each node stores list of links to his neighbours as well as a link to his actual neighbour.

nodes that share a direct edge with the source node get an initial excess by saturating this edge at the beginning. In this way all those nodes become active and can push their excess towards the sink. In this way it will also

never be necessary again to push any flow from the source into the graph.

Analysing the behaviour of the algorithm, one can see, that the height of the nodes increases monotonically. At some point, all direct neighbours of the sink node will have height strictly larger than one. Since we only allowed to push flow to nodes with height exactly one unit smaller than oneself, from this stage on, no flow can be pushed into the sink any more. One can observe this height gap growing up to a value of $|\mathcal{V}|$ between these nodes and the sink.

The next interesting point of execution is when the height of all non-terminal nodes exceeds $|\mathcal{V}|$. At this stage, the minimal cut as well as the maximum flow value are already fixed, even though the algorithm has not terminated yet. The maximal flow value is equal to the excess of the sink node, and the minimal cut can be determined by the set of nodes which can reach the sink in the residual graph $\mathcal{G}_f$. All following steps of the algorithm will increase the landscape such that the stored excess can be discharged into the source node again. Note that since we are interested in minimal cuts, we will terminate our computations at the mentioned stage, moreover we will even stop all push and relabel operations at height $|\mathcal{V}|$. If one executes the algorithm until complete termination, it is obvious, that all excess is transported back into the source again, and by that the preflow is turned into a valid flow.

There are various actual implementations of the push-relabel algorithm in the literature [32, 33, 19], that differ especially in the choice of the active nodes list implementation and behaviour. For example, by using a FIFO list, the runtime of the algorithm can be shown to be in $O(|\mathcal{V}|^3)$([32]). While cubic complexities are not acceptable in practice, [19] proposed that use heuristics to reduce the empirical runtime drastically.

Heuristics In [19], the usage of two heuristics was proposed.

The first heuristic they propose is to completely relabel the heights of all nodes periodically, by setting them to the distance of the node to the sink. This can be realised by a breadth first search starting from the sink node t towards the source node s, see also [25]. Note that this operation must be performed in the residual network, but with *reversed* edge capacities, in order to reflect the reachability from the inner network nodes to the sink.

The big advantage of this action is that all following push operations are directed in a probably good direction by that. This measure in conjunction with a reordering of the FIFO queue produces a tremendous performance jump in practice. But since this global relabelling is computationally ex-

pensive compared to local push and relabel operations, it is only performed periodically, e.g. after $|\mathcal{V}|$ relabel operations.

The second heuristic being proposed in [19] is called *gap heuristic* and relies on an important observation: Assume for some height label $g < |\mathcal{V}|$, that at some stage of the algorithm no vertex has height g, but there are nodes v with heights $g < height(v) < |\mathcal{V}|$. The observation now is, that since there is a gap, all these nodes will not be able to send any flow across this gap into the sink any more. Therefore, it is valid to just set the height of all these nodes to $|\mathcal{V}|$ directly. Note that these node will by that not have to be considered any more by us, since we are only interested in minimal cuts.

2.4.3 Algorithms for special graph structures

If the structure of the arising graphs is known and fulfils some special invariants, it might pay off to invent specialised algorithms tailored to the specific problem. One instance of such an algorithm was given in [10]. It is tailored perfectly to the graphs that arise in early vision problems. We will see that such graphs consist only of the two terminal nodes and a number of so-called pixel nodes. The speciality is that each pixel node has a direct link to both terminals. Further, each pixel node is at most connected to 4 other pixel nodes. It is obvious that developing a specialised algorithm for such graphs is possible and was realised in [10]. Interestingly, this algorithm does not base on the Push-Relabel method, which is in its classical variant considerably faster than all other discussed algorithms. Instead, the Ford-Fulkerson algorithm is engaged, with additionally the possibility to reuse network information from previous augmentations.

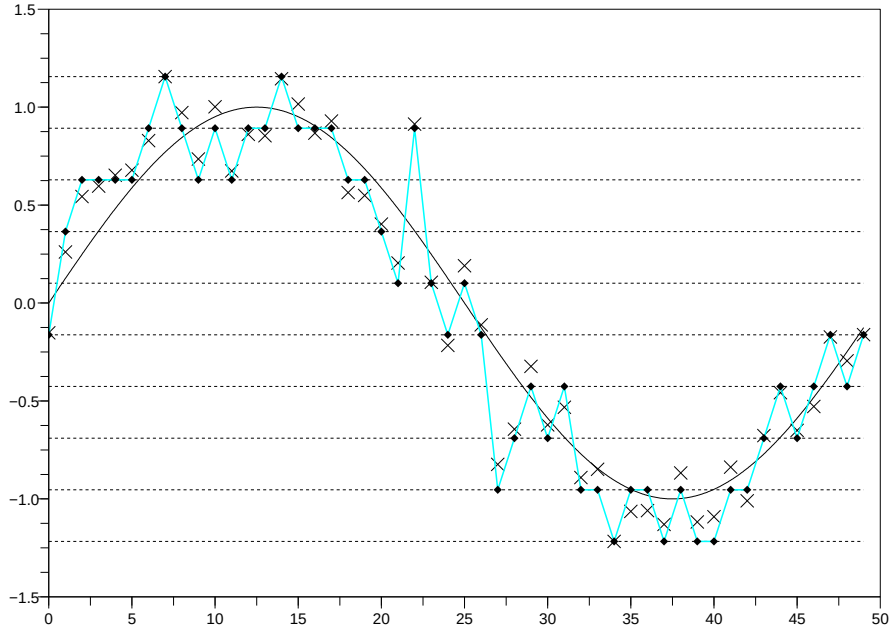


Figure 7: Example of an input signal as considered in this section. The black line shows the sin function that was used as example. The black crosses denote the samples that were taken from function and perturbed with additive Gaussian noise with standard deviation $\sigma = 0.2$. The perturbed samples were quantised at 10 equidistant quantisation levels, see table 1 for the actual values. The black diamond shaped marks show the result of this quantisation. The blue line interpolates the quantised samples with a piecewise linear function.

3 1-D Image Regularisation

Our very first tests on graph cuts are concerned with one dimensional discrete image (or signal-) regularisation, like proposed by Whittaker in 1923 [62] and later in 1977 by Tikhonov [55].

In this field, we consider 1-D signals $f : \mathbb{R} \to \mathbb{R}$, that are sampled on a regular grid with grid size h. Figure 7 presents a signal of such type. Instead of designing this filter for discrete datasets directly, we will perform the modelling of this regularisation method in continuous space. This leads us to variational expressions, which we will discretise in a second step. Usually, the input for regularisation methods are distorted or noisy signals.

The objective of image regularisation is to remove these perturbations and to output a filtered version u of f not containing such noise artefacts any more.

To realise this filter [62, 55] introduced an energy functional that is to be minimised by the desired function $u : \mathbb{R} \to \mathbb{R}$.

$$E(u) = \frac{1}{2} \int_\Omega (u - f)^2 \, \mathrm{d}x + \frac{\alpha}{2} \int_\Omega \left(\tfrac{\mathrm{d}}{\mathrm{d}x} u\right)^2 \mathrm{d}x \tag{3.1}$$

In this functional the first integration term is called *data term* because it penalises deviations of the filtered function from the noisy input signal f. If the output was the same as the input signal, the integral would evaluate to zero. Thus this term *keeps u near f*.

The second term in (3.1) is called *smoothness term*. It contributes to the energy functional whenever the filtered signal changes its value. This can be realised by considering the derivative, which is non-zero wherever the function changes. The reason to choose such a smoothness term is that empirically noise is a high-frequent component of a signal which consists of many small perturbations. Obviously, in such perturbations, the derivative is large, and hence, by penalising large derivatives a noisy signal cannot minimise the proposed energy functional. The solution must be smooth. The parameter $\alpha > 0$ gives us the possibility to control the influence of the smoothness term in this functional, it is often called regularisation parameter.

There are several practical implementation and realisation strategies, how to apply and perform such a filter in practice. In particular, the calculus of variations allows us to apply the so-called Euler-Lagrange theorem, which leads to a differential equation that has to be solved.

Any minimiser of equation (3.1) has to fulfil the corresponding Euler-Lagrange equation which reads

$$\frac{u - f}{\alpha} = \frac{\mathrm{d}^2}{\mathrm{d}x^2} u \tag{3.2}$$

This equation is only valid in conjunction with so-called natural boundary conditions

$$\frac{\mathrm{d}}{\mathrm{d}x} u = 0 \tag{3.3}$$

Equation (3.2) is known to be closely related to the (linear) *diffusion* equation, which describes how heat propagates. As a side remark we do not

want to conceal that one simple solution of this diffusion equation is a convolution of the signal with a Gaussian kernel whose width depends on the value of the mentioned regularisation parameter α.

Since we are designing our filters in continuous space, the main additional task we have to cope with after the design itself is the discretisation of the continuous ideas in order to get a filter that can be applied in practice.

Even though it is a common methodology to discretise the Euler-Lagrange equations, there also exists the possibility to discretise the energy functional and to disregard the Euler-Lagrange formalism. Having a comparison with graph cut methods in mind, this is also the way we want to progress.

Note that the Euler-Lagrange equations are differential equations that *just* have to be solved. Once discretised, we immediately would get a system of linear or non-linear equations. Besides this, a discretisation of the energy functional itself leads to a discrete function, for that we still need to find a minimiser.

Regarding the energy functional from equation (3.1), we see that it is necessary to approximate the differential equation $|\frac{d}{dx}u|^2$ in discrete space. In order to keep the resulting discretisation simple, and in order to stay immediately comparable to graph cut methods later on, we realise this by using simple one-sided finite differences methods. More precisely, we will approximate the derivative expression by so-called *forward differences* which can be computed as

$$\left(\frac{\mathrm{d}}{\mathrm{d}x}u\right)_i \approx \frac{u_{i+1} - u_i}{h} \tag{3.4}$$

where the discrete sample at position $i \in \mathbb{N}$ is sampled at i times the spatial grid size (sampling distance) $h > 0$

$$u_i = u(i \cdot h) \tag{3.5}$$

Assuming a grid size $h = 1$, we obtain the discretised version of the continuous energy functional from (3.1) as

$$E(\mathbf{u}) = \frac{1}{2}\sum_{i=1}^{n}(u_i - f_i)^2 + \frac{\alpha}{2}\sum_{i=1}^{n-1}(u_{i+1} - u_i)^2 \tag{3.6}$$

where $\mathbf{u} \in \mathbb{R}^n$. The result of this discretisation step is a function $E : \mathbb{R}^n \to \mathbb{R}$, where n is the size or length of the discrete signal.

As already mentioned, it is still necessary to find the minimum of this function. In general we know that the derivative of a function has to vanish

in extreme points. Here in this n-dimensional setting, we have to find positions, where the gradient of E vanishes

$$\nabla E := \begin{pmatrix} \frac{\partial E}{\partial x_1} \\ \vdots \\ \frac{\partial E}{\partial x_n} \end{pmatrix} \stackrel{!}{=} 0 \tag{3.7}$$

This in turn implies a linear system of equations which reads

$$\underbrace{\begin{pmatrix} 1+\alpha & -\alpha & & & \\ -\alpha & 1+2\alpha & -\alpha & & \\ & \ddots & \ddots & \ddots & \\ & & -\alpha & 1+2\alpha & -\alpha \\ & & & -\alpha & 1+\alpha \end{pmatrix}}_{=:A} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n \end{pmatrix} \tag{3.8}$$

One can easily verify that the system matrix $A \in \mathbb{R}^{n \times n}$ has tri-diagonal shape and has rank n. Thus the inversion of it is straightforward. After computing the system matrix A, one can obtain the filtered signal $\mathbf{u}$ by

$$\mathbf{u} = A^{-1}\mathbf{f} \tag{3.9}$$

In figure 8 we present results for the previously introduced filter for different choices of α. One can clearly see the smoothing and regularising character of this filter. Especially the sharp noise peak around $x \approx 22$ is rapidly removed. A major drawback of the regularisation method as introduced in equation (3.1) becomes apparent when looking at positions of a signal where an edge is present in the original signal f. Figure 9 illustrates such a situation together with the result of filtering it. The problem is that if an edge – or position where the signal has a large derivative – is filtered, the smoothness term will penalise such constellations severely and suppress them. An idea how to overcome this problem of *over-penalisation* is to use other penalising functions than the quadratic one used in equation (3.1). By introducing a function $\Psi : \mathbb{R} \to \mathbb{R}$ as

$$\Psi(s^2) := \sqrt{s^2 + \epsilon^2} \tag{3.10}$$

which is a regularised variant [1] of the total variation regularisation framework [46, 21], we can impose a sub-quadratic process which approximates

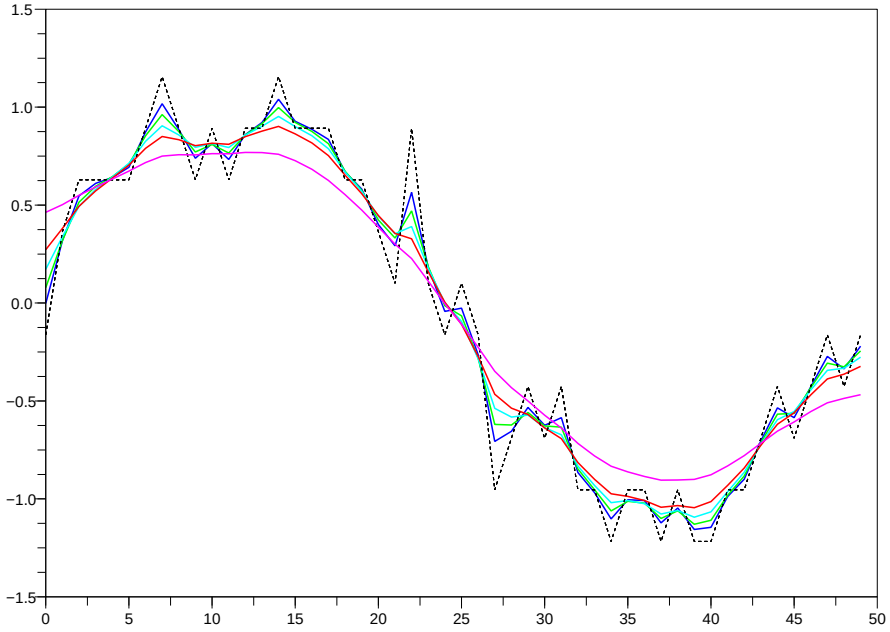


Figure 8: 1-D quadratic signal regularisation with different choices for the regularisation parameter α. The dashed line denotes the input signal from figure 7. **Blue line:** Filtered signal with $\alpha = 0.5$. **Green line:** $\alpha = 1$. **Cyan line:** $\alpha = 2$. **Red line:** $\alpha = 4$. **Violet line:** $\alpha = 16$. All variational compuations have been realised in Scilab [22].

an L^1 penalisation. The parameter $\epsilon \approx 0.01$ is chosen very small and ensures the differentiability of Ψ in $s = 0$. Using this function, the following continuous energy functional arises

$$E_{TV}(u) = \frac{1}{2}\int_\Omega (u-f)^2 \,\mathrm{d}x + \alpha \int_\Omega \Psi\left(\left(\tfrac{\mathrm{d}}{\mathrm{d}x}u\right)^2\right)\mathrm{d}x \tag{3.11}$$

Again, instead of deriving the Euler-Lagrange equations and discretising them in order to get a numerical scheme, we want to discretise the given energy functional directly. As in the previous case, in order to get simple and immediately comprehensible terms we stick to one-sided finite (forward) difference methods to discretise the derivatives. Then, a discretised version of 3.11 reads

$$E_{TV}(\mathbf{u}) = \frac{1}{2}\sum_{i=1}^{n}(u_i - f_i)^2 + \frac{\alpha}{2}\sum_{i=1}^{n-1}\Psi\left((u_{i+1}-u_i)^2\right) \tag{3.12}$$

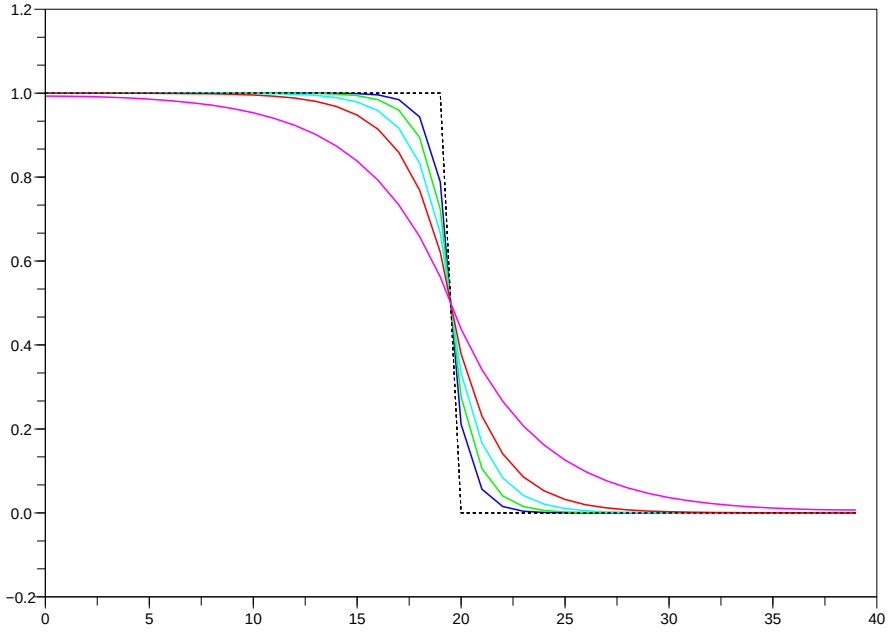


Figure 9: Variational quadratic regularisation of a step function. One can observe that the discontinuity at $x = 20$ is removed with increasing smoothness parameter α. The dashed line shows the original step function. **Blue line:** Filtered signal with $\alpha = 0.5$. **Green line:** $\alpha = 1$. **Cyan line:** $\alpha = 2$. **Red line:** $\alpha = 4$. **Violet line:** $\alpha = 16$.

One can easily prove the convexity of Ψ and with this result also the convexity of the whole energy function $E_{TV}(\mathbf{u})$ follows. Hence, a vector that fulfils $\nabla E_{TV} = 0$ must be a minimiser of the functional. Thus we know that the partial derivatives of E_{TV} must vanish. Abbreviating $\psi_i := \Psi'\left((u_{i+1} - u_i)^2\right)$, the derivatives are given by

$$\begin{aligned} 0 &\stackrel{!}{=} u_1 - f_1 - \alpha\psi_1(u_2 - u_1) \\ 0 &\stackrel{!}{=} u_i - f_i + \alpha\left(\psi_{i-1}(u_i - u_{i-1}) - \psi_i(u_{i+1} - u_i)\right) \quad {\scriptstyle (i=2,\ldots,n-1)} \\ 0 &\stackrel{!}{=} u_n - f_n + \alpha\psi_{n-1}(u_n - u_{n-1}) \end{aligned} \tag{3.13}$$

These equations define a non-linear system of equations similar to the system from (3.8). Note that the abbreviating terms ψ_i are the non-linear parts and steer the rate of diffusion *pointwise*. The system matrix is given

as

$$A_{TV} := \begin{pmatrix} 1+\alpha\psi_1 & -\alpha\psi_1 & & & \\ -\alpha\psi_1 & 1+\alpha(\psi_1+\psi_2) & -\alpha\psi_2 & & \\ & \ddots & \ddots & \ddots & \\ & -\alpha\psi_{n-1} & 1+\alpha(\psi_{n-1}+\psi_n) & -\alpha\psi_n & \\ & & -\alpha\psi_n & 1+\alpha\psi_n & \end{pmatrix} \tag{3.14}$$

As in (3.8), the system that has to be solved reads

$$A_{TV}\mathbf{u} = \mathbf{f} \tag{3.15}$$

As before, the strict positive definiteness of A_{TV} can easily be shown. All non-linear terms fulfil $\psi_i > 0$. Thus, A_{TV} is strictly diagonally dominant and by that invertible. In contrast to the system for quadratic regularisation, we need to iterate a procedure like

$$\mathbf{u}^{k+1} = \left(A_{TV}^k\right)^{-1}\mathbf{f}$$

for the actual system, where we have to refresh the non-linear terms in A_{TV}^k after each iteration with the updated values of $\mathbf{u}^{k+1}$. This proceeding is also known under the name of time-lagged diffusivity or Kačanov method.

Results for this variational approach applied to the simple step function are given in figure 10. One can clearly see that the discontinuity survives and is not smoothed out. Also applied to more complex signals such as the signal from figure 7, the introduction of the total variation term clearly influences the filtering results, as can be observed in figure 11.

After having introduced briefly variational regularisation strategies together with classical minimisation methods in the past section of this thesis, we want to come now to a closely related class of regularisation methods, using a different minimisation procedure, namely graph cuts.

3.1 Graph Cuts in Signal Regularisation

In this section we want to document our first attempts towards graph cut methods. The reason to start with 1-D image regularisation is that the corresponding graphs have a very simple structure and can be constructed easily.

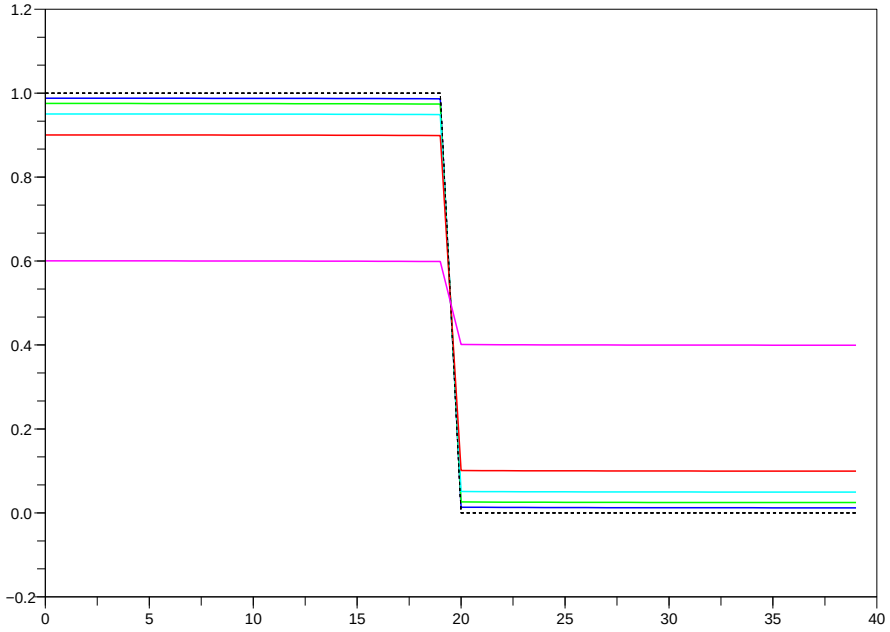


Figure 10: Sub-quadratic variational regularisation of a step function. The steep discontinuity at $x = 20$ is preserved also with increasing smoothness parameter α. The original signal is invisible since it is covered by the filtered versions. **Blue line:** Filtered signal with $\alpha = 0.5$. **Green line:** $\alpha = 1$. **Cyan line:** $\alpha = 2$. **Red line:** $\alpha = 4$. **Violet line:** $\alpha = 16$.

The main idea behind graph cuts will now be to construct an appropriate graph that resembles the image situation. This graph should be built in such a way that a cut of this graph should induce a new labelling. Then we can rely on graph theory, which says that if we can find a minimum cut, then the corresponding labelling whould also be a minimum of our energy function $E(f)$.

One of the main differences between classical methods such as the variational method discussed in the previous section and graph cuts is that graph cuts work in an absolutely discrete world. Solving the system from equation (3.14), we could determine a solution vector $\mathbf{u} \in \mathbb{R}^n$. Using graph cuts, we will only be able to determine a solution vector having one label of a finite set of available labels $\mathcal{L} = \{f_1, \ldots, f_l\}$ in each component, hence a solution by graph cuts would be $\mathbf{u} \in \mathcal{L}^n$. Note that already in the variational setting the spatial axis was discretised. But still, the function values were

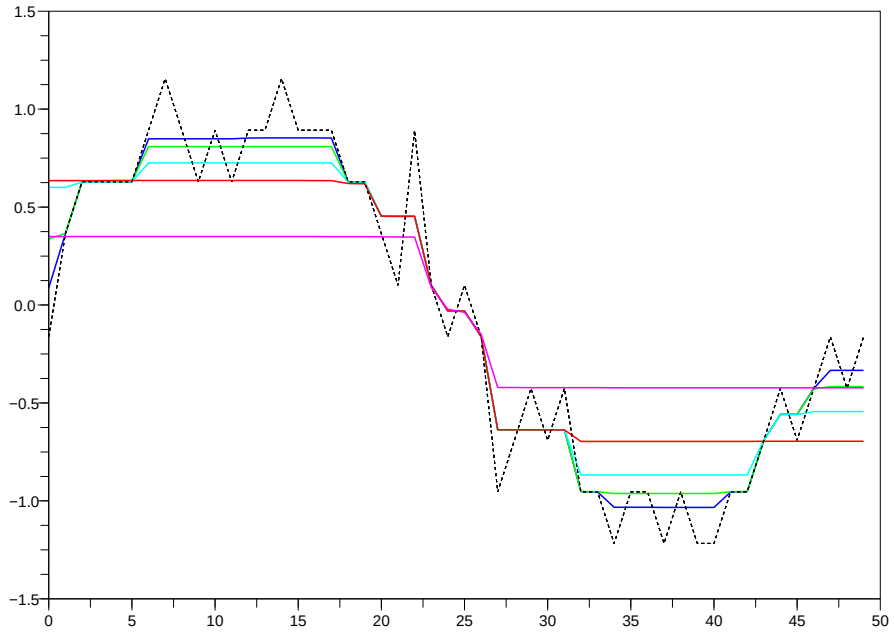


Figure 11: Regularised total variation regularisation of the signal from figure 7. Also in this case one can see that edges are not rounded as in the quadratic case. **Blue line:** Filtered signal with $\alpha = 0.5$. **Green line:** $\alpha = 1$. **Cyan line:** $\alpha = 2$. **Red line:** $\alpha = 4$. **Violet line:** $\alpha = 16$.

chosen from the continuous space. Now, also the space of function values will be finite. For the sake of completeness, table 1 give the quantisation values corresponding to the (equidistant) labels of $\mathcal{L}$ for the signal being used throughout the section.

Thus, we will frequently use the notion of a labelling f, in correspondence to a solution vector, which assigns to each pixel $p \in \mathcal{P}$ a label $f_p \in \mathcal{L}$.

As for the variational approach, for graph cuts we will set up and minimise a similar energy function E consisting of a data term as well as a smoothness term. This function reads

$$E(f) = E_{\text{data}}(f) + E_{\text{smooth}}(f) \tag{3.16}$$

Similarly to the data terms introduced for the variational methods, the data

label	value
l_1	-1.2172
l_2	-0.9535
l_3	-0.6898
l_4	-0.4261
l_5	-0.1624
l_6	0.1012
l_7	0.3649
l_8	0.6286
l_9	0.8922
l_{10}	1.1559

Table 1: Label set $\mathcal{L}$ belonging to the function from figure 7. Note that for image regularisation the elements of the label set $\mathcal{L}$ coincide with the quantisation of the signal.

term here will have the form

$$E_{\text{data}}(f) = \sum_{p \in \mathcal{P}} D_p(f_p) \tag{3.17}$$

where as data penaliser D_p usually a quadratic term is chosen

$$D_p(f_p) = (f_p - I_p)^2 \tag{3.18}$$

and where I_p represents the intensity of the input signal (or image) at pixel p. The choice of the smoothness term is a more complicated issue, since as in the variational setting, many different functions have been proposed. Generally speaking, the smoothness term will resemble inter-pixel dependencies and can be formulated as

$$E_{\text{smooth}}(f) = \sum_{p,q \in \mathcal{N}} V_{p,q}(f_p, f_q) \tag{3.19}$$

where $\mathcal{N}$ is the neighbourhood set. Note already here, that this formulation only allows for the interaction of one pixel with one other pixel. Higher order terms such as the interaction of a clique of three pixels are possible [65] but go beyond the scope of this introductory discussion of graph cuts. Concerning the smoothness term, due to its central importance, we will devote the upcoming section 4 entirely to different possible choices for this term. In this section, we will try to establish a direct comparison of variational

ideas with graph cuts. Therefore, we will examine two choices for $V_{p,q}$, a quadratic one resembling quadratic image regularisation

$$V_{\text{quadr}}(f_p, f_q) := \frac{\alpha}{2}(f_p - f_q)^2 \tag{3.20}$$

as well as a linear one resembling total variation regularisation and reads

$$V_{\text{TV}}(f_p, f_q) := \frac{\alpha}{2}|f_p - f_q| \tag{3.21}$$

Note that since graph cut methods only need to evaluate these terms, and not to differentiate them, it is not necessary to introduce an parameter ϵ as in equation (3.10). Another big advantage we will detail on later, is that replacing one smoothness term with an other when using graph cut methods can be done without any effort, which is most often not the case for variational methods.

3.2 Graph Construction

We now want to construct a graph in which we want to resemble our energy function. Note that the exact determination of the global minimum of functions such as in equation (3.16) is an NP-hard problem [13, 24]. Thus the only thing we can do is approximating the global minimum with a hopefully good local minimum. For this task, graph cuts provide us with powerful means.

3.2.1 Moves

In this context, so-called *moves* play an important role. Classical methods like e.g. simulated annealing [37] perform what we call *standard* moves, whose property is that only one pixel changes at a time. In contrast, we will introduce two types of more powerful moves, where many pixels can change at a time. The notion is also of central importance for the definition of a local minimum: Only using standard moves, an algorithm reaches a local minimum, if the energy cannot be decreased by changing any pixel value. For our moves, if the energy cannot be decreased any more even by adjusting many pixels at a time, then also the quality of this local minimum should rise.

Given a current labelling $f \in \mathcal{L}^n$, this induces a partitioning of the pixels by their assigned label. Thus we can define a partition of the image pixels as $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ represents the set of all

pixels being currently assigned to label l. This equivalence also holds vice versa: A given partition uniquely induces a labelling. Therefore we can use the two notions interchangingly.

In the following sections we will introduce two types of moves: *swap*-moves and *expansion*-moves.

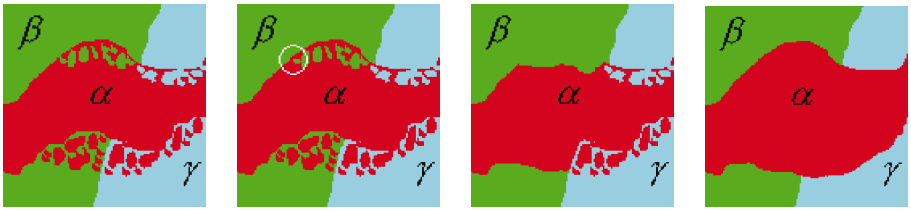
In a swap move we will always consider two labels α and β, and all pixels being currently assigned to either α or β. Then, a swap move can transform a partition $\mathbf{P}$ into a new partition $\mathbf{P}'$ if it holds that $P_l = P'_l$ for all $l \neq \alpha, \beta$. We will abbreviate the set $\mathcal{P}_\alpha \cup \mathcal{P}_\beta$ with $\mathcal{P}_{\alpha\beta}$ in the following. So, by a swap move, only pixels from $\mathcal{P}_\alpha$ can swap their assignment to β (or keep their current assignment), or elements in $\mathcal{P}_\beta$ can change to $\mathcal{P}_\alpha$ (or also keep their β-assignment). Given a partition $\mathbf{P}$ and two labels α and β, we say that a partition $\mathbf{P}'$ is one α-β-swap away from $\mathbf{P}$ if the above requirements are fulfilled.

An expansion move considers all pixels being currently assigned to one label α in contrast to all other pixels, being not assigned to this label α. Then one partition $\mathbf{P}$ can be transformed into a new partition $\mathbf{P}'$ by one expansion move for label α, if $\mathcal{P}_\alpha \subset \mathcal{P}'_\alpha$ and $\mathcal{P}'_l \subset \mathcal{P}_l$ for any $l \neq \alpha$. Intuitively, this means that only more pixels can be assigned to α, i.e. $\mathcal{P}_\alpha$ cannot get smaller. All pixels that were already assigned to label α must keep this assignment and only pixels that were assigned to other labels can change to the expanded label. As for swap moves, we define a partition $\mathbf{P}'$ to be one α-expansion move away from a partition $\mathbf{P}$, if the given requirements hold. We want to refer to figure 12 for an example for the different moves here.

3.2.2 Swap moves

As described, given an input labelling f and two labels α and β, an α-β-swap move should find a labelling $\hat{f}$, being the minimum of the energy function $E(f)$ for this α-β-swap.

Once having such swap moves at hand we can then define an algorithm to approximate the global minimum of the energy function from equation (3.16) in listing 4. We do not want to conceal here, that the algorithm as given here has some negative theoretical properties. Especially the order in which the pairs of labels are processed has a big impact on how the labelling is updated. Ideally, in each cylce one would go through all pairs of labels and update the labelling not until after all pairs have been seen. One would take the swap move providing the highest decrease of energy then. In this



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

Figure 12: Illustration of different moves. Image (a) is an input image with three labels α, β and γ. The second image depicts a standard move, where only one pixel can be changed in each step. The image (c) shows the possible result of a swap move of the labels α and β. Note that $\mathcal{P}_\gamma$ has not changed. The last image (d) illustrates a expansion move of label α. All pixels that were already assigned to alpha keep this assignment, only other pixels can change to α. Images courtesy [13].

way, the order of processing the label pairs would not be an issue any more. But the algorithm would have to perform $\mathcal{O}(n^2)$ swap moves for each single applied swap move, which would lead to a drastical increase of computation effort. Empirically, we did not find an increase of quality using the latter algorithm strategy, therefore we took the algorithm as introduced in listing 4 by [13] altough it is ordering dependent in theory.

We will now detail on the still missing central part of the whole algorithm which is denoted by **swap-move** (α,β,f) in listing 4.

Given the two labels α and β and the labelling f, we will construct a graph $\mathcal{G}_{\alpha\beta} =< \mathcal{V}_{\alpha\beta}, \mathcal{E}_{\alpha\beta} >$. This graph will have two terminal nodes s and t corresponding to α and β, respectively. Further we will see that a cut through this graph separating the two terminals will uniquely specify a new labelling $\hat{f}$.

Given the construction, we will show certain properties of this construction. Especially, we will show that the labelling induced by a minimum cut of $\mathcal{G}_{\alpha\beta}$ must be the desired minimum of our energy function.

The construction of the graph is straightforward. The set of vertices will consist of all pixels being assigned either to α or β and the two terminal nodes: $\mathcal{V}_{\alpha\beta} = \mathcal{P}_\alpha \cup \mathcal{P}_\beta \cup \{s,t\} = \mathcal{P}_{\alpha\beta} \cup \{s,t\}$. Each pixel vertex will share an edge with each of the terminals s and t, such a terminal-link connecting a pixel p with s will be denoted by t_p^α and similarly for the sink t by t_p^β. Moreover, there will be an edge between any two nodes from $\mathcal{P}_\alpha \cup \mathcal{P}_\beta$ that

```
Step 1: Input labelling f
Step 2: success := false
Step 3: for all pairs of labels {α,β} ⊂ L × L do
            f̂ = swap–move ( α, β, f )
            if (E(f') < E(f)) then
                f = f'
                success = true
            end
        end
Step 4: if (success) then
            goto Step 2
        else
            Return f
        end
```

Listing 4: Swap move algorithm according to [13].

are direct spatial neighbours. The assigned weights for these edges can be looked up in table 2.

edge	weight	for
$t_p^\alpha = e_{[s,p]}$	$D_p(\alpha) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha\beta}}} V(\alpha, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
$t_p^\beta = e_{[p,t]}$	$D_p(\beta) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha\beta}}} V(\beta, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
$e_{[p,q]}$	$V(\alpha, \beta)$	$\substack{p,q \in \mathcal{N} \\ p,q \in \mathcal{P}_{\alpha\beta}}$

Table 2: Edge weights for swap move graph construction. All edges are symmetric and undirected.

We will refer to all edges corresponding to the first two lines of table 2 as t-links, since they connect the terminals to the pixels. Similarly we will call the set of all edges satisfying the last line of table 2 as the n-links, because they describe the neighbourhood relations.

Obviously, any cut that separates the two terminals must include one t-link for each pixel. If this was not the case, then there would be a path from s to t and hence the cut would not separate the terminals. This fundamental

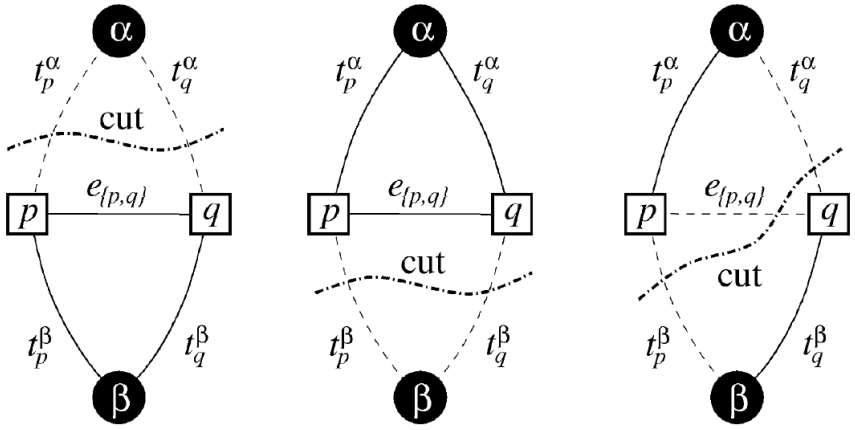


Figure 13: Illustration on how two neighbouring pixels can be cut (dashed line). **Left:** t^α-link for p and q is cut. Both nodes will be assigned to label α. Since they have the same label, there is no contribution of the smoothness term, whose edge $e_{[p,q]}$ is not part of the cut. **Middle:** Same as the first case, just with label and terminal β. **Right:** p is assigned to label β and q is assigned α. Since the label changes, also the n-link between them is cut. Illustrations from [13].

property is the key how to relate a labelling uniquely to a cut through a graph: Each pixel is assigned that label, whose link is part of the cut. Given a graph $\mathcal{G}_{\alpha\beta} =< \mathcal{V}_{\alpha\beta}, \mathcal{E}_{\alpha\beta} >$ and an associated cut $\mathcal{C} \subset \mathcal{E}_{\alpha\beta}$ we can derive the corresponding labelling $f^{\mathcal{C}}$ as

$$f_p^{\mathcal{C}} = \begin{cases} \alpha, & \text{if } t_p^\alpha \in \mathcal{C} \quad \wedge \quad p \in \mathcal{P}_{\alpha\beta} \\ \beta, & \text{if } t_p^\beta \in \mathcal{C} \quad \wedge \quad p \in \mathcal{P}_{\alpha\beta} \\ f_p, & \text{if} \qquad\qquad\quad p \notin \mathcal{P}_{\alpha\beta} \end{cases} \tag{3.22}$$

In order to proof that the capacity of a cut on $\mathcal{G}_{\alpha\beta}$ coincides to the energy of the induced labelling, we need to state some lemmas. These lemmas are illustrated best by figure 13 which explains all possibilities how to separate the terminals. Regarding the neighbourhood links, by construction, we can state that an n-link can only be part of a cut, if different t-links of the two concerned p-nodes are cut. Formally we can state for all $p, q \in P_{\alpha\beta}$ and

$\{p,q\} \in \mathcal{N}$ that

$$\begin{array}{llll} \text{if} & t_p^\alpha, t_q^\alpha \in \mathcal{C} & \text{then} & e_{[p,q]} \notin \mathcal{C} \\ \text{if} & t_p^\beta, t_q^\beta \in \mathcal{C} & \text{then} & e_{[p,q]} \notin \mathcal{C} \\ \text{if} & t_p^\alpha, t_q^\beta \in \mathcal{C} & \text{then} & e_{[p,q]} \in \mathcal{C} \\ \text{if} & t_p^\beta, t_q^\alpha \in \mathcal{C} & \text{then} & e_{[p,q]} \in \mathcal{C} \end{array} \tag{3.23}$$

The first two lines of the latter equation follow from the fact that only one t-link is part of a cut for each pixel. If this was not the case, then there existed a subset of $\mathcal{C}$ which was still a cut, which cannot be the case. The last two lines must also hold, since a cut must separate the terminals. Since in these cases the two nodes will be connected to different terminals in the induced graph, also the link connecting the two nodes must be eliminated.

Now we can summarise equations (3.22) and (3.23) to one compact statement

$$|\mathcal{C} \cap e_{[p,q]}| = V(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}) \tag{3.24}$$

Note that for a strict proof of this equation, we need V to be a semi-metric, which is true for our case here. See section 4 for a deep discussion. Now we have everything at hand to state and proof the following

Theorem The cost of a cut $\mathcal{C}$ on $\mathcal{G}_{\alpha\beta}$ equals the energy of its induced labelling plus a constant: $|\mathcal{C}| + K = E(f^{\mathcal{C}})$.

Proof For each pixel node, one t-link must be cut. Thus, the cost of the cut is

$$|\mathcal{C}| = \sum_{p \in \mathcal{P}_{\alpha\beta}} |\mathcal{C} \cap \{t_p^\alpha, t_p^\beta\}| + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ p,q \in \mathcal{P}_{\alpha\beta}}} |\mathcal{C} \cap e_{[p,q]}| \tag{3.25}$$

From table 2 and the fact that $p \in \mathcal{P}_{\alpha\beta}$ we can directly use

$$|\mathcal{C} \cap \{t_p^\alpha, t_p^\beta\}| = \begin{cases} |t_p^\alpha| & \text{if } t_p^\alpha \in \mathcal{C} \\ |t_p^\beta| & \text{if } t_p^\beta \in \mathcal{C} \end{cases} = D_p(f_p^{\mathcal{C}}) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha\beta}}} V(f_p^{\mathcal{C}}, f_q) \tag{3.26}$$

Combining the latter equality together with equation (3.24) we can rewrite the total cost of a cut as

$$\begin{aligned} |\mathcal{C}| &= \sum_{p \in \mathcal{P}_{\alpha\beta}} D_p(f_p^{\mathcal{C}}) + \sum_{p \in \mathcal{P}_{\alpha\beta}} \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha\beta}}} V(f_p^{\mathcal{C}}, f_q) + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ p,q \in \mathcal{P}_{\alpha\beta}}} V(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}) \\ &= \sum_{p \in \mathcal{P}_{\alpha\beta}} D_p(f_p^{\mathcal{C}}) + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ \{p,q\} \cap \mathcal{P}_{\alpha\beta} \neq \emptyset}} V(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}) \end{aligned} \tag{3.27}$$

The energy contribution missing in the last equation are just

$$K = \sum_{p \notin \mathcal{P}_{\alpha\beta}} D_p(f_p) + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ \{p,q\} \cap \mathcal{P}_{\alpha\beta} = \emptyset}} V(f_p, f_q) \tag{3.28}$$

This constant K has the same value for all swap moves for given labels α and β.

Thus, we can conclude that using this graph construction, we can shift the problem of finding the minimum of a certain energy function to determining a minimal cut on an associated flow network. Once this minimal cut is found, the former theorem shows that also the (local) minimum of the function is found.

The following figures 14 and 15 show results obtained by filtering the signal from figure 7 using graph cut methods. Then, in figures 16 and 17 we directly compare results obtained by graph cuts and variational regularisation methods. For all computations of the swap moves, we used the algorithm depicted in listing 4. As initial labelling we used a labelling having label zero everywhere. We also tried other initial labellings, but the method has rendered to be insensitive with respect to initialisation: Using e.g. a random initial labelling, only about 1% of the pixels in the final labeling changed. Also the order in which the pairs of labels are processed play a role. For all tests we took a regular processing order, passing through all pairs by going row by row through the following table, from left to right, top to bottom.

$$\begin{array}{ccccc} \square & (1,2) & (1,3) & \dots & (1,n) \\ & \square & (2,3) & \dots & (2,n) \\ & & \ddots & & \vdots \\ & & & \square & (n-1,n) \end{array} \tag{3.29}$$

3.2.3 Expansion moves

Besides the swap moves introduced in the last section, there also exist so-called *expansion moves* acting slightly differently. A main restriction we already have to stress here is that in order to approximate our energy function and to really find a minimum, we have to ensure that the smoothness

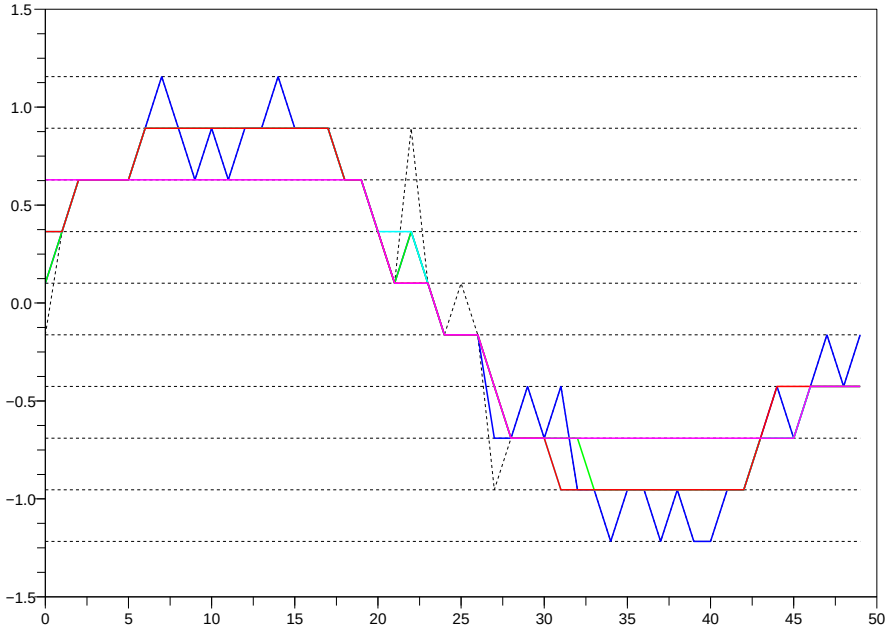


Figure 14: Graph cut based 1-D image regularisation. Quadratic smoothness term from equation (3.20) used. Only swap moves used. **Blue line:** Filtered signal with $\alpha = 0.5$. **Green line:** $\alpha = 1$. **Cyan line:** $\alpha = 2$. **Red line:** $\alpha = 4$. **Violet line:** $\alpha = 16$.

term V fulfils an additional constraint. It must be what we call a *metric*, see section 4 for details on this topic. For the discussion of 1-D image regularisation it is sufficient to know that the quadratic smoothness term $V_{\text{quadratic}}$ from equation (3.20) does not fulfil all necessary requirements for being a metric, it is just a *semi-metric*. But the linear model from equation (3.21) satisfies all constraints and can hence be used to regularise our data using expansion moves.

Similarly to the last section, also for expansion moves we need to find an algorithm, that defines how and when to apply the new minimisation move. This is done in listing 5.

In an α-expansion move we will consider all pixels, no matter which label they are currently assigned to. The minimisation process for the move then considers for each pixel whether it is cheaper to have label α or to keep its actual label. Obviously, pixels that were assigned to α before the move will always keep this assignment. This means that the number of pixels having

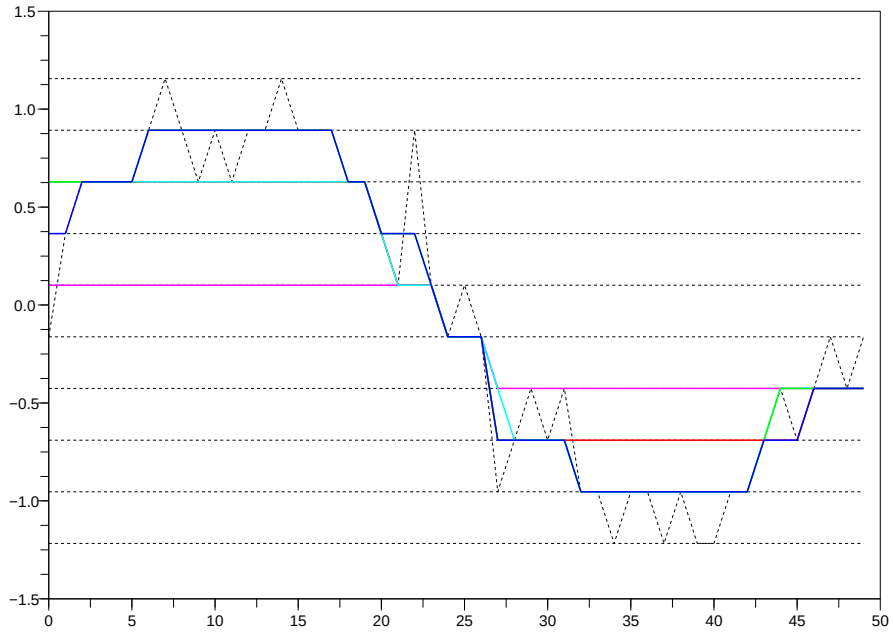


Figure 15: Graph cut based 1-D image regularisation. Linear smoothness term from equation (3.21) used. Only swap moves used. **Blue line:** Filtered signal with $\alpha = 0.5$. **Green line:** $\alpha = 1$. **Cyan line:** $\alpha = 2$. **Red line:** $\alpha = 4$. **Violet line:** $\alpha = 16$.

label α can only grow by an α-expansion move: $|\mathbf{P'}_\alpha| \geq |\mathbf{P}_\alpha|$.

The graph construction for expansion moves is slightly different from the construction for α-β-moves. Instead of $|\mathcal{P}_\alpha| + |\mathcal{P}_\beta| + 2$ nodes, this construction $\mathcal{G}_\alpha =< \mathcal{V}_\alpha, \mathcal{E}_\alpha >$ will have one vertex per pixel, independently of the number of current α-pixels $|\mathcal{P}_\alpha|$. The sink and the source node will correspond to the label α and the opposite label *"not α"*, respectively. Besides the vertices corresponding to pixels, we will need additional auxiliary nodes whenever two neighbouring pixels are assigned to different labels.

To write the set of vertices down formally, as described we get

$$\mathcal{V}_\alpha = \left\{ \alpha, \bar{\alpha}, \mathcal{P}, \bigcup_{\substack{\{p,q\}\in\mathcal{N} \\ f_p \neq f_q}} a_{[pq]} \right\}$$

where $a_{[pq]}$ shall denote the auxiliary node between the pixel nodes p and q.

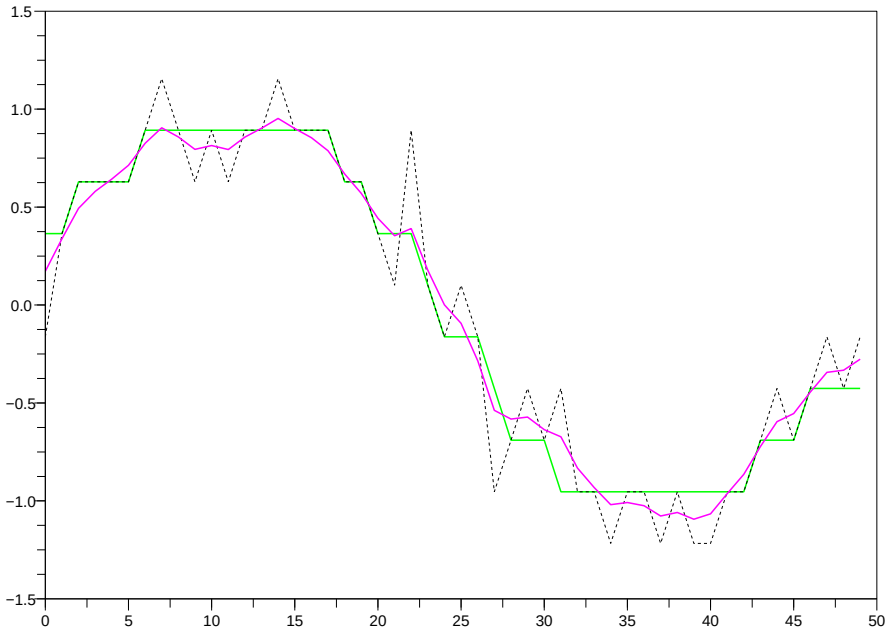


Figure 16: Comparison between graph cut based 1-D image regularisation and variational ideas. Only swap moves used. Quadratic smoothness term used. The regularisation parameter was for both methods $\alpha = 2$. One can see that the results are very similar. Also one should note that the quantisation of the graph cut method is not happening for the variational ansatz. **Violet line:** Variational method. **Green line:** Graph cut method.

Concerning the edges of this expansion graph, as before, each pixel node will be connected to the sink as well as to the source, representing the link to α and $\bar{\alpha}$, respectively. These links will be denoted by t_p^{α} and $t_p^{\bar{\alpha}}$, respectively.

If present, an auxiliary node will be connected to the label $\bar{\alpha}$ only, the link to the source is not inserted.

The inter-pixel links are similar to the swap graph construction: Any two spatial neighbours will be connected by an edge $e_{[pq]}$ if they have the same current label. If their labels differ, instead of being directly connected, they will be connected to their auxiliary node.

Closely resembling the proceeding from [13], we can then write the set

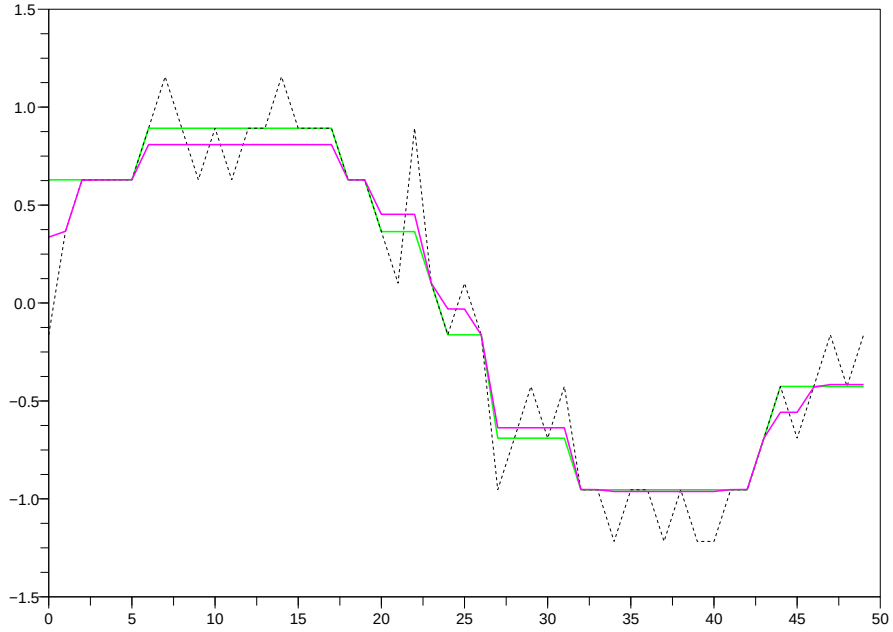


Figure 17: Comparison between graph cut based 1-D image regularisation and variational ideas. Only swap moves used. Linear (TV) smoothness term used. The regularisation parameter was for both methods $\alpha = 1$. One can see that the results are very similar. **Violet line:** Variational method. **Green line:** Graph cut method.

of edges as

$$\mathcal{E}_\alpha = \left\{ \bigcup_{p\in\mathcal{P}} \{t_p^\alpha, t_p^{\bar{\alpha}}\}, \bigcup_{\substack{\{p,q\}\in\mathcal{N} \\ f_p \neq f_q}} \mathcal{E}_{[pq]}, \bigcup_{\substack{\{p,q\}\in\mathcal{N} \\ f_p = f_q}} e_{[pq]} \right\}$$

where $\mathcal{E}_{[p,q]} = \{e_{[pa]}, e_{[aq]}, t_a^{\bar{\alpha}}\}$ resembles the case when an auxiliary node a is inserted. The weights for the edges of this graph construction are given in table 3. Again, this graph construction incorporates two terminal nodes, that will have to be separated by a valid cut through this graph. Having a closer look on such a construction, one can recognise that any cut has to sever either the t_p^α or the $t_p^{\bar{\alpha}}$ edge. This property gives the criterion how to read the induced labelling from the cutted graph. We read the labelling

```
Step 1: Input labelling f
Step 2: success := false
Step 3: for all labels α ∈ L do
            f̂ = expansion−move ( α, f )
            if (E(f′) < E(f)) then
                f = f′
                success = true
            end
        end
Step 4: if (success) then
            goto Step 2
        else
            Return f
        end
```

Listing 5: Expansion move algorithm according to [13].

defined by the cut $\mathcal{C} \subset \mathcal{E}_\alpha$ on $\mathcal{G}_\alpha$ as

$$f^{\mathcal{C}} = \begin{cases} \alpha & t_p^\alpha \in \mathcal{C} \\ f_p & t_p^{\bar{\alpha}} \in \mathcal{C} \end{cases} \qquad \forall p \in \mathcal{P} \tag{3.30}$$

This means that if the edge from the pixel to the source (which represents the α-node) is cut, then this pixel will be labelled α. For pixels having already been mapped to α, the construction ensures that such a pixel will keep this assignment by setting the capacity of the edge to $\bar{\alpha}$ to infinity. By that, a cut using this edge would have an infinite capacity. Since we are searching minimal cuts, the algorithm can be sure to not choose this edge.

Similarly to the proceeding in the last section, we will now give and proof several properties of this graph construction, leading finally to the proof that the capacity of cuts on $\mathcal{G}_\alpha$ can be related linearly to the energy of corresponding labellings. Thus we will see as for the swap moves, that an accordingly constructed graph together with a minimal cut on it, can efficiently realise the approximation of the minimum of the energy function.

First, concerning the neighbourhood links we have to distinguish the case when $f_p = f_q$ and when $f_p \neq f_q$. In the first situation, by (3.30) we can be sure that the edge $e_{[pq]}$ is cut if and only if p and q are connected to different terminals in the induced graph $\mathcal{G}_\alpha^{\mathcal{C}} =< \mathcal{V}_\alpha, \mathcal{E}_\alpha \setminus \mathcal{C} >$. This can

edge	weight	for
$t_p^{\bar{\alpha}}$	∞	$p \in \mathcal{P}_\alpha$
$t_p^{\bar{\alpha}}$	$D_p(f_p)$	$p \notin \mathcal{P}_\alpha$
t_p^{α}	$D_p(\alpha)$	$p \in \mathcal{P}$
$e_{[pa]}$	$V(f_p, \alpha)$	$\{p,q\} \in \mathcal{N}\ ,\ f_p \neq f_q$
$e_{[aq]}$	$V(\alpha, f_q)$	
$t_a^{\bar{\alpha}}$	$V(f_p, f_q)$	
$e_{[pq]}$	$V(f_p, \alpha)$	$\{p,q\} \in \mathcal{N}\ ,\ f_p = f_q$

Table 3: Edge weights for expansion move graph construction. All edges are symmetric and undirected

be related closely to the swap moves and equation (3.23), by substituting β by $\bar{\alpha}$. Analogously we can then also get the result (3.24) for the expansion graph construction. For $f_p = f_q$ we get

$$|\mathcal{C} \cap e_{[p,q]}| = V(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}) \tag{3.31}$$

The second case, where $f_p \neq f_q$ is slightly different. In this case we defined the set of edges $\mathcal{E}_{[pq]}$. There are several possibilities how to cut these edges, depending on which terminal links for the corresponding pixel nodes p and q are cut. Assuming that the node $a = a_{[pq]}$ is the corresponding auxiliary node for the two neighbour nodes p and q, and that the cut is a minimal cut, we can state the following

$$\begin{array}{llllll} (a) & \text{if} & t_p^{\alpha}, t_q^{\alpha} \in \mathcal{C} & \text{then} & \mathcal{C} \cap \mathcal{E}_{[pq]} = \emptyset \\ (b) & \text{if} & t_p^{\bar{\alpha}}, t_q^{\bar{\alpha}} \in \mathcal{C} & \text{then} & \mathcal{C} \cap \mathcal{E}_{[pq]} = t_a^{\bar{\alpha}} \\ (c) & \text{if} & t_p^{\bar{\alpha}}, t_q^{\alpha} \in \mathcal{C} & \text{then} & \mathcal{C} \cap \mathcal{E}_{[pq]} = e_{[pa]} \\ (d) & \text{if} & t_p^{\alpha}, t_q^{\bar{\alpha}} \in \mathcal{C} & \text{then} & \mathcal{C} \cap \mathcal{E}_{[pq]} = e_{[aq]} \end{array} \tag{3.32}$$

The described cases are illustrated in figure 18 to clarify their validity. The first part of (3.32) follows just from the fact that a cut is only a cut if no proper subset of it would also separate the terminals, cutting both α links is sufficient for the separation. The latter parts (b), (c) and (d) can be proven using the properties of the smoothness penaliser V. Since we required V to be a metric, we know that it fulfils the triangle inequality (see section 4). We prove part (b) exemplarily. If both $\bar{\alpha}$-links are cut, we have exactly two

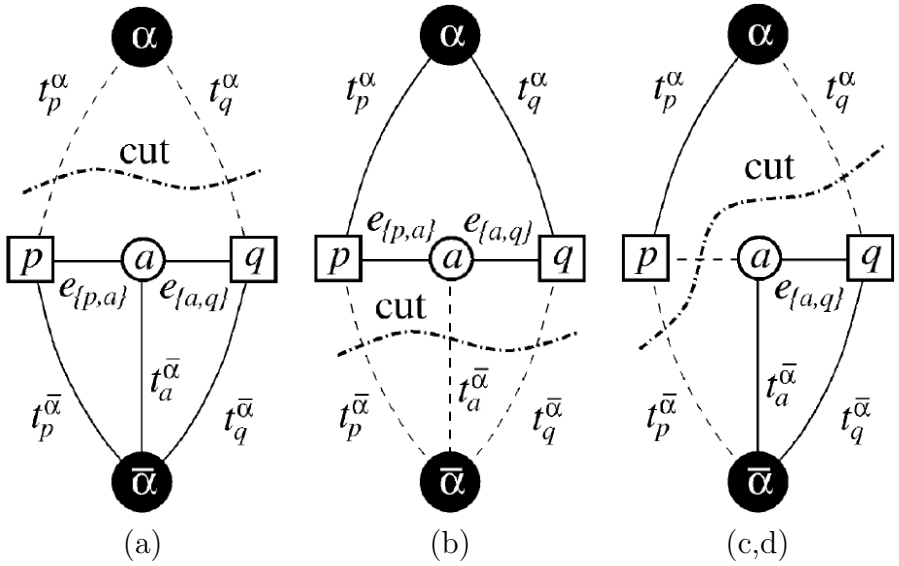


Figure 18: Illustration of the four possible cases how to cut $\mathcal{E}_{[pq]}$. Note that $f_p \neq f_q$ in this case and that $\{p, q\} \in \mathcal{N}$.

possibilities of cutting $\mathcal{E}_{[pq]}$. Either $\mathcal{C} \cap \mathcal{E}_{[pq]} = t_a^{\bar\alpha}$ or $\mathcal{C} \cap \mathcal{E}_{[pq]} = \{e_{[pa]}, e_{[aq]}\}$. Since V is metric and fulfils the mentioned triangle inequality, we know that the following inequality holds:

$$\underbrace{V(f_p, f_q)}_{|t_a^{\bar\alpha}|} \leq \underbrace{V(f_p, \alpha)}_{|e_{[pa]}|} + \underbrace{V(\alpha, f_q)}_{|e_{[aq]}|} \tag{3.33}$$

Thus, a minimum cut on $\mathcal{G}_\alpha$ must fulfil (3.32). Combining the results of (3.30) and (3.32) in one statement, we can write for $f_p \neq f_q$

$$|\mathcal{C} \cap \mathcal{E}_{[pq]}| = V(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}) \tag{3.34}$$

As described and analogously to the swap move construction, we were able to state that property (3.31) holds for any cut on $\mathcal{G}_\alpha$. For the last statement (3.34) we needed that the cut is a minimal cut. But generally speaking, in order to get that properties (3.32) hold, it is not absolutely necessary to have a minimum cut. Relaxing these requirements, and just demanding that (3.32) is satisfied by a cut, we can define *elementary* cuts on $\mathcal{G}_\alpha$, for which we can now prove the following

Theorem For a flow network $\mathcal{G}_\alpha$ according to the above description and given f and α, there exists a one to one correspondence between elementary

cuts on $\mathcal{G}_\alpha$ and labellings within one α-expansion of f. Further, for any elementary cut $\mathcal{C}$ we have

$$|\mathcal{C}| = E(F^{\mathcal{C}}).$$

Proof Given a cut, (3.30) defined how to read out a labelling of it. Hence what still has to be proven is the other direction, given a labelling $f^{\mathcal{C}}$ determine the cut edge set $\mathcal{C}$.

Also by definition, we know which terminal links have to be part of the cut ($\rightarrow$ eq. (3.30)), so what stays are the n-links. For those we distinguished two cases, depending on the equality of neighbouring assignments. Using properties (3.31) and (3.34) we can uniquely define the remaining cut edges.

For the cost of an elementary cut we can state

$$|\mathcal{C}| = \sum_{p\in\mathcal{P}} |\mathcal{C} \cap \{t_p^\alpha, t_p^{\bar{\alpha}}\}| + \sum_{\substack{\{p,q\}\in\mathcal{N} \\ f_p = f_q}} |\mathcal{C} \cap e_{[pq]}| + \sum_{\substack{\{p,q\}\in\mathcal{N} \\ f_p \neq f_q}} |\mathcal{C} \cap \mathcal{E}_{[pq]}| \tag{3.35}$$

From the capacity table 3 together with (3.30) we can infer that

$$|\mathcal{C} \cap \{t_p^\alpha, t_p^{\bar{\alpha}}\}| = D_p(f_p^{\mathcal{C}})$$

Then incorporating the results from (3.31) and (3.34) we conclude

$$|\mathcal{C}| = \sum_{p\in\mathcal{P}} D_p(f_p^{\mathcal{C}}) + \sum_{\{p,q\}\in\mathcal{N}} V(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}) = E(f^{\mathcal{C}}) \tag{3.36}$$

□

Thus, if by finding a minimal cut on $\mathcal{G}_\alpha$, we can find the labelling within one α-expansion from f with lowest energy.

3.2.4 Optimality

After having shown that the cost of cuts on expansion graphs equals the energy of the corresponding labelling, we now want to go one step further. Using the expansion move algorithm we can iterate single expansions until no better configuration can be found for any label. We will now show that we can guarantee for such a local minimum to be within a certain range from the *global* minimum.

Let V be a metric according to all requirements (4.1)-(4.3). Further, we define an important constant

$$c = \frac{\max_{\alpha\neq\beta\in\mathcal{L}} V(\alpha,\beta)}{\min_{\alpha\neq\beta\in\mathcal{L}} V(\alpha,\beta)}$$

At this point, it might be important to mention the fact that our label set is discrete. Thus, the smallest and a largest difference of two distinct labels can be found, and these differences are neither 0 nor tending to ∞. E.g. if we use as label set all integers between 0 and 255, $\mathcal{L} = \{0, 1, \ldots, 255\}$, then we can give even for the non-truncated linear or quadratic metrics this constant as

$$c = \frac{(255-0)^2}{(0-1)^2} = 255^2$$

Later in this document we will introduce special terms additionally to the interaction penalties that depend per pixel on the images. To generalise this constant already now to this case, we define for any two neighbouring pixel positions

$$c = \max_{p,q \in \mathcal{N}} \left(\frac{\max_{\alpha \neq \beta \in \mathcal{L}} V_{p,q}(\alpha, \beta)}{\min_{\alpha \neq \beta \in \mathcal{L}} V_{p,q}(\alpha, \beta)} \right)$$

In essence, we have an upper bound for the ratio between the largest and the smallest possible penalty. With this, in complete accordance to [13], one can then claim

Theorem For a labelling $\hat{f}$ being a local minimum of the expansion move algorithm and for f^* being the global minimum, we claim that

$$E(\hat{f}) \leq 2 \cdot c \cdot E(f^*). \tag{3.37}$$

Proof For all $\alpha \in \mathcal{L}$ we can define the set of all pixels of the global minimum that have disparity α.

$$\mathcal{P}_\alpha = \left\{ p \in \mathcal{P} | f_p^* = \alpha \right\}$$

Next, we can create a new labelling f^α from $\hat{f}$ by replacing all pixels that are assigned to α in the global minimum by α in the locally minimal solution $\hat{f}$. This new configuration reads

$$f_p^\alpha = \begin{cases} \alpha & \text{if } p \in \mathcal{P}_\alpha \\ \hat{f}_p & \text{else} \end{cases}$$

Since we assumed that $\hat{f}$ is a local minimum of the energy function, we definitely know that

$$E(f^\alpha) \geq E(\hat{f}). \tag{3.38}$$

In the following we need the ability to restrict our energy function to subsets of the pixel set. To this end we define for the set S consisting of any number of pixels and neighbourhood pixel pairs the energy of the restricted set S as

$$E(f|S) := \sum_{p \in S} D_p(f_p) + \sum_{\{p,q\} \in S} V(f_p, f_q)$$

The actual set to which we want to restrict our selves are inner, outer and boundary pixels of $\mathcal{P}_\alpha$. More precisely we need

$$\begin{array}{rccl} I^\alpha = & \mathcal{P}_\alpha & \cup & \{\{p,q\} \in \mathcal{N} | p \in \mathcal{P}_\alpha, q \in \mathcal{P}_\alpha\} \\ B^\alpha = & & \cup & \{\{p,q\} \in \mathcal{N} | p \in \mathcal{P}_\alpha, q \notin \mathcal{P}_\alpha\} \\ O^\alpha = & (\mathcal{P} \setminus \mathcal{P}_\alpha) & \cup & \{\{p,q\} \in \mathcal{N} | p \notin \mathcal{P}_\alpha, q \notin \mathcal{P}_\alpha\} \end{array}$$

Note already now that the union of these three sets is the set of all pixel and all neighbourhood pairs, i.e. $I^\alpha \cup B^\alpha \cup O^\alpha = \mathcal{P} \cup \mathcal{N}$. Since f^α does not differ from $\hat{f}$ in O^α, we know that

$$E(f^\alpha | O^\alpha) = E(\hat{f} | O^\alpha) \tag{3.39}$$

Also by the previous definitions we can be sure that

$$E(f^\alpha | I^\alpha) = E(f^* | I^\alpha) \tag{3.40}$$

By the properties of the interaction penalty V we can estimate definitely that for any neighbourhood pair $\{p,q\} \in B^\alpha$ it holds that

$$V(f_p^\alpha, f_q^\alpha) \leq c \cdot V(f_p^*, f_q^*).$$

This is true since we found out that the left side of this inequality cannot grow more than by the computed constant $c > 1$. Applying this estimate to all elements of B^α, we can state

$$E(f^\alpha | B^\alpha) \leq c \cdot E(f^* | B^\alpha) \tag{3.41}$$

We can now use these findings to rewrite equation (3.38) as follows: We split the total energy of $\hat{f}$ into three parts.

$$E(\hat{f}) = E(\hat{f} | O^\alpha) + E(\hat{f} | I^\alpha) + E(\hat{f} | B^\alpha)$$

This is valid since the three sets are disjoint and their union is $\mathcal{P} \cup \mathcal{N}$, as described before. Then we can write

$$\underbrace{E(\hat{f} | O^\alpha) + E(\hat{f} | I^\alpha) + E(\hat{f} | B^\alpha)}_{=E(\hat{f})} \leq \underbrace{E(f^\alpha | O^\alpha) + E(f^\alpha | I^\alpha) + E(f^\alpha | B^\alpha)}_{=E(f^\alpha)} \tag{3.42}$$

This expression can be significantly simplified by plugging in the results from (3.39),(3.40) and (3.41) as

$$E(\hat{f}|O^\alpha) + E(\hat{f}|B^\alpha) \leq E(f^*|I^\alpha) + c \cdot E(f^*|B^\alpha) \tag{3.43}$$

All these terms are naturally non-negative, thus it is valid to sum over all labels.

$$\sum_{\alpha\in\mathcal{L}} \big(E(\hat{f}|O^\alpha) + E(\hat{f}|B^\alpha)\big) \leq \sum_{\alpha\in\mathcal{L}} \big(E(f^*|I^\alpha) + c \cdot E(f^*|B^\alpha)\big) \tag{3.44}$$

Consider $B = \bigcup_{\alpha\in\mathcal{L}} B^\alpha$ which consists of neighbourhood pairs only. Thus, for each single included pair $\{p, q\} \in B$ we only get a smoothness contribution, i.e.

$$E(\hat{f}|\{p,q\}) = V(\hat{f}_p, \hat{f}_q) \quad \text{and} \quad E(f^*|\{p,q\}) = V(f^*_p, f^*_q)$$

Moreover, these V-terms are symmetric in their argument and occur *twice* on the left as well as on the right side of equation (3.44): For the left hand side $E(\hat{f}|B^\alpha)$ we get this term once for $\alpha = f^*_p$ and once for $\alpha = f^*_q$. Similarly the right hand side where we get the term $V(f^*_p, f^*_q)$ $2 \cdot c$ times.

Once having this result, we can get the main result

$$E(\hat{f}) + E(\hat{f}|B) \leq E(f^*) + (2c-1)E(f^*|B) \leq 2cE(f^*) \tag{3.45}$$

Thus we have proven that a if we compute a local minimum using expansion moves, the energy of this minimum is within a factor of $2c$ from the energy of a global minimum. Especially we can fix the constant c for the Potts model at $c = 1$. □

Before concluding this section, we refer to an US patent on the just discussed swap and expansion moves that Zabih et al. filed in 2000 and which was finally issued in 2004 to them, see [66]. This patent covers basically all contents of the publication [13], including their swap and expansion moves.

We want to close this section on expansion moves by presenting corresponding results. We reconsidered the signal used throughout the discussion of 1-D image regularisation from figure 7 and applied the given expansion move algorithm to it. Figure 19 shows results for different choices of the regularisation parameter α. Next, figure 20 compares the results obtained by the swap and expansion move algorithms.

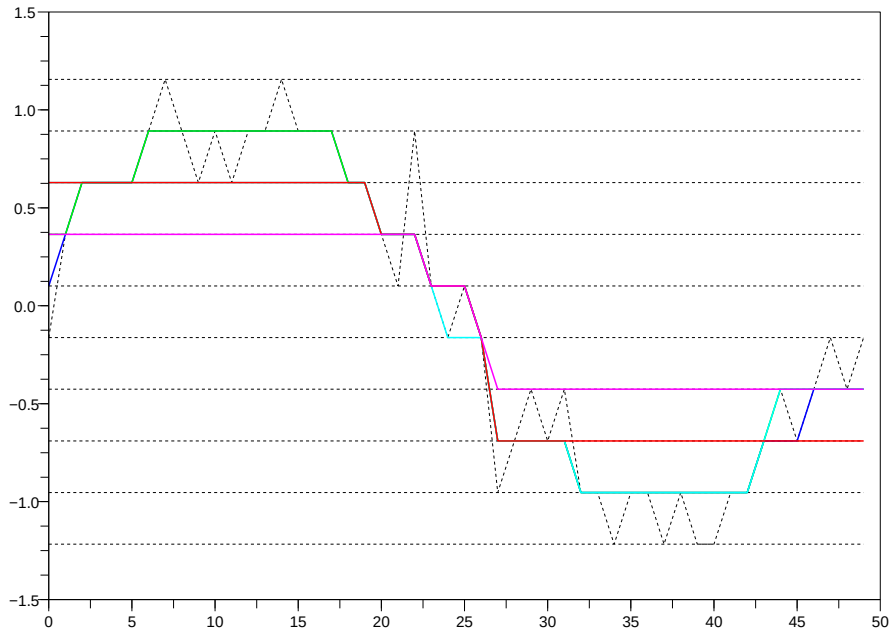


Figure 19: Expansion move algorithm applied to the input signal from figure 7. **Blue line:** Filtered signal with $\alpha = 0.5$. **Green line:** $\alpha = 1$. **Cyan line:** $\alpha = 2$. **Red line:** $\alpha = 4$. **Violet line:** $\alpha = 16$.

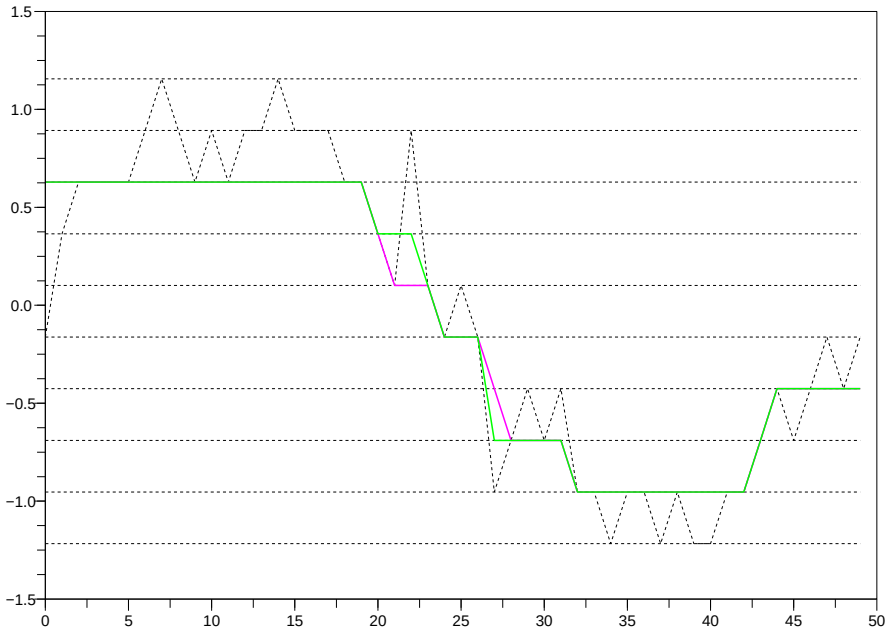


Figure 20: Comparison of swap and expansion move algorithms. Both methods used the linear smoothness penaliser and regularisation parameter $\alpha = 2$. One can recognise that the filter results are very similar **Green line:** Expansion move algorithm result. **Violet line:** Swap result (only visible where the two signals do not coincide).

4 Metrics

As already briefly mentioned in sections 3.2.2 and 3.2.3, interaction penalty functions play a central role in all applications of this thesis. Starting with quadratic functions we already introduced the a bit more advanced concept of linear penalising functions. In this section we want to detail on such functions, and provide more instances of such penalisers being well suited for our applications. But before listing new functions, we want to introduce the standard requirements that such functions have to at least partly fulfil in order to qualify for our purposes.

We define a penaliser function $V : \mathcal{L} \times \mathcal{L} \rightarrow \mathbb{R}$ to be a *metric* if it fulfils the following three requirements:

$$V(\alpha, \beta) = 0 \quad \Leftrightarrow \quad \alpha = \beta \tag{4.1}$$

$$V(\alpha, \beta) \quad = \quad V(\beta, \alpha) \quad \geq \quad 0 \tag{4.2}$$

$$V(\alpha, \beta) \quad \leq \quad V(\alpha, \gamma) + V(\gamma, \beta) \tag{4.3}$$

As already touched on, there are also interesting functions satisfying only the first two requirements (4.1) and (4.2). Such functions will be called *semi-metrics* throughout this document.

For the sake of formality, let us notice that every metric is a semi-metric. One should keep this obvious corollary in mind when minimising metric functions, because besides expansion moves, also for metrics swap moves are applicable, but often neglected.

Already for the proof of the correctness of the expansion graph construction in section 3.2.3, the triangle inequality (4.3) was the central ingredient for the whole construction to be valid. But there is a deeper connection that we want to detail on in the next part of this section.

4.1 Metrics and Subadditivity

With the requirement (4.3), we demand a function to satisfy the triangle inequality, in order to be a metric. Fulfilling this triangle inequality in practice for a function always means, intuitively speaking, that it should be cheaper to go directly from A to B, instead of making a detour over C. Since the functions V that we are considering in practice all can be represented by a function of the difference of the two arguments of V, we can use this to get the transition to a well known concept in analysis. Using

the transformation

$$V(\alpha, \beta) \;=\; \Psi(\alpha - \beta) \tag{4.4}$$

we can rewrite the triangle inequality (4.3) to the form

$$\Psi(\alpha - \beta) \quad \leq \quad \Psi(\alpha - \gamma) + \Psi(\gamma - \beta). \tag{4.5}$$

By defining $x = \alpha - \gamma$ and $y = \gamma - \beta$, we get

$$\Psi(x + y) \quad \leq \quad \Psi(x) + \Psi(y). \tag{4.6}$$

This latter result is the so-called Minkowski inequality, and also the requirement for a function to be *sub-additive*. Note that in the definition of Ψ, it does not matter whether the argument is positive or not, since all functions we use are symmetric, see (4.2). Thus it must hold by assumption that $\Psi(\alpha - \beta) = \Psi(\beta - \alpha)$, which is obviously fulfilled for all functions we use.

Now, we know that metrics are subadditive. Graphically speaking, subadditive functions have to lie "*above*" a certain corresponding linear function. Given $x, y \in \mathbb{R}$, wlog. $x < y$, we know that by the subadditivity it holds that $\Psi(x + y) \leq \Psi(x) + \Psi(y)$, see also figure 21. Then we can state

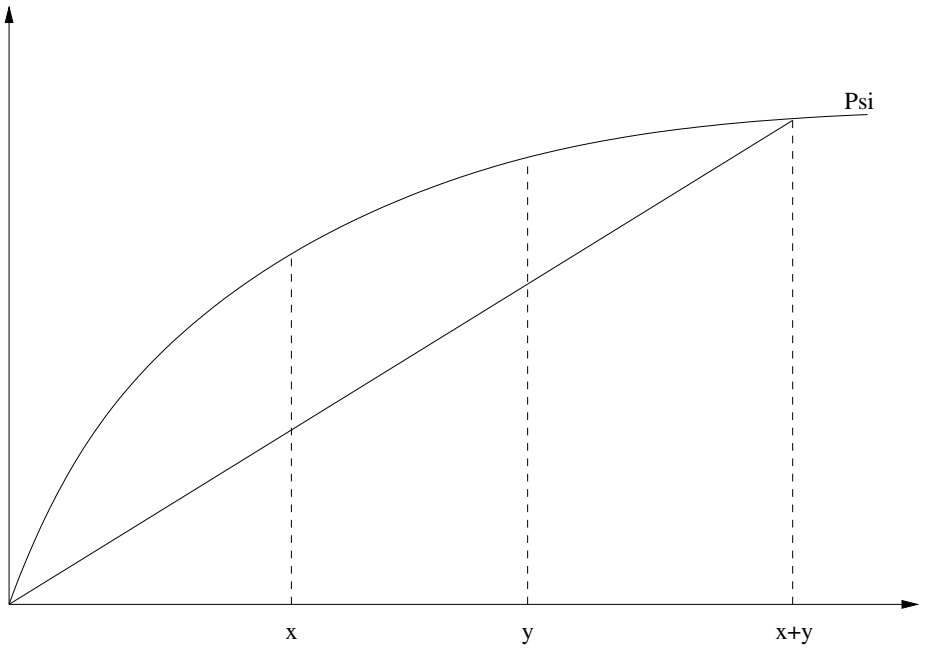


Figure 21: A function Ψ has to lie on or above the linear function in order to be subadditive.

the following Lemma, which relates subadditivity to concavity.

Lemma A concave function is subadditive.

Proof A function $f : \mathbb{R} \to \mathbb{R}$ is concave, if for all $x, y \in \mathbb{R}$, $\lambda \in [0, 1]$ it holds that

$$f\left(\lambda x + (1 - \lambda)y\right) \geq \lambda f(x) + (1 - \lambda) f(y)$$

This is illustrated in figure 22. Thus, let Ψ be concave. For any two real

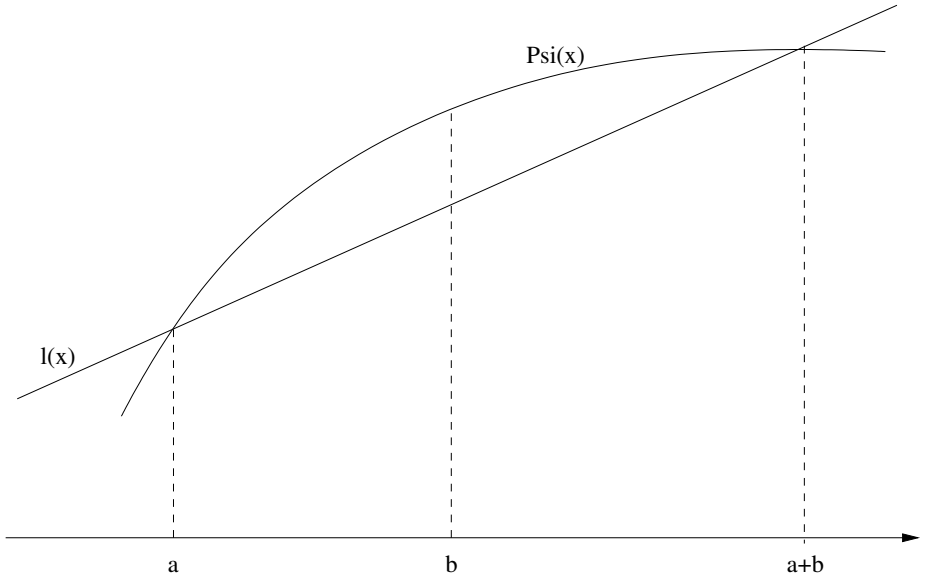


Figure 22: Sketch of a concave function. Between any two points, the graph of Ψ is above the corresponding linear function.

numbers a and b, without loss of generality we can assume $a < b$, we know that Ψ is concave on the interval $[a, a + b]$. Hence we know since b lies in that interval that there exists a $\lambda \in [0, 1]$ such that

$$\Psi(b) \geq \lambda \cdot \Psi(a) + (1 - \lambda) \cdot \Psi(a + b).$$

The corresponding value of λ can be easily determined, it is $\lambda = \frac{b}{a}$. Plugging this in and adding $\Psi(a)$ on both sides from the left we get

$$\Psi(a) + \Psi(b) \geq \Psi(a) + \underbrace{\frac{b}{a} \cdot \Psi(a) + (1 - \frac{b}{a}) \cdot \Psi(a + b)}_{= l(b)}$$

In the latter equation we annotated the quantity $l(b)$, which is the value of the linear function $l : \mathbb{R} \to \mathbb{R}$ going through the two points $(a, \Psi(a))$

and $(a+b, \Psi(a+b))$ at position b. Since l is linear, and $l(a) = \Psi(a)$ and $\Psi(a+b) = l(a+b)$ we can write

$$\begin{aligned}\Psi(a) + \Psi(b) &\geq l(a) + l(b) \\ &= l(a+b) \\ &= \Psi(a+b)\end{aligned}$$

which concludes the proof. Note that the opposite way of this lemma is not true in general. As an example we want to consider the function $f : \mathbb{R}^+ \to \mathbb{R}$

$$f(x) = e^{-x} - e^{-ax} + x \qquad \text{for } a > 1,\ x > 0 \tag{4.7}$$

which is plotted in figure 23. By straightforward computations we can find

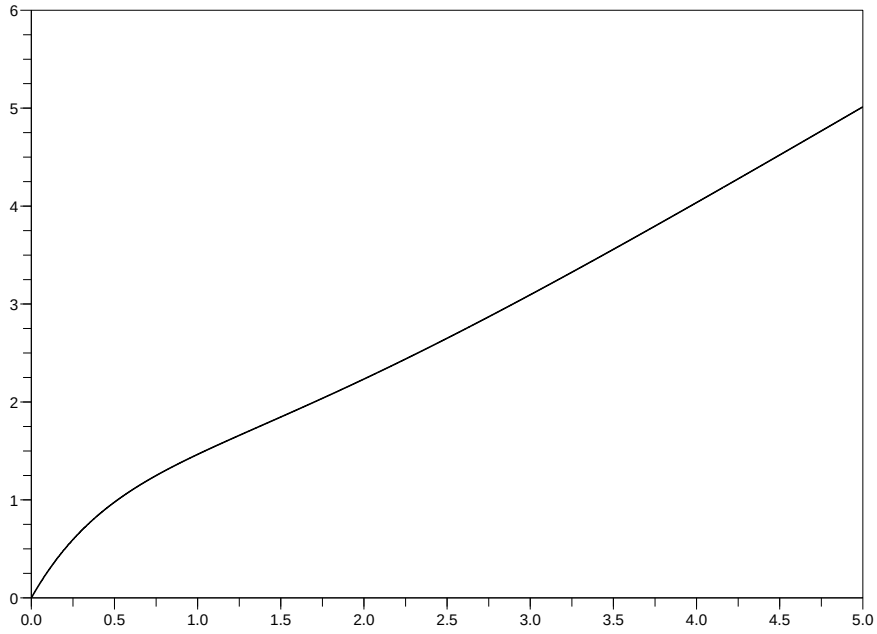


Figure 23: Example for the subadditive function from (4.7) for $a = 2$. f starts concave, but after the inflection point at $x = \frac{\ln(a^2)}{a-1}$ it continues as a convex function.

out that

$$\begin{aligned}f(x+y) &= e^{-(x+y)} - e^{-a(x+y)} + x + y \\ &= e^{-x}e^{-y} - e^{-ax}e^{-ay} + x + y \\ &\overset{*}{\leq} e^{-x} + e^{-y} - e^{-ax} - e^{-ay} + x + y \\ &= f(x) + f(y)\end{aligned}$$

At step $(*)$ we used that for $x \geq 0$ it holds that $0 < exp^{-x} \leq 1$. Hence, the function fulfils the Minkowski inequality while at the same time being convex for all $x > \frac{\ln(a^2)}{a-1}$.

As already mentioned, the choice of the penalising function is a central question for our algorithms. Moreover, by using one or an other penalty, we can already in the modelling process influence, how the minimum should look like. It turns out that one central question is, whether the used penalty fulfils the triangle inequality or not. The reason for this is can be intuitively explained as follows: The input data together with the data term tells us that the disparity or the grey value must change by some certain value. On the other hand, the smoothness term tries to avoid changes. Depending now on the shape of this smoothness penalty, either bridging this gap is cheaper when taking it in one step, or it could be cheaper to make more than one small steps. The third possibility is that it could be equally costly no matter how many steps we take. The first case resembles our metrics: We have to bridge a gap of height $x + y$ and we know that $\Psi(x + y) \leq \Psi(x) + \Psi(y)$. Thus making one large step is always cheapest. In the second case, e.g. for a quadratic penaliser, it is always cheaper to make multiple small steps, since $\Psi(x + y) \geq \Psi(x) + \Psi(y)$. The third case is a special case, which we will call the linear penalty $\Psi(x) = |x|$. For this function, it will not matter whether to bridge the gap in one large or several small portions. The resulting penalty will be the same.

This means, that if a function minimises such an energy consisting of data and smoothness term, it will automatically make use of these preferences. Thus, using a quadratic penaliser, the solution cannot have sharp jumps or discontinuities, they must be smooth. Conversely, using a subadditive penalty function we will not be able to get smooth solutions.

After the discussion of the theory behind different penalty functions, we want to introduce several possible choices of metrics and semi-metrics in the next part of this section.

4.2 Interaction Functions

4.2.1 Quadratic Interaction Penalty

We define the quadratic semi-metric as

$$V^{\mathbf{quadratic}}(\alpha, \beta) \;=\; (\alpha - \beta)^2 \tag{4.8}$$

and a truncated variant of it as

$$V_K^{\mathbf{quadratic}}(\alpha,\beta) \;=\; \min\{\,(\alpha-\beta)^2, K\,\}. \tag{4.9}$$

A plot of all presented functions of this section is given in figure 24 in the end of the discussion.

Concerning the requirements that have to be fulfilled by a semi-metric, we can directly see, that (4.1) and (4.2) hold by construction. For the triangle inequality, we can easily give an example showing that (4.3) does not hold:

$$\begin{gathered}\alpha = 1 \quad , \quad \gamma = 2 \quad , \quad \beta = 4 \\ V(1,4) = (1-4)^2 = 9 > V(1,2) + V(2,4) = 1^2 + 2^2 = 5\end{gathered}$$

4.2.2 Linear Interaction Function

The linear smoothness penalty function is defined as

$$V^{\mathbf{linear}}(\alpha,\beta) \;=\; |\alpha-\beta| \tag{4.10}$$

As before we will also consider a truncated version

$$V_K^{\mathbf{linear}}(\alpha,\beta) \;=\; \min\{\,|\alpha-\beta|, K\,\}. \tag{4.11}$$

The linear interaction penalty is a metric. Given two labels $\alpha,\beta \in \mathbb{R}$, wlog. let $\alpha < \beta$, then for any third label $\gamma \in \mathbb{R}$ we distinguish 3 cases:

$$\begin{aligned}
\alpha \le \gamma \le \beta: \quad V(\alpha,\gamma) + V(\gamma,\beta) &= |\alpha-\gamma| + |\gamma-\alpha| \\
&= \gamma - \alpha + \beta - \gamma \\
&= \beta - \alpha \\
&= V(\alpha,\beta) \\
\gamma \le \alpha \le \beta: \quad V(\alpha,\gamma) + V(\gamma,\beta) &= \alpha - \gamma + \beta - \gamma \\
&= \alpha + \beta - 2\gamma \\
&\overset{\gamma\le\alpha}{\le} \alpha + \beta - 2\alpha \\
&= \beta - \alpha \\
&= V(\alpha,\beta) \\
\alpha \le \beta \le \gamma: \quad V(\alpha,\gamma) + V(\gamma,\beta) &= \alpha - \gamma + \gamma - \beta \\
&= 2\gamma - \alpha - \beta \\
&\overset{\gamma\ge\beta}{\le} 2\beta - \alpha - \beta \\
&= \beta - \alpha \\
&= V(\alpha,\beta)
\end{aligned}$$

4.2.3 Truncated linear penalty

Inspired by [3], we also use the following linear truncated penalty function

$$V_K^{\mathbf{lintrunc}}(\alpha, \beta) \;=\; -\ln\left(e^{-K} + (1 - e^{-K})e^{-|\alpha-\beta|}\right) \tag{4.12}$$

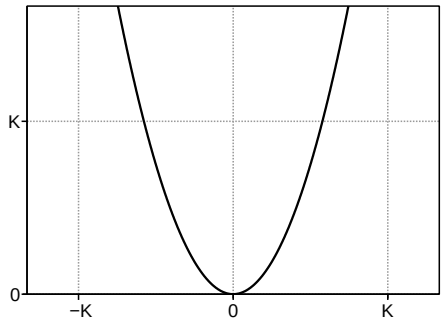
Since the function is globally concave (on the positive x-axis), we know that it is subadditive and fulfils all requirements of a metric.

4.2.4 Potts metric

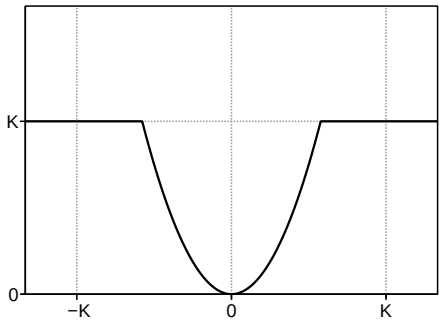
As last representative of penalty functions, we want to introduce the so-called Potts model which is an extreme case for penalising functions. It assigns to any input a constant value $K > 0$ except if the argument is zero.

$$V_K^{\mathbf{Potts}}(\alpha, \beta) \;=\; \begin{cases} K & \text{, if } \alpha \neq \beta \\ 0 & \text{, else} \end{cases} \tag{4.13}$$

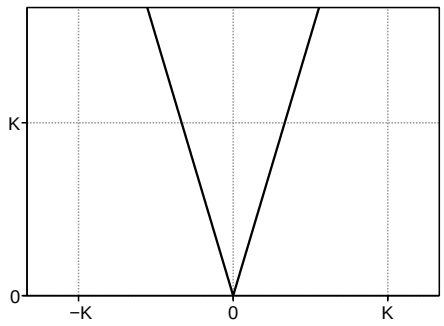
By this definition, one can see that this model prefers piecewise constant results, because any variations in the result are penalised equally severely. For the Potts model, we get $c = 1$ for the considerations in section 3.2.4, thus the solution can be bounded to be within a factor of two of the global minimum. Moreover, we found out that in general $c \geq 1$, which in turn implicates that the Potts model is in some sense optimal. No other penaliser can bound our estimate tighter.



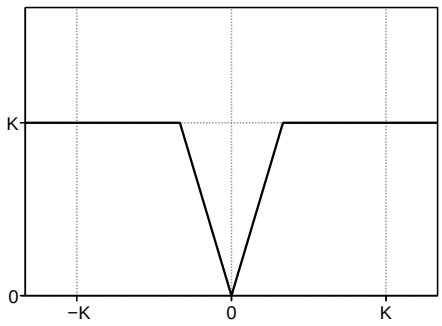
$V^{\mathbf{quadratic}}(\alpha, \beta) = (\alpha - \beta)^2$



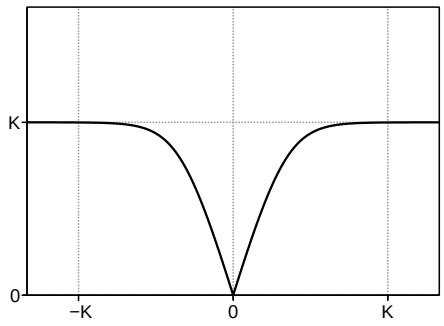
$V_K^{\mathbf{quadratic}}(\alpha, \beta) = \min\{ (\alpha - \beta)^2, K \}$



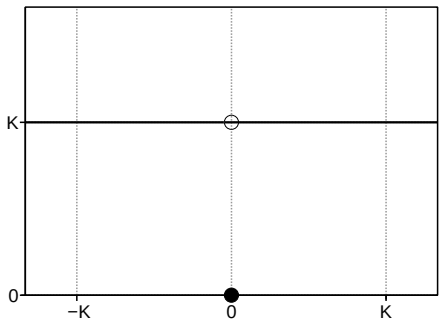
$V^{\mathbf{linear}}(\alpha, \beta) = |\alpha - \beta|$



$V_K^{\mathbf{linear}}(\alpha, \beta) = \min\{ |\alpha - \beta|, K \}$



$V_K^{\mathbf{lintrunc}}(\alpha, \beta) = -\ln\left(\epsilon + (1-\epsilon)e^{-|\alpha-\beta|}\right)$



$V_K^{\mathbf{Potts}}(\alpha, \beta) = K \cdot T(\alpha \neq \beta)$

Figure 24: Plot of the interaction penalties discussed in this section.

5 Stereo Vision

With this section we want to introduce the major topic of this thesis, *Stereo Vision*.

The main problem we want to solve is as follows: Given two views of a scene, taken from different view points, we want to recover the depth in the scene. This means that from two two-dimensional images, we want to determine one 3-D model of the objects of the scene. An example of two such images together with the result is given in figure 25. There is also a more general related problem, called *multi-view scene recovery* where more than two images are given.

An Experiment The general principle, why and how this task can be realised at all, can be illustrated with the following short experiment: We will use the two images of our two eyes as the two views we want to consider. By closing one eye, our brain perceives the 2-D image of the still open eye solely. The closing of one eye to get a 2-D image for us as humans is mandatory, since if we use both our eyes, our brain immediately solves the stereo vision task almost perfectly. In fact, the general idea of performing stereo vision based depth reconstruction must have been inspired by the human visual system and its excellent capabilities.

Coming back to the experiment, after closing one eye, we place an object, for example a thumb of our hand, closely in front of our eyes. Then, by closing the open eye and opening the previously closed one, we can observe, that the position of the thumb in the perceived image jumps widely. By moving the thumb further away from the eyes, and by re-performing the opening and closing of the eyes, we should observe that the distance by which the thumb moves between the images becomes smaller. The result of this simple experiment is that depth in the scene can be closely related to offsets of objects between the images - the larger the offset, the closer the object to the camera.

Thus, the problem is to find and determine these offsets. Objects occurring in both images have to be identified and the difference of their position is what we are searching for. The unit of such offsets are 2-D vectors that describe the displacement of an object. Note that there are also situations where an object that is visible in one view does not occur in the second image. Such constellations are called *occlusions*. Also, there might be cases

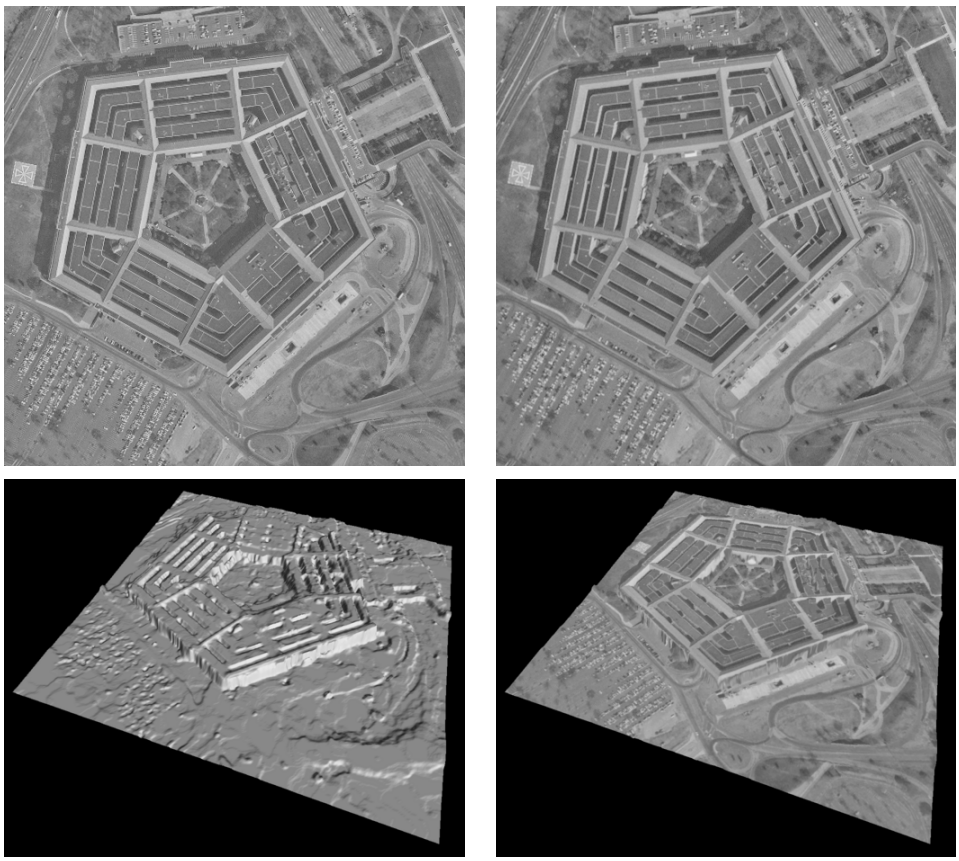


Figure 25: Introductory stereo vision example. The task is to compute a 3-D model of the scene given just the first two images. **(a) Top left:** First input image. **(b) Top right:** Second given view of the scene. **(c) Bottom left:** Computed model with just lighting. **(d) Bottom right:** Computed model from (c) with texture mapping. Images and results from [53].

that describe physically impossible configurations that should of course be avoided by our models.

Camera Model and Geometry Figure 26 illustrates the underlying so-called *pinhole camera model* which is essential to describe formally the image formation process and geometry behind all our considerations. One can see in this figure that each real world point $M \in \mathbb{R}^3$ is projected through the optical centre C of the pinhole camera - the pinhole. The image plane is located *behind* the optical centre, in distance f of C, where f is called

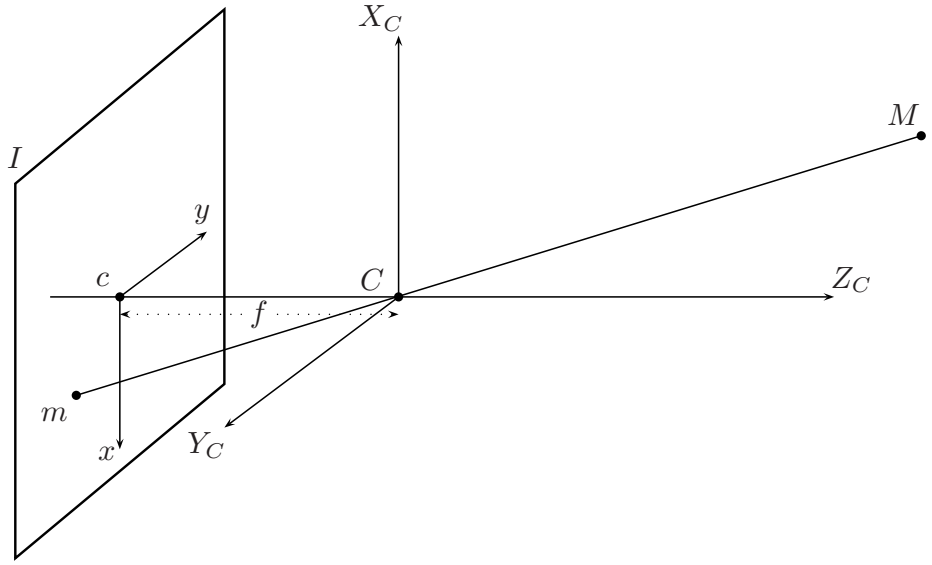


Figure 26: Pinhole camera model. Image courtesy of Markus Mainberger.

the *focal length*. Furthermore, the Z-axis of the world coordinate system (X_C, Y_C, Z_C) is arranged orthogonally to the image plane and also to the image coordinate system (x, y). Note that images acquired in this constellation would be upside down. Therefore, often one shifts the image plane to the other positive side of the optical centre at position $Z = f$. Then the arising images are upright and instead of projecting in the classical way, the intersection of the ray from the world point to the optical centre with the image plane is measured. We do not want to go too deep into detail of this topic here, since it goes beyond the scope of this thesis. Let us concentrate on the arrangement of the second camera instead.

The generally considered arrangement for classical stereo vision problems is the so-called *converging camera setup*, where two cameras are arranged like illustrated in figure 27. Usually the cameras are tilted slightly towards each other. This means that starting from one optical centre, we can come to the other centre by applying a rotation $\mathcal{R}$ as well as an translation t.

One can show that in this setup, one point in the first image cannot lie everywhere in the second image. There are well defined constraints that limit the corresponding point to lie on the so-called epipolar line, denoted by

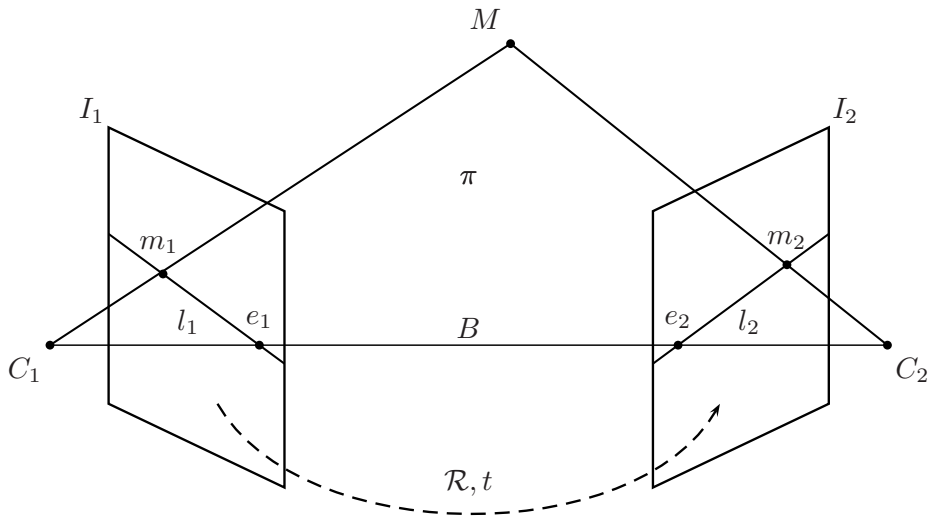


Figure 27: Converging camera setup. The pose of the two cameras does not have many restrictions. Image courtesy of Markus Mainberger.

e_1 and e_2 in figure 27. These epipolar lines depend on the camera geometry and the arrangements only. We can exploit this fact very well, since this reduces the dimension of our search space by one. Instead of searching in the whole 2-D image, we now can be sure to find a corresponding point on one (1-D) line in the other image. Detailing further on this topic of the geometry of stereo vision would go beyond the scope of this thesis. This topic is well understood and examined and can be read e.g. in [28, 34].

Another common setup, which we will use throughout this document is the so called *ortho-parallel camera setup*. In this setup, the image planes of the two cameras are assumed to be perfectly parallel to each other and at the same depth position. Further, also the corresponding axes of the internal coordinate systems of both cameras are parallel. Figure 28 illustrates this special arrangement. Obviously, the ortho-parallel setup is a special case of the converging setup, where no rotation and just a translation between the optical centres takes place. The big advantage that this setup provides is that if the images are taken like that, all occurring displacements are solely along the x-axis of the images. This means that the displacement vector we mentioned before will always have the second component being zero. In practice we will always have to search only in the same pixel scan line.

We also want to mention that given a pair of stereo images recorded in the general converging camera setting, there exist methods that transform

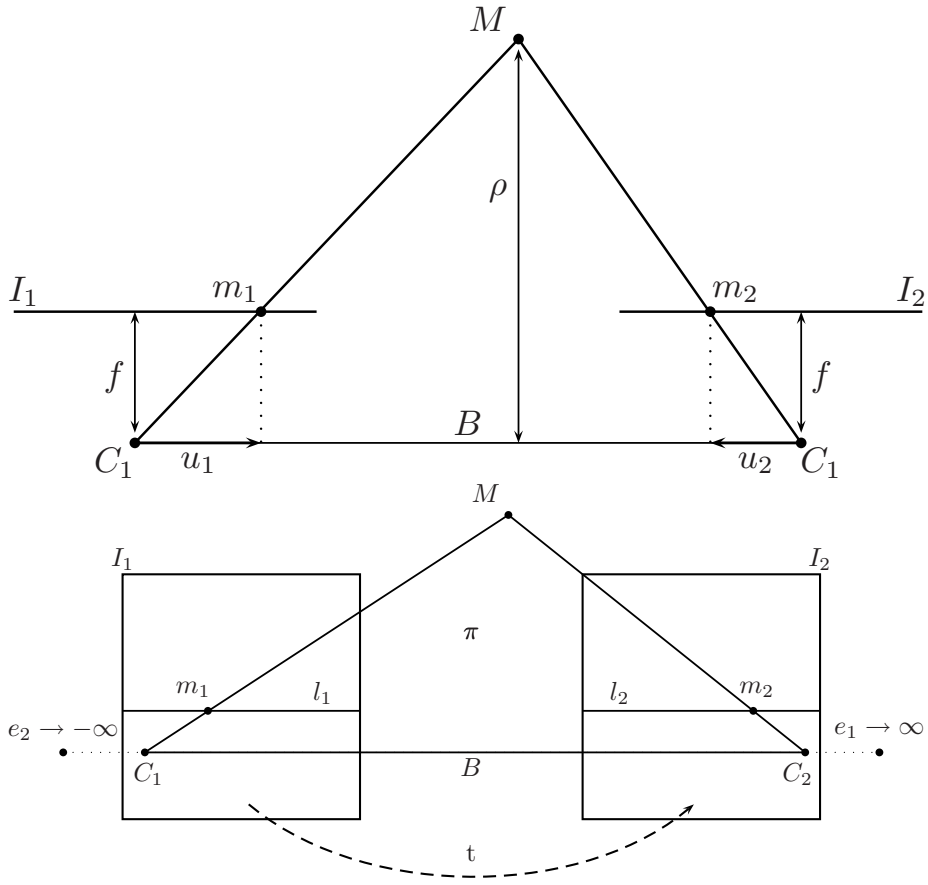


Figure 28: Ortho-parallel camera setup. The two image planes I_1 and I_2 are lying in one plane and the internal coordinate systems are parallel as well. **Top:** Schematic view from top on both cameras and world object point M. **Bottom:** Schematic view from behind. Images courtesy of Markus Mainberger.

such images to the ortho-parallel case. Such techniques are known under the name *image rectification*, see [31, 34] and references therein.

Establishing Correspondences Hence, we will focus on the ortho-parallel setting for this document. The task there will be to identify pixels in both images that *correspond*. This means that one pixel in the first frame that corresponds to another pixel in the second frame should belong to the same object or show the same part of the same object. Since we are in the ortho-parallel setting, we already said, that only 1-D displacements along the x-axis will occur, in which case one speaks of the *disparity field* which is to be estimated.

Let the left and right grey value images

$$I_l : \Omega_l \to \mathbb{R} \quad \text{and} \quad I_r : \Omega_r \to \mathbb{R}$$

in their usually rectangular domains $\Omega_l, \Omega_r \subset \mathbb{R}^2$ be given. We say that the disparity field $d_l : \Omega_l \to \mathbb{R}$ relates a point $\mathbf{x} = (x, y)^\top \in \Omega_l$ in the left frame to a point $\mathbf{x}' \in \Omega_r$ in the right frame via

$$\mathbf{x}' = (x + d_l(\mathbf{x}), y)^\top \tag{5.1}$$

Note that in this notation, the disparity d_l must attain negative values. This effect is illustrated in figure 29, where one can see that a feature in the left frame must be to the right of the corresponding feature in the right frame. Otherwise, the object would have to be located behind the camera, which obviously does not make sense.

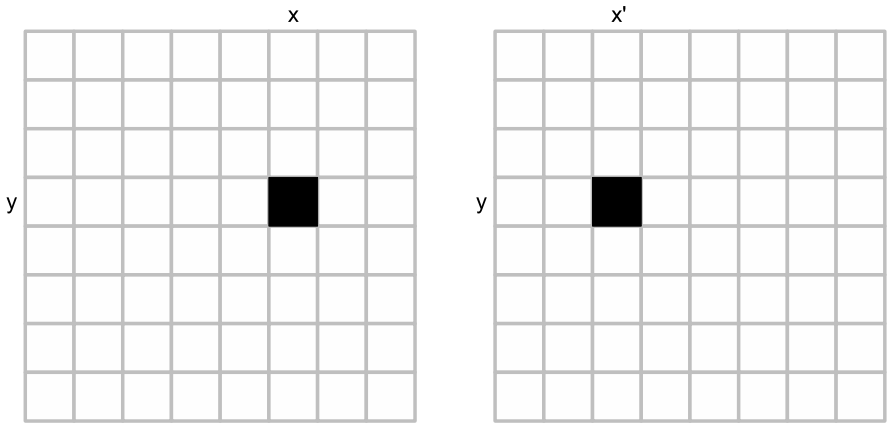


Figure 29: Schematic sketch of the image pixel rasters for the ortho-parallel setup. Displacements only occur along the horizontal x-axis.

As for the latter left-to-right disparity d_l, the right-to-left disparity d_r relates a point $\mathbf{x} \in \Omega_r$ to $\mathbf{x}' \in \Omega_l$ using the analogous equation

$$\mathbf{x}' = (x + d_r(\mathbf{x}), y)^\top \quad \mathbf{x} \in \Omega_r\,,\ \mathbf{x}' \in \Omega_l$$

The only difference is that for the right-to-left disparity, geometrically correct and meaningful fields attain positive values.

As for the image regularisation methods from the previous section 3, the now presented stereo algorithms will amount to defining and minimising an energy function that will deliver a disparity field like just introduced. Also we will concentrate and present the common left-to-right case predominantly, even though we designed all software implementations for both directions, see also our extensions in section 8.

Error quantification and statistics To be able to judge the performance and quality of a certain stereo algorithm, a common methodology is to test the algorithm on special image pairs for which the perfect result is known. This so-called *ground truth* can then be used to compute statistical measures against the output of the algorithm. In this section, we will use two such frequently used measures:

- **Percentage of bad pixels (BP)**
  A pixel is said to be a *bad* pixel if the computed disparity differs more than a certain threshold $\epsilon > 0$ from the corresponding ground truth pixel. For a ground truth disparity d^{truth} and the computed disparity d^{estimate} the bad pixels measure is calculated

  $$BP_\epsilon := \frac{100}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} T(\mid d_p^{\mathbf{truth}} - d_p^{\mathbf{estimate}} \mid > \epsilon) \tag{5.2}$$

  where the indicator function T is 1 if its argument is true, and 0 otherwise. Obviously, the choice of this threshold value has a crucial meaning for this measure. In comparative literature [47, 49] the choice $\epsilon = 1.0$ is predominantly used.

  Note that, even though the bad pixel measure is a popular statistic, it is not as meaningful and unambiguous as one might think. As an example, if one stereo sequence only contains disparity values between 0 and 10, and in an other sequence occur disparities in the range from 0 to 100, then the $BP_{1.0}$ statistics is much more relaxed for the first pair than for the second.

Furthermore, especially for algorithms working on discrete labels and regarding the fact that the in practice used and evaluated ground truths from [47, 49] only contain integer valued disparity values, another inconsistency suggests itself: Using integer valued labels with the given ground truths will will not assign a pixel being labelled $d_p^{\mathbf{estimate}} = d_p^{\mathbf{truth}} \pm 1$ to the bad pixels.

- **Average absolute disparity error (AADE)**
  The second statistical measure we will consider is the average absolute disparity error which is computed via

$$AADE = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} | \, d_p^{\mathbf{truth}} - d_p^{\mathbf{estimate}} \, | \tag{5.3}$$

In section 7, we will introduce a stereo algorithm that will take occlusions into direct consideration and that will identify occluded pixels explicitly. This means that this algorithm does not assign a disparity value to such pixels, and hence the previously defined statistical measures have to be adapted accordingly. For the latter AADE it is straightforward to just compute the average on those pixels that were not assigned to the occluded region. For the BP measure, the adoption is more difficult, there are more possible solutions. The ground truth maps also provide occlusion maps where those pixels that are really occluded are marked. The discussion of suitable error measures for this case is postponed to the point where we will need them.

The remainder of this document is organised as follows: The upcoming classical graph cut based stereo approach in section 6 will estimate a 100% dense disparity field, this means that for each pixel we will determine a disparity value. The following approach in section 7 will systematically take occlusions into consideration and will also mark such areas as being occluded instead of assigning a disparity value. After having discussed these foreign approaches from literature, we will finally present new extensions and own algorithmic ideas in section 8.

6 Graph Cuts in Stereo Vision

This section is based on the successful publication due to Yuri Boykov et al. *"Fast Approximate Energy Minimisation via Graph Cuts"* [13]. We will like already introduced in section 3 reuse the swap- and expansion graph construction ideas. Since these constructions work for both moves essentially analogously to the constructions from the section on image regularisation, we will just have to detail on the differences between these two methods here.

For signal restoration, the quantity we estimated, and by that the values of our label space $\mathcal{L}$, were function values. For stereo vision, the quantities of interest are disparities. This means that the most natural upper bound on these labels should be the image width, because establishing a correspondence to a pixel outside one image is not what we want. In practice we will for each stereo rig know an upper bound on the maximal occurring disparity value. Thus the set of labels will be disparities, whose sign depends on whether to compute left-to-right or right-to-left disparity fields, see last section.

$$\mathcal{L} = \{0, 1, \dots, l\} \tag{6.1}$$

This is one point where the difference between e.g. variational methods and graph cuts becomes particularly obvious. The label space is completely discretised, and inherent features of variational or classical minimisation methods like sub-pixel accuracy are problematic for graph cut methods and only realisable with additional efforts.

The next big difference of stereo methods to image restoration concerns the data term. For regularisation methods, the purpose of the data term was to keep the estimated signal similar to the input function. For stereo, the data term shall express whether the two pixels $p \in \Omega_l$ and $q \in \Omega_r$ correspond. This notion of correspondence is what we still have to define now. The most classical model assumption on the correspondence of two pixels is that their intensity (their grey value) should be the same or at least similar. This assumption is called the grey value constancy assumption. Thus we can write the stereo data term as

$$D_p(f_p) = (I_p - I_q)^2 \tag{6.2}$$

where, in the left-to-right case the pixel position q should be computed in accordance to equation (5.1) from position p, using f_p as disparity value. We want to mention already now that there are several other choices than a quadratic penalisation possible, that we will examine in section 8.4.

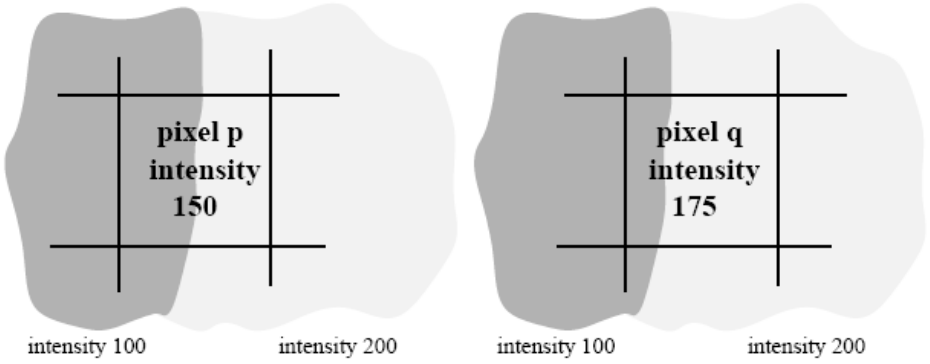


Figure 30: Illustration of the idealised sampling process happing in the left and right camera. The resulting intensity difference is neither due to noise, nor a bias, but it is a property of sampling process happening in the camera and cannot be avoided. Images taken from [57].

Sensitivity to Image Sampling One important additional ingredient concerning the data penalty is due to Birchfield and Tomasi [4]. They assert that the sampling and quantisation process that happens when capturing a pair of stereo images produces several problems in practice. Basically the first problem is that if the cameras are not in the same position, then the light being reflected from a static surface cannot be same in general. Secondly, there might occur differences by slightly varying camera electronics, noise, etc.. But the main observation is that the intensity of a pixel is determined not by the light coming from one single point in the scene, but by integrating the light coming from a whole surface patch in scene. If this surface patch contained a depth discontinuity or if the colour of material changed within the patch, then the integration performs an averaging process to the stored pixel intensity. This phenomenon happens even in an idealised setting, in the absence of any noise, with ideally equal cameras, etc.. See also figure 30 for an illustration of this effect. The remedy that [4] propose is to examine the local neighbourhood of the left and the right image position for the least possible contribution. Speaking in continuous terms, for the right image position we search

$$C_{\text{right}}(p,d) = \min_{d-\frac{1}{2}\leq x\leq d+\frac{1}{2}} |I_l(p) - I_r(p+x)| \tag{6.3}$$

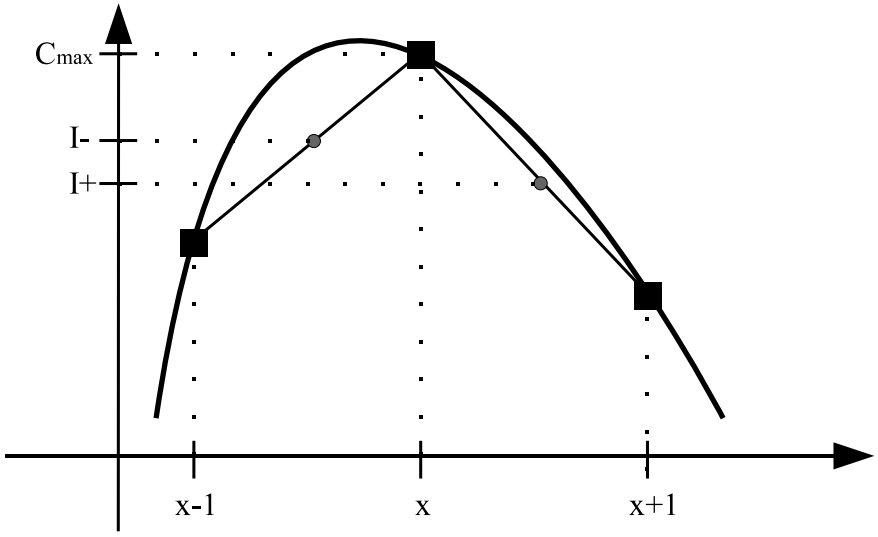


Figure 31: Illustration of the practical realisation of equations (6.3) and (6.4). Squares denote the sampled intensity at grid positions, straight lines their linear interpolation. In essence, the upper and lower bound of the image intensity is determined. For linear interpolation, the extrema can only lie either in the unchanged grid position x, or at the left and right boundaries of the interval.

where d is the disparity to consider and p the pixel coordinate. The grid size is assumed to be 1 in these equations. Similarly we find

$$C_{\text{left}}(p,d) = \min_{-\frac{1}{2}\leq x\leq\frac{1}{2}} |I_l(p+x) - I_r(p+d)|. \tag{6.4}$$

In this way, C_{left} and C_{right} try to adjust the exact pixel position p such that the corresponding intensity difference is minimal. Then the data penalty is found as the minimum of both local minima. In practice the implementation needs an interpolation procedure in order to be able to look up the values of I_l and I_r at sub-pixel positions. By using linear interpolation, also the number possible locations for such minima is limited to 3, as figure 31 illustrates. Note that there could be many more possibilities, if we used another interpolation scheme, hence the fact that the interpolated function arising from linear interpolation is piecewise linear simplifies the problem drastically.

Using for the left-to-right stereo case the abbreviations $x_l = p$ and $x_r =$

$p + d$ we can then compute

$$\begin{aligned} I_r^+ &= I_r(x_r + \frac{1}{2}) \\ I_r^- &= I_r(x_r - \frac{1}{2}) \\ C_r^{\min} &= \min\{\ I_r(x_r), I_r^+, I_r^-\ \} \\ C_r^{\max} &= \max\{\ I_r(x_r), I_r^+, I_r^-\ \} \\ C_{\text{right}}(p,d) &= \max\{\ 0, I_l(x_l) - C_r^{\max}, C_r^{\min} - I_l(x_l)\ \} \end{aligned}$$

And for adjusting the left position we get

$$\begin{aligned} I_l^+ &= I_l(x_l + \frac{1}{2}) \\ I_l^- &= I_l(x_l - \frac{1}{2}) \\ C_l^{\min} &= \min\{\ I_l(x_l), I_l^+, I_l^-\ \} \\ C_l^{\max} &= \max\{\ I_r(x_l), I_l^+, I_l^-\ \} \\ C_{\text{left}}(p,d) &= \max\{\ 0, I_r(x_r) - C_l^{\max}, C_l^{\min} - I_r(x_r)\ \} \end{aligned}$$

Finally we can then compute the data penalty via

$$D(p,d) = \min\{\ C_{\text{left}}(p,d)\ ,\ C_{\text{right}}(p,d)\ \}. \tag{6.5}$$

In practical implementations, applying this method by Birchfield and Tomasi costs only little computational overhead, but gives a remarkable quality gain as shown in figure 32.

The purpose of the smoothness penalty $V_{p,q}(f_p, f_q)$ for stereo vision is very similar as for the discussed image restoration case. It shall penalise all changes in the disparity field. The only difference is the fact that stereo computations will take place on 2-D material. Even though we stated that in the ortho-parallel case we are confronted with a 1-D search problem, still we want to preserve an inter-line consistency of the disparity field. Thus, the links between different scan lines are necessary. But as long as the smoothness assumption just incorporates pairs of pixels, the construction from section 3 is applicable.

An example for a more advanced construction incorporating neighbours from both sides at a time, is [65]. By having the possibility to use two neighbours, second order smoothness terms can be introduced there. But

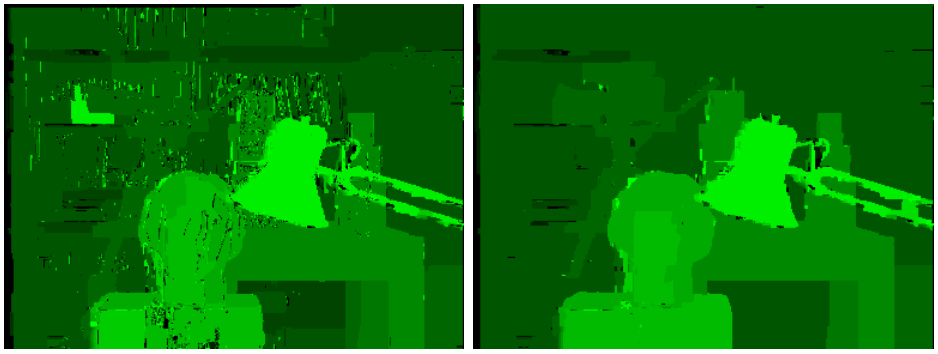


Figure 32: Evaluation of the method by Birchfield and Tomasi [4]. Brighter colors mean larger disparity values. One can clearly see the drastic improvement. Tests were run using expansion moves on the Tsukuba sequence [49].

in this thesis we will not consider such advanced constructions and rely on [13] in this section and the method [38] in section 7. As an extension we want to refer to section 8.2 where we increase the number of considered neighbours from 4 to 8. As we will show there, this can improve the accuracy of our approach.

Image Driven Smoothness In the main reference [13], Boykov et al. propose another successful add-on to their methodology, so called *static-cues*. The underlying assumption is as follows: Without even considering the second frame, empirically the probability that the disparities of two neighbouring pixels $p, q \in \mathcal{N}$ differ is somehow related to the intensity contrast of the pixels in one frame. This means that if two neighbour pixel have the same intensity and thus locally low contrast, then they usually also have the same or similar disparities. The other way round: if the intensity changes remarkably, then this could indicate an object boundary and by that a change of the disparity. Incorporating such assumptions can especially help in low textured regions and constant areas.

This concept is well known in the variational community under the name image driven smoothness term [2], where the weight of the smoothness assumption is decreased with increasing length of the gradient of the input image f.

Inspired by this observation, one can realise this concept on graph cuts as follows. The Potts penaliser will be augmented by an image driven

multiplicative expression $u_{\{p,q\}}$ as

$$V_{pq}(f_p, f_q) = u_{\{p,q\}} \cdot T(f_p \neq f_q) \tag{6.6}$$

Note that in this expression p and q are always neighbours. For simplicity, the image driven term is itself a simple binary case distinction again

$$u_{\{p,q\}} = \begin{cases} K & , \text{ if } |I_p - I_q| \geq 5 \\ \delta \cdot K & , \text{ else} \end{cases} \tag{6.7}$$

where $K > 0$ is the usual Potts model parameter already introduced in section 4 and $\delta > 1$ controls the influence of the static cues. This expression approximates the derivative by just the intensity difference of the first image. If it is below some threshold, the smoothness penalty is simply increased. Note that there are other possibilities how to realise such a technique, one could take the output of an edge detector as well, or even pre-segment the image and use such features, see also our extensions in section 8.

As in [13], we want to motivate image driven smoothness priors with a simple example. Consider the following two frames of a stereo sequence. These images are assumed to not contain any noise, which allows us to

Figure 33: Left and right frame stereo sequence. There shall be a horizontal shift of one pixel.

explicitly determine minimisers of the stereo problem. Without an image driven prior, there are two possible minimisers, as the following figure illustrates. Which of the both disparity solutions is best, depends on the height of foreground and background of input images. Obviously, these solutions do not resemble human perception. By introducing static cues, which would assign less weight to the smoothness prior to neighbour pixels $p, q \in \mathbb{N}$ where $I_p \neq I_q$, then the following disparity field would be a minimiser. Such solu-

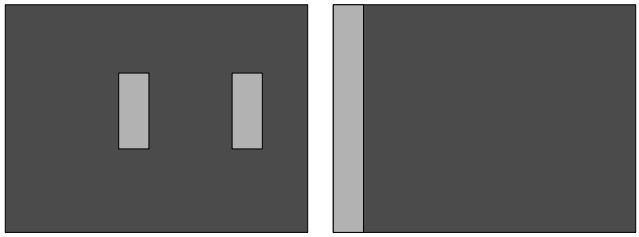


Figure 34: Two possible solutions *without* an image driven term. Brighter colors indicate larger disparity.

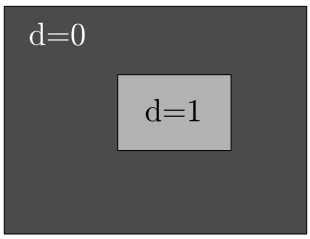


Figure 35: Solution with static cues.

tions often resemble the human perception better than solutions as in figure 34 obtained without image driven terms. Also in terms of quantitative error statistics usually image driven terms improve the result.

Boundary treatment Another issue in the context of stereo vision is the correct treatment of image boundaries. Coming from the field of variational methods, we know that there are necessary constraints on the solution of Euler-Lagrange equations, which have to be fulfilled. Also the indispensable approximation of derivatives can cause problems at the image boundaries.

One the one hand, when working with graph cuts, such problems do not appear. The approximation of derivatives is not necessary and there are no constraints on the solution at the boundary that might cause problems. What has to be done is the extensive evaluation of the data and the smoothness term. This is the point where graph cuts might run into problems: It is an open question how to define the evaluation result of the data term for a pixel – disparity pair, that would point out of the image domain. For example in the left-to-right setting, consider the leftmost pixel in the left frame, located at position $p = (0, \cdot)$. For any disparity $d < 0$, the corresponding pixel in the right frame is at $p' = (0 + d, \cdot)$. Note that in the left-to-right setting all disparities must be non-positive. Thus, the pixel at

p' is not lying inside the right image domain.

In our implementation, we avoided this problem by just setting the energy of any such disparity field to infinity. The negative consequence from this decision is that all disparity fields have to approach disparity zero at the left or the right border. Especially, since every pixel must be assigned to a label, the leftmost pixel can only be mapped to disparity zero, the second pixel can only have disparity zero or one, etc. Obviously, this is not an ideal choice. But still the question which choice is ideal cannot be answered uniquely. In practice, the question is only of minor importance, since the occurence of artifacts at the image boundaries is not unusual. Besides problems related to boundary treatment, one has to keep in mind that the image information ends at the boundaries and that the correct correspondence for many pixels is just *invisible*. One could identify such pixels as being occluded by the image boundary.

6.1 Swap Moves

Like we did for image restoration, we will now construct an analogous 2-D graph for stereo vision having a very similar structure. Respecting the previously mentioned necessary adoptions for stereo vision, we can build up and cut the graph. Also a new labelling is read in the same way from a cut as before. Concerning the general swap move algorithm that controls how, when and in which order the swap moves are used to decrease the energy, there is just one minor difference to the image restoration application. The algorithm as presented in listing 4 does not stop until one cycle has been performed where none of the single swap moves could decrease the energy. But in practice, we found out that the first and partly the second cycle perform the major part of the energy decrease. All following cycles just lead to negligible further improvement, since only singular pixels will change their label, see figure 38. This is the reason why we stopped our algorithms after the second iteration in practice.

The implementation of all stereo vision related algorithms and methods was realised in ANSI C. For visualisation and testing, Dr. Andrés Bruhn provided his frontend [16] which was very helpful and simplified the evaluation significantly.

We evaluate the performance quantitatively in table 4. Figure 36 gives results for the swap move algorithm using different interaction penalties. Surprisingly, it becomes apparent that the swap move algorithm has poor performance for the non-truncated linear and quadratic interaction penal-

ties. The presentation of results for the image driven smoothness penalties is deferred to section 8 where also more advanced instances of this idea are examined.

The presented computation results show the applicability of the swap move algorithm. But we have to notice that that the performance of this method could be better, especially the non-truncated quadratic and linear penalties give unacceptably bad results. These results are not as we expected them to be, because swap moves were introduced in [13] without mentioning this drawback. When considering the truncated variants of these penalty terms, one should not forget that our label set is discrete and integer valued in our tests. Thus, a truncation level of 1.68 means that in practice only one non-zero disparity difference is not cut off. But we will see in the next section on expansion moves that this move performs very well.

	$BP_{1.0}$	$BP^{\text{non-occ}}$	$AADE$
$V^{\mathbf{quadr}}$	62.5	62.5	3.028
$V^{\mathbf{quadr}}_{1.68}$	8.15	6.16	0.53
$V^{\mathbf{quadr}}_{4.79}$	36.21	35.04	1.8
$V^{\mathbf{quadr}}_{10.26}$	56.26	55.64	2.651
$V^{\mathbf{linear}}$	30.77	29.47	1.642
$V^{\mathbf{linear}}_{1.68}$	8.15	6.16	0.53
$V^{\mathbf{linear}}_{4.79}$	27.11	25.7	1.435
$V^{\mathbf{linear}}_{10.26}$	30.64	29.34	1.631
$V^{\mathbf{lintrunc}}_{1.68}$	11.49	9.63	0.725
$V^{\mathbf{lintrunc}}_{4.79}$	24.89	23.44	1.37
$V^{\mathbf{lintrunc}}_{10.26}$	30.74	29.45	1.655
$V^{\mathbf{lintrunc}}_{20}$	30.62	29.33	1.643
$V^{\mathbf{Potts}}_{1.68}$	7.37	5.27	0.478
$V^{\mathbf{Potts}}_{4.79}$	5.73	3.59	0.328
$V^{\mathbf{Potts}}_{10.26}$	6.4	4.25	0.254
$V^{\mathbf{Potts}}_{20}$	5.22	3.04	0.199
$V^{\mathbf{Potts}}_{42.9}$	4.4	2.16	0.153
$V^{\mathbf{Potts}}_{83.5}$	4.56	2.39	0.192

Table 4: Error statistics for the **swap** move algorithm on the Tsukuba sequence. $BP^{\text{non-occ}}$ calculates the bad pixels error measure only in regions that are not marked as occluded by the ground truth occlusion map. All necessary images and occlusion maps are provided in [49, 47].

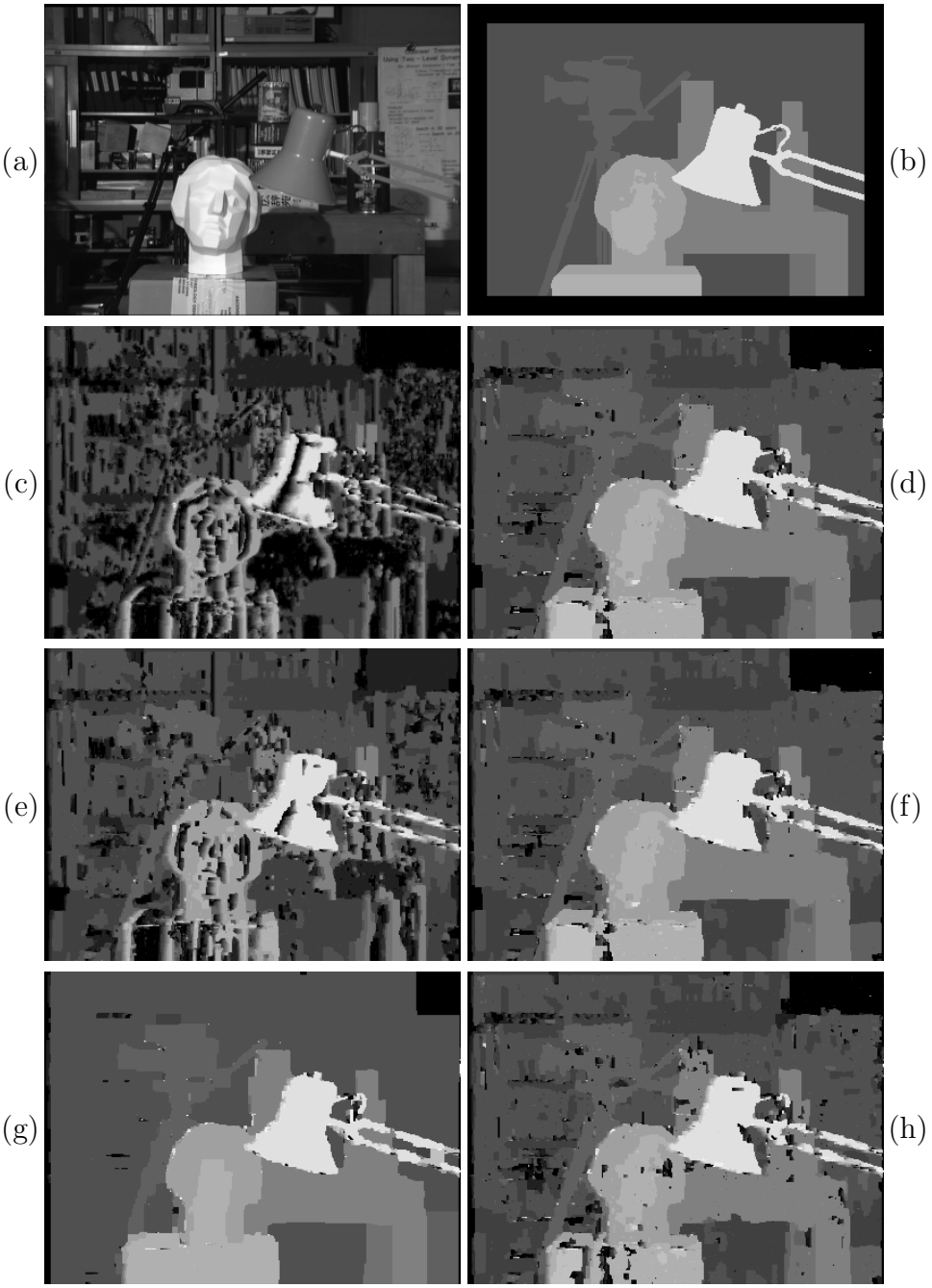


Figure 36: Results for **swap** move algorithm. Brighter colours mean larger disparity values. **(a)** Frame 3 of Tsukuba sequence [49]. **(b)** Ground truth disparity field between frame 3 and frame 6. **(c)** Obtained disparity field using $V^{\mathbf{quadr}}$ **(d)** Result using $V_{1.68}^{\mathbf{quadr}}$ **(e)** $V^{\mathbf{linear}}$ **(f)** $V_{1.68}^{\mathbf{linear}}$ **(g)** $V_{20}^{\mathbf{Potts}}$ **(h)** $V_{1.68}^{\mathbf{loglinear}}$.

6.2 Expansion Moves

Like for swap moves, the graph construction for expansion moves in stereo vision works in principle analogously to the image regularisation setting. We have to resemble the 2-D neighbourhood interactions, and insert auxiliary nodes whenever the labelling of two adjacent pixels varies. In practice, also because of their guarantee to find a solution near the global minimum, we predominantly used expansion moves for our computations. The drawback we have to accept is that we can work on a theoretically well-founded basis only with smoothness penalties V that satisfy all requirements of a metric, especially the triangle inequality (4.3). As a theoretical consequence, the quadratic penalty term cannot be used here. In practice however we show that even for semi-metrics the expansion move algorithm finds a very good local minimum – A minimum which is even remarkably better than the result of the swap move algorithm! We want to mention that more recent publications (e.g. [59] and references therein) investigate into this direction and especially develop methods that are able to compute solutions for convex semi-metrics with a guarantee of being within a certain distance from the global minimum. Concerning the computation times, using the push-relabel method our implementation needed about 50 seconds on an Intel Core 2 Duo with 2 GHz on the Tsukuba dataset. Note that for all graph cut algorithms treated in this thesis, the computation time depends linearly on the number of labels to compute. Thus, for other image pairs such as the Cones stereo sequence [49, 48] the computation times are significantly longer, since the computation has to deal with 60 instead of only 16 labels for the Tsukuba set. We did not use any parallelisation technique and our implementation was kept simple to be able to adapt it to several approaches. This is by far not the fastest implementation on can think of. We also did not implement the specialised max flow algorithm [10].

Like for the swap algorithm, we stopped expanding after the second cycle, although the expansion move algorithm from listing 5 terminates earlier than the swap algorithm, refer to figure 38.

In table 5 we give a quantitative evaluation of the performance of expansion moves for stereo reconstruction. Corresponding disparity images are given in figure 37.

With figure 39 we want to draw direct comparison of the results obtained

	$BP_{1.0}$	$BP^{\text{non-occ}}$	$AADE$
$V^{\mathbf{quadr}}$	6.3	4.4	0.435
$V^{\mathbf{quadr}}_{1.68}$	6.15	4.1	0.425
$V^{\mathbf{quadr}}_{4.79}$	5.08	3.06	0.39
$V^{\mathbf{quadr}}_{10.26}$	5.52	3.55	0.395
$V^{\mathbf{linear}}$	6.05	4.09	0.408
$V^{\mathbf{linear}}_{1.68}$	6.15	4.1	0.425
$V^{\mathbf{linear}}_{4.79}$	5.84	3.83	0.402
$V^{\mathbf{linear}}_{10.26}$	6.01	4.03	0.407
$V^{\mathbf{lintrunc}}_{1.68}$	6.12	4.07	0.433
$V^{\mathbf{lintrunc}}_{4.79}$	5.79	3.78	0.398
$V^{\mathbf{lintrunc}}_{10.26}$	5.89	3.88	0.4
$V^{\mathbf{lintrunc}}_{20}$	5.93	3.95	0.404
$V^{\mathbf{Potts}}_{1.68}$	7.1	5.02	0.468
$V^{\mathbf{Potts}}_{4.79}$	5.75	3.61	0.326
$V^{\mathbf{Potts}}_{10.26}$	6.41	4.27	0.254
$V^{\mathbf{Potts}}_{20}$	5.14	2.97	0.189
$V^{\mathbf{Potts}}_{42.9}$	4.33	2.15	0.152
$V^{\mathbf{Potts}}_{83.5}$	4.54	2.37	0.195
$V^{\mathbf{Potts}}_{1.68}$ SC	5.68	3.58	0.372
$V^{\mathbf{Potts}}_{4.79}$ SC	4.94	2.79	0.244
$V^{\mathbf{Potts}}_{10.26}$ SC	5.02	2.84	0.216
$V^{\mathbf{Potts}}_{20}$ SC	5.19	2.94	0.187
$V^{\mathbf{Potts}}_{42.9}$ SC	5.01	2.82	0.201
$V^{\mathbf{Potts}}_{83.5}$ SC	6.24	4.06	0.231

Table 5: Error statistics for the **expansion** move algorithm on the Tsukuba sequence. $BP^{\text{non-occ}}$ calculates the bad pixels error measure only in regions that are not occluded. The last six lines show results with enabled static cues. All necessary images and occlusion maps are provided in [49, 47]. Note that for the results obtained using semi-metrics, we do not have any optimality guarantee.

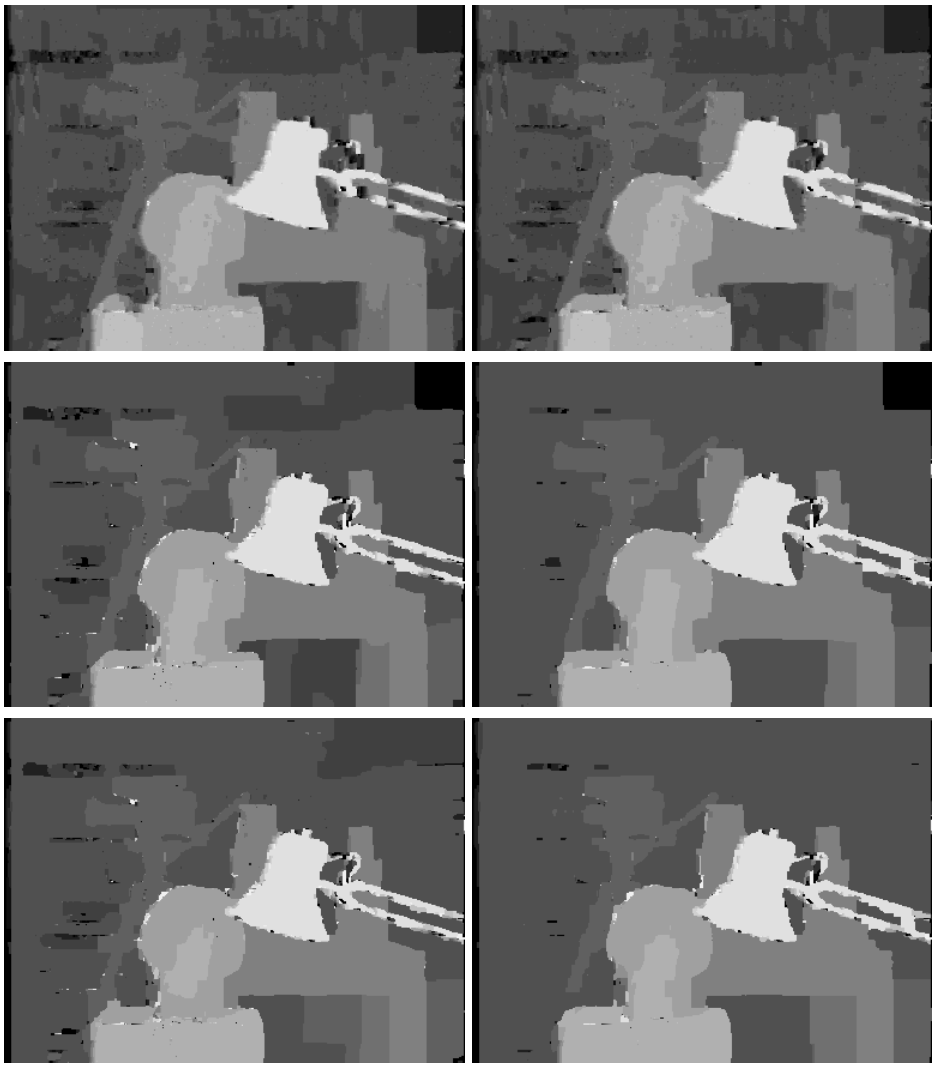


Figure 37: Results for **expansion** move algorithm. Note that even for the quadratic penalty the algorithm outperforms the swap method significantly. We also want to direct the readers attention to the effect of the static cues. Almost all methods not using this technique have problems in the upper right corner of the image. Image driven priors can help in such situations of low texture. **Top left:** Obtained disparity field using $V^{\mathbf{quadr}}$ **Top right:** $V^{\mathbf{linear}}$ **Centre left:** $V^{\mathbf{Potts}}_{4.79}$ **Centre right:** $V^{\mathbf{Potts}}_{20}$ **Bottom left:** $V^{\mathbf{Potts}}_{4.79}$ with static cues enabled. **Bottom right:** $V^{\mathbf{Potts}}_{20}$ with static cues enabled.

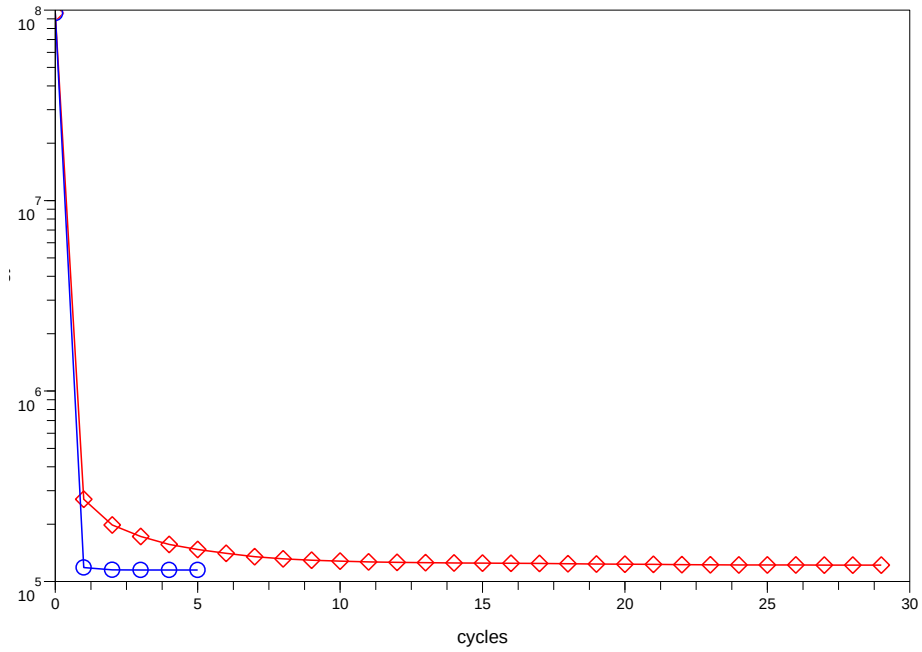


Figure 38: Energy value versus cycle number. Red line shows swap move algorithm. Blue line visualises expansion move results. The expansion move algorithm terminated regularly after the 6th iteration. Linear penalty V_{linear} used on Tsukuba sequence [49].

with swap and expansion moves with identical choice of parameters.

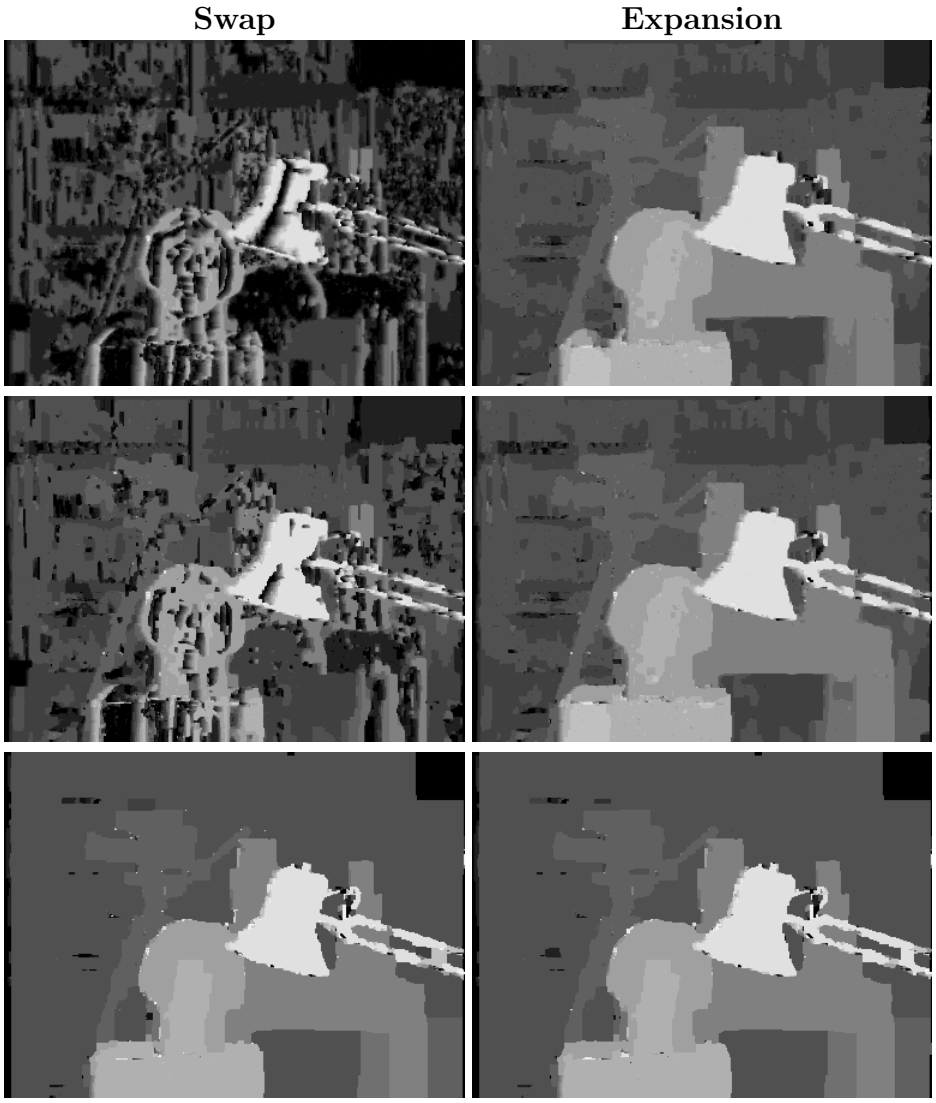


Figure 39: Comparison of swap and expansion moves. **Top row:** Quadratic penalty. **Middle row:** Linear penalty. **Bottom row:** Potts penalty, truncation level 20.

7 Incorporating Occlusions

Besides the classical approach discussed in the last section that used a straight forward and intuitive graph construction, we also implemented a more advanced approach [38]. In contrast to [13], which was also suited for image restoration, the approach we want to consider now is a specialised stereo vision method. The main difference is that the new graph construction will not associate pixels with graph nodes as before. Instead, the construction will set up one node for each possible assignment. The reason to do this is that by such constructions, both left and right frame can be treated symmetrically and special constraints can be enforced.

The main drawback of the classical approach from last section was that the notion of occlusions, which are frequently occurring image constellations, were not taken into consideration at all. There was no special treatment, and no countermeasures were taken. Moreover the main model assumption that each pixel has a corresponding pixel in the other frame is violated in occluded regions: a part of the scene which is occluded in one frame, cannot fulfil this assumption because there does not even exist a corresponding pixel. For this reason, we will now introduce a more elaborate construction being able to identify and treat such pixels adequately.

7.1 Occlusions

Figure 40 illustrates a common image scene where occlusions naturally occur. The left side of the loudspeaker is only visible in the left frame. Any method that assigns to each pixel in one frame a corresponding pixel in the other frame can only fail on this part of the image, since this underlying model assumption of the global existence of a corresponding visible pixel does not fit to the observed images. Hence we have to relax this constraint, and we will allow pixels to be not assigned to any other pixel. Such pixels will be called *occluded*.

Additionally, we will also enforce a uniqueness constraint, meaning that there might not exist two pixels in one frame being assigned to the same pixel in the other frame.

7.2 Problem Reformulation

Instead formulating the problem over pixels as before, we will consider all possible assignments between the two frames as quantity over which to min-

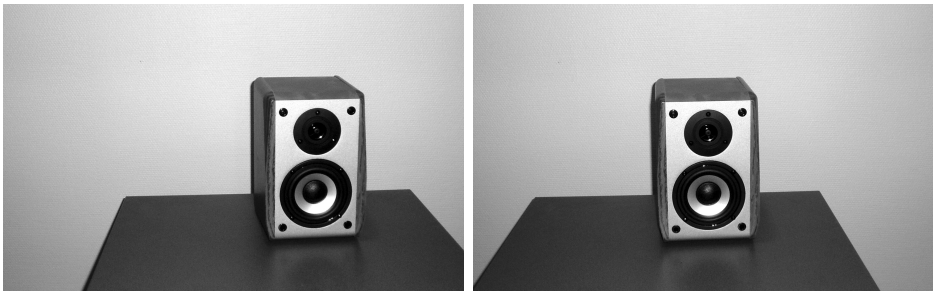


Figure 40: Real world example stereo images. The left side of the loudspeaker is only visible in the left image and all pixels in the left frame belonging to the side do not have an associated corresponding pixel in the other image.

imise. At some points it will be important to distinguish between pixels in the left and the right frame, which we will denote by $\mathcal{L}$ and $\mathcal{R}$, respectively. Further the set of all pixels will be combined into $\mathcal{P} = \mathcal{L} \cup \mathcal{R}$. Note that the previously used symbol $\mathcal{L}$ for the label set is not used for this method any more. Thus, in complete accordance to [38] we define the set of all possible assignments

$$\mathcal{A} = \{ < p, q > \mid p \in \mathcal{L}\,,\, q \in \mathcal{R}\,,\, p_y = q_y, 0 \leq q_x - p_x \leq k \,\}$$

In this definition we assume to work with the left-to-right ortho-parallel stereo setup and that all disparities that occur lie in the range $[0, k]$. Thus, disregarding pixels at the image boundary and assuming only integral disparities we could state the number of possible assignments as

$$|\mathcal{A}| = |\mathcal{L}| \cdot (k + 1)$$

since each pixel in the left frame can only correspond to at most $k+1$ pixels in the right frame.

Later, we will present a graph construction for expansion moves, which will *activate* assignments instead of assigning disparities to pixels. Of course there must be correspondence between these two algorithmic ideas, but to be able to identify occluded pixels we will need the possibility to deactivate all assignments for a pixel.

This leads us to the notion of activity of a pixel. Formally we define a configuration as a binary function $f : \mathcal{A} \to \{0, 1\}$ which is 1 if the assignment is active and 0 if the assignment is inactive. We can then define the set of all currently active assignments in the configuration f as

$A(f) = \{a \in \mathcal{A} | f_a = 1\}$ and $N_p(f)$ as the set of active assignments involving the pixel p in the given configuration. The uniqueness of a configuration can now be just expressed as the case where it holds that

$$|N_p(f)| \leq 1 \quad \forall p \in \mathcal{P}$$

which means that each pixel can just occur once in the set of active assignments. As already touched on, the criterion for a pixel p to be occluded is in this notion then just the question if $|N_p(f)| = 0$.

For several reasons, we also have to define a translation rule how to read the disparity from an assignment. To this end we define the disparity $d : \mathcal{A} \rightarrow \mathbb{R}$ as $d(a) = d(< p, q >) = q_x - p_x$. Note that those terms are all related to the ortho-parallel left-to-right stereo setup. Further, we define $\mathcal{A}^\alpha$ to be the set of all assignments resembling a pair of pixels having the disparity α.

With these notions at hand, a configuration f' is said to be within one (single) α-expansion of f, if $A(f') \subset A(f) \cup \mathcal{A}^\alpha$. This implies that any active assignment might become inactive and that any α-assignment can be activated. But it is not possible to activate any other assignment. If an assignment a is deactivated by an α-expansion, it might happen that by this deactivation no assignment incorporating this pixel p becomes active afterwards, i.e. $|N_p(f')| = 0$ then, and thus by an α-expansion pixels can become occluded.

Figure 41 gives a simplified example of such a set of assignments. The dashed lines are assignments that are not active. In this figure, the assignment $< b, g >$ has disparity 1, is currently active, and hence might become inactive by an expansion move for label 1. In the main reference

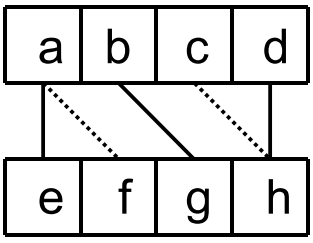


Figure 41: Exemplary situation for a 1-D signal of 4 pixels. $\mathcal{R} = \{a, b, c, d\}$, $\mathcal{L} = \{e, f, g, h\}$. Solid lines indicate active assignments, thus currently a and d have an active disparity of 0, b of 1, and the pixels c and f do not participate on any active assignment. This means that they are currently marked as occluded.

to this method [38], also a variant of α-β-swap moves is introduced, which we omit since Kolmogorov et al. already there annotated that it seems that swap moves are not powerful enough to find good local minima for the corresponding energy function, that we will now discuss in detail.

In order to be able to resemble the new class of occluded pixels in our energy, we have to introduce a new term to the existing smoothness prior and data term, a so-called occlusion penalty term E_{occ} as

$$E(f) = E_{\text{data}}(f) + E_{\text{occ}}(f) + E_{\text{smooth}}(f) \tag{7.1}$$

We want to stress here that we will always enforce the uniqueness of a configuration as a hard constraint that must be fulfilled. Formally it would be valid to add a term like $\sum_{p\in\mathcal{P}} \infty \cdot T(|N_p(f)| > 1)$, but since we will encode this constraint naturally into our graph construction, we can omit this term in (7.1).

The new occlusion term in this energy function is necessary in order to limit the number of pixels being marked as occluded, since the solution should label only as many pixels occluded as necessary. We use the following realisation for the occlusion penalty

$$E_{\text{occ}}(f) = \sum_{p\in\mathcal{P}} C_p \cdot T(|N_p(f)| = 0) \tag{7.2}$$

where $C_p > 0$ is the penalty for labelling pixel p occluded. Note that the summation over $\mathcal{P}$ already indicates the underlying symmetry of this approach, since this penalty is added for pixels from both, the left and the right frame.

The definition of the data term is relatively straight forward. Using the function $D : \mathcal{A} \to \mathbb{R}$ to represent the dissimilarity of the pixels of an assignment, $D(< p, q >) = (I_l(p) - I_r(q))^2$, we define

$$E_{\text{data}} = \sum_{a\in A(f)} D(a). \tag{7.3}$$

The smoothness term for this formulation over assignments is the most problematic part of this energy function. First, we observe that the neighbourhood set we used in the previous sections is not valid here any more. We need to define some neighbourhood relation for assignments, i.e.

$$\mathcal{N} \subset \{\{a_1, a_2\} \mid a_1, a_2 \in \mathcal{A}\}$$

Then the most logical choice for the smoothness term would be

$$E_{\text{smooth}} = \sum_{\substack{\{a_1,a_2\}\in\mathcal{N} \\ a_1,a_2\in A(f)}} V(a_1,a_2) \tag{7.4}$$

which would sum up all variations of active assignments that are neighbours. A suitable neighbourhood relation would then be a relation that only includes assignments of spatially adjacent pixels, that have *different* disparities. Kolmogorov et al. showed in [38] that the computation of just one α-expansion of an energy of this form is NP-hard. This theory was also generalised in a successful publication [40], where generic graph constructions for certain classes of energy function are presented.

Thus we have to find another formulation for the smoothness term. This formulation uses another neighbourhood relationship, which only incorporates assignments that resemble the *same* disparity. Thus we can define formally for $a =< p, q >$ and $a' =< p', q' >$

$$\mathcal{N} = \{\{a, a'\} \mid (\{p, p'\} \in \mathcal{N}^* \vee \{q, q'\} \in \mathcal{N}^*) \wedge d(a) = d(a')\} \tag{7.5}$$

where $\mathcal{N}^*$ denotes the known spatial neighbourhood relation for pixels. With this new neighbourhood system at hand, we can define

$$E_{\text{smooth}} = \sum_{\{a_1,a_2\}\in\mathcal{N}} V(a_1,a_2) \cdot T(f(a_1) \neq f(a_2)) \tag{7.6}$$

Using this formulation for a smoothness term, we get a contribution whenever one of two spatially adjacent assignments is active and the other is not. We can understand this choice as the Potts model over assignments.

7.3 Graph construction

The overall algorithm that we will use to compute a local minimum using this method is essentially the same as the expansion move algorithm from listing 5 in section 3. We will now define how to compute one α-expansion for a given labelling f and label α. The algorithm will pick a label α each time and compute one expansion move for this label. If this decreases the total energy, it will accept this new labelling and go on from there. As before, instead of doing this loop until no label can decrease the energy further, we will stop after a certain number of iterations, because the changes just affect single pixels.

We will construct a graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ with two terminal vertices s and t. A cut on this graph is a set of edges $\mathcal{C} \subset \mathcal{E}$ such that the terminals are separated in the induced graph $\mathcal{G}^{\mathcal{C}} =< \mathcal{V}, \mathcal{E} \setminus \mathcal{C} >$. Equivalently we can consider the two disjoint sets of separated sub-graphs by this cut, $\mathcal{V}^s$ and $\mathcal{V}^t$. All nodes in $\mathcal{V}^s$ are reachable from s in $\mathcal{G}^{\mathcal{C}}$ and all vertices in $\mathcal{V}^t$ from t. Thus we will now try to set up a graph on which a cut coincides to a labelling. Hence if we can determine a minimum cut, we should also have found a labelling with minimum energy.

Given a label α and a current configuration or labelling f^0 the graph construction is as follows: The graph will consist of the two terminals s and t and one vertex corresponding to each assignment being included in the set

$$\tilde{A} = \mathcal{A}^0 \cup \mathcal{A}^\alpha$$

where

$$A^0 = \{a \in \mathcal{A}(f^0) \mid d(a) \neq \alpha\}$$

is the set of all assignments that are currently active, but that do not have disparity α, for short: all active other assignments. Further, we need to define

$$\mathcal{A}^\alpha = \{a \in \mathcal{A} \mid d(a) = \alpha\}$$

in this formulation. It resembles all α-assignments.

Before we give the edges and their weight, and in order to understand more intuitively the system behind this construction, we will briefly explain the so-called *voxel-labelling* idea. Kolmogorov et. al published in 2005 a book chapter on stereo vision [41]. There they completely re-motivated their approach [38] from 2001 that we are currently discussing here, using the voxel labelling idea. In essence, the graphs of those those two approaches completely coincide, only the explanation and their motivation is more intuitive and can be illustrated nicely, which is done in figure 42. Given a labelling or configuration f, one can understand each assignment as a voxel whose *depth* is related to its disparity.

In essence, we will now *wire* all voxels as illustrated accordingly to a logical system. The important point there is, that a cut on such a graph will coincide with a labelling. In contrast to the previous constructions, we will have to insert one directed edge into this graph and also the rule how to read a labelling from a graph will be a bit more difficult. To really be able to explain the construction, we have to repeat the figure 42 with additionally marked voxels for an α-expansion to make clear the important

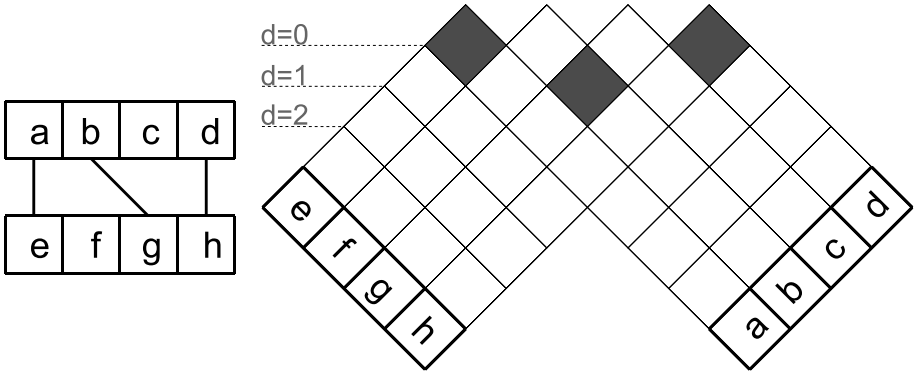


Figure 42: Voxel la being idea. Each solid assignment line in the left assignment visualisation coincides with a dark grey marked diamond shaped voxel in the right figure. The different disparity levels are marked there.

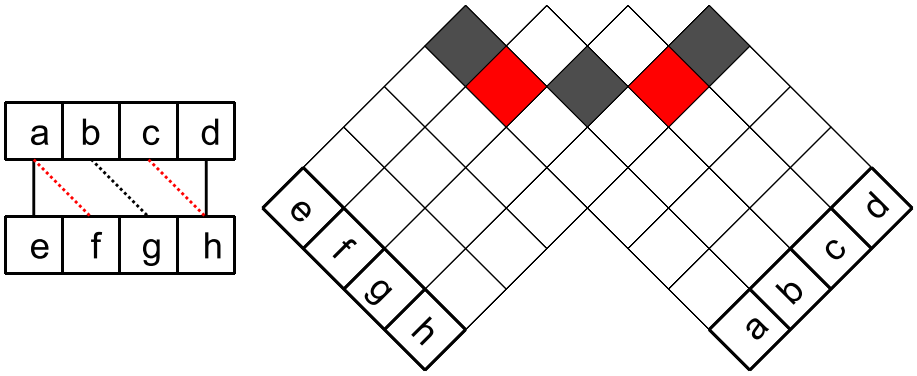


Figure 43: Graph for voxel labelling expansion algorithm for expansion of label $\alpha = 1$. The solid lined assignments belong to $\mathcal{A}^0$. The dashed assignments belong to $\mathcal{A}^\alpha$. Red colour indicates that this assignment is not active at present.

points. This is done in figure 43. There, one can see that for an expansion of label 1, the graph would consist of 5 assignment nodes. Note that for the label 1, not more assignments than currently marked are possible, because any other assignment with disparity 1 would point out of the images.

Thus, each pixel will be involved at least in one $\mathcal{A}^\alpha$ assignment – if this is possible. For example pixel e cannot take part in an 1-expansion, due to its position at the boundary. Next, if for a pixel the currently expanded

label is not active, then this pixel will be involved in two expansions, which is a frequently occurring case (e.g. pixels a and h).

The list of all edges and corresponding weights is presented in table 6. If

edge	**weight**	**for**
(s, a)	$D_{\text{occ}}(a)$	$a \in \mathcal{A}^0$
(a, t)	$D_{\text{occ}}(a)$	$a \in \mathcal{A}^\alpha$
(a, t)	$D(a) + D_{\text{smooth}}(a)$	$a \in \mathcal{A}^0$
(s, a)	$D(a)$	$a \in \mathcal{A}^\alpha$
$\substack{(a_1,a_2)\\(a_2,a_1)}$	$V(a_1, a_2)$	$\substack{\{a_1,a_2\}\in\mathcal{N}\\ a_1,a_2\in\tilde{A}}$
(a_1, a_2)	∞	$\substack{p\in\mathcal{P},a_1\in\mathcal{A}^0,a_2\in\mathcal{A}^\alpha\\ a1,a2\in N_p(\tilde{f})}$
(a_2, a_1)	C_p	

Table 6: Voxel labelling edge list with corresponding capacities. To identify the edges later on we will call all edges corresponding to the first two lines *t-links*. Further the edge with weight C_p will be called a *c-link*.

a graph consisting of nodes as given previously and edges as given in table 6 is cut into the two disjoint sets of vertices $\mathcal{V}^s$ and $\mathcal{V}^t$, we can define how to read a new configuration from this cut as follows:

$$\begin{aligned} \forall a \in \mathcal{A}^0 \quad & f_a^{\mathcal{C}} = \begin{cases} 1, & \text{if } a \in \mathcal{V}^s \\ 0, & \text{if } a \in \mathcal{V}^t \end{cases} \\ \forall a \in \mathcal{A}^\alpha \quad & f_a^{\mathcal{C}} = \begin{cases} 1, & \text{if } a \in \mathcal{V}^t \\ 0, & \text{if } a \in \mathcal{V}^s \end{cases} \\ \forall a \notin \tilde{A} \quad & f_a^{\mathcal{C}} = 0 \end{aligned} \tag{7.7}$$

In order to really understand why this construction does what we want, we give with figure 44 one additional illustration on the standard setting where two assignments $a_1 \in \mathcal{A}^0$ and $a_2 \in \mathcal{A}^\alpha$ are involved with the same pixel p, i.e. $a_1, a_2 \in N_p(\tilde{f})$. In this sketch we see the two terminals and the two assignments, which shall concern the same pixel as described. With the given edges and weights, we get the illustrated situation. Any valid cut must separate the terminals and hence for this constellation, we have

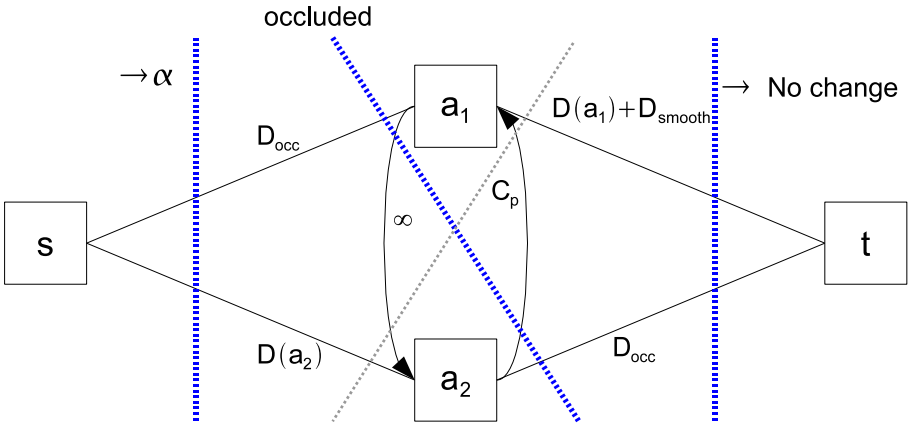


Figure 44: Situation for one pixel: Wiring of the two assignments $a_1, a_2 \in N_p(\tilde{f})$ where $a_1 \in \mathcal{A}^0$ and $a_2 \in \mathcal{A}^\alpha$.

4 possibilities to realise this separation, indicated by the dashed lines. The grey line is not a valid cut in that sense, that its capacity is infinite, since it contains the edge (a_1, a_2) which has capacity ∞. Thus there are 3 possibilities left how to cut this situation. The diagonal cut would deactivate both assignments and leave the pixel occluded, the other two possibilities would either activate the α-assignment or the previously active assignment.

Thus, each node will have a link to the source and a link to the sink. We have seen what is happening between nodes that concern the same pixel. What edges are still missing are the neighbourhood links from the fifth line in table 6. Since we defined our neighbourhood system to only contain assignments having equal disparities, these edges will only connect horizontally, on the same disparity level. Note that the edge exists independently of the activity of the two concerned assignments.

Further, we have to define the occlusion term for an assignment

$$D_{\text{occ}}(a) = D_{\text{occ}}(< p, q >) = D_{\text{occ}}(p) + D_{\text{occ}}(p) \tag{7.8}$$

where the occlusion term per pixel is defined to be either $D_{\text{occ}}(p) = C_p$ if there is only one assignment among the currently considered assignments $\tilde{A}$ accounting to p, and $D_{\text{occ}}(p) = 0$ else. Also the smoothness cost used in table 6 still needs to be specified. We say that is sums up the smoothness terms of all neighbouring assignments being not part of $\tilde{A}$, but with the

same disparity.

$$D_{\text{smooth}}(a_1) = \sum_{\substack{\{a_1,a_2\}\in\mathcal{N} \\ a_2\notin\tilde{A}}} V(a_1, a_2) \tag{7.9}$$

Again, we have to redefine the smoothness term $V(a_1, a_2)$, because if we took the known terms, we would not get any contribution: With the first requirement to semi-metrics (4.1) we forced that if the two labels are equal, then the contribution must be zero as well. The previously mentioned Potts interaction idea is in this graph construction encoded in the graph itself and not included in the smoothness penalty term V. Here we just reuse the idea of an image driven prior to get for two neighbouring assignments $a_1 =< p, q >$ and $a_2 =< r, s >$

$$V(a_1, a_2) = \begin{cases} \lambda, & \text{if } \max\{|I_l(p) - I_l(r)|, |I_r(q) - I_r(s)|\} > 8 \\ 3\lambda, & \text{else} \end{cases} \tag{7.10}$$

With this formulation we enforce more smoothness if the image intensities are similar. Kolmogorov et. al. proposed to use parameter values $\lambda = 3$ and $C_p = 3\lambda$, but we found out that the method is quite stable with respect to the choice of these variables. This is also the reason why we decided to just take the choice for $V(a_1, a_2)$ as proposed. Working at these parameters will not give tremendous improvements, and thus we just have to alter the single parameter λ.

Please keep in mind that every expression here that considers pixels $p \in \mathcal{P}$ always has to be evaluated twice, once for the left and once for the right frame.

Moreover, please note that every configuration arising form this method is symmetric, in that sense that a new labelling in the right-to-left setup can be just translated *uniquely* to the left-to-right setup. But again, an important restriction is that not every pixel is mapped to a disparity value. Occluded pixels are not labelled at all. But this proceeding of not assigning a label to an occluded pixel at all, is in principle the only correct possibility, because there cannot exist a corresponding pixel in the other frame.

7.4 Optimality Considerations

In the main reference to this method [38] a proof is given showing that the cost of a cut can be related to the energy given in (7.1). We want to rephrase this proof here now, giving some more details.

Let $\mathcal{G}$ be a graph that has been constructed according to the just given rules for an input configuration f and the expansion of the label α. Further let $\mathcal{C}$ be a cut on $\mathcal{G}$.

After having specified the graph construction and after having fixed the rules how to relate a new configuration to a cut on this graph, we obviously can state that any such new configuration must lie within one α-expansion of the input configuration.

The next important appraisal relates uniqueness and finiteness of configurations and cuts.

Lemma The finiteness of the cost of the cut $\mathcal{C}$ is equivalent to the uniqueness of the configuration $f^{\mathcal{C}}$.

Proof If the cost of the cut was infinite, then one of the directed edges with cost ∞ must be part of the cut. The situation is also illustrated in figure 44, where the dashed line in grey colour represents this cut. Reusing without loss of generality the notion used in the mentioned sketch, we have that $a_1 \in \mathcal{A}^0$ and $a_2 \in \mathcal{A}^\alpha$. One can see that if the ∞-edge is part of the cut, it must hold that $a_1 \in \mathcal{V}^t$ and $a_2 \in \mathcal{V}^s$. Translating this situation into a configuration, we get that both assignments would be active, which means that the configuration is not unique.

The proof of the opposite way of the implication is very similar. If the $f^{\mathcal{C}}$ is not unique, both assignments must be active, and hence it must again hold that $a_1 \in \mathcal{V}^t$ and $a_2 \in \mathcal{V}^s$. Obviously this is only possible if the ∞-edge is cut. Thus the cost must be infinite. □

We will split the reasoning why the cost of a cut is equal to the energy of the arising labelling in two parts. For this let $f^{\mathcal{C}}$ be unique. The first part is a

Lemma The cost of all c-links a and all t-links equals $E_{\text{occ}}(f^{\mathcal{C}})$ plus a constant.

Proof Given the cut $\mathcal{C}$ and the corresponding partition of the node set into $\mathcal{V}^s$ and $\mathcal{V}^t$, we can calculate the cost of the t-links by simply summing up

$$\sum_{a \in \mathcal{A}^0} D_{\text{occ}}(a) \cdot T(a \in \mathcal{V}^t) + \sum_{a \in \mathcal{A}^\alpha} D_{\text{occ}}(a) \cdot T(a \in \mathcal{V}^s) \tag{7.11}$$

Similarly, the c-links can only contribute to the cost if they are part of the cut, i.e. if the corresponding pixel is marked as occluded. Hence we get for the cost of the c-links

$$\sum_{\substack{p\in\mathcal{P},a_1\in\mathcal{A}^0,a_2\in\mathcal{A}^\alpha \\ a_1,a_2\in N_p(\tilde{f})}} C_p \cdot T\left(a_1 \in \mathcal{V}^t \wedge a_2 \in \mathcal{V}^s\right) \tag{7.12}$$

On the other hand we defined in equation (7.2)

$$E_{\text{occ}}(f) = \sum_{p\in\mathcal{P}} C_p \cdot T(|N_p(f^{\mathcal{C}})| = 0)$$

This summation can be split up by the number of involved assignments per node. Thus the complete occlusion energy is a constant plus

$$\sum_{\substack{p\in\mathcal{P} \\ |N_p(\tilde{f})|=1}} C_p \cdot T(|N_p(f^{\mathcal{C}})| = 0) + \sum_{\substack{p\in\mathcal{P} \\ |N_p(\tilde{f})|=2}} C_p \cdot T(|N_p(f^{\mathcal{C}})| = 0) \tag{7.13}$$

The two terms are not perfectly identical because at the boundaries of the images it might occur that there are pixels with $|N_p(\tilde{f})| = 0$, but their contribution can be bounded by a constant. For the first term of the last equation, we know that either an assignment $a \in \mathcal{A}^0$ or an α-assignment is concerned, thus

$$\begin{aligned} &\sum_{\substack{p\in\mathcal{P} \\ |N_p(\tilde{f})|=1}} C_p \cdot T(|N_p(f^{\mathcal{C}})| = 0) \\ = &\sum_{\substack{p\in\mathcal{P} \\ N_p(\tilde{f})=\{a\}\subset\mathcal{A}^0}} C_p \cdot T(a \notin A(f^{\mathcal{C}})) + \sum_{\substack{p\in\mathcal{P} \\ N_p(\tilde{f})=\{a\}\subset\mathcal{A}^\alpha}} C_p \cdot T(a \notin A(f^{\mathcal{C}})) \end{aligned} \tag{7.14}$$

and the second term in (7.13) is just the standard situation

$$\begin{aligned} &\sum_{\substack{p\in\mathcal{P} \\ |N_p(\tilde{f})|=2}} C_p \cdot T(|N_p(f^{\mathcal{C}})| = 0) \\ = &\sum_{\substack{p\in\mathcal{P}a_1\in\mathcal{A}^0,a_2\in\mathcal{A}^\alpha \\ a_1,a_2\in N_p(\tilde{f})}} C_p \cdot T(a1, a2 \notin A(f^{\mathcal{C}})) \end{aligned} \tag{7.15}$$

Using equation (7.8) to translate from C_p to $D_{\text{occ}}(p)$ we have everything at hand to rewrite (7.13) as

$$\begin{aligned} &\sum_{\substack{p\in\mathcal{P}\\ N_p(\tilde{f})=\{a\}\subset\mathcal{A}^0}} D_{\text{occ}}(p)\cdot T(a\notin A(f^{\mathcal{C}})) \\ +&\sum_{\substack{p\in\mathcal{P}\\ N_p(\tilde{f})=\{a\}\subset\mathcal{A}^\alpha}} D_{\text{occ}}(p)\cdot T(a\notin A(f^{\mathcal{C}})) \\ +&\sum_{\substack{p\in\mathcal{P}a_1\in\mathcal{A}^0,a_2\in\mathcal{A}^\alpha\\ a_1,a_2\in N_p(\tilde{f})}} C_p\cdot T(a1,a2\notin A(f^{\mathcal{C}})) \end{aligned} \tag{7.16}$$

Comparing the latter result with equations (7.11) and (7.12) shows that the stated holds. □

Lemma If $\mathcal{C}$ is a minimal cut on $\mathcal{G}$, then $f^{\mathcal{C}}$ is unique and minimises E.

Proof We have shown the uniqueness (with respect to assignments to pixels), and with the last lemma it is sufficient to proof the equality of the cost of the remaining edges and the remaining parts of the energy function E_{data} and E_{smooth}. Thus we have

$$\begin{aligned} &E_{\text{data}}(f^{\mathcal{C}})+E_{\text{smooth}}(f^{\mathcal{C}})= \\ &\sum_{a\in A(f)} D(a)+\sum_{\{a_1,a_2\}\in\mathcal{N}} V(a_1,a_2)\cdot T(f(a_1)\neq f(a_2)) \end{aligned} \tag{7.17}$$

Our graph construction was focussed on a subset of all possible assignments $\tilde{A}\subset\mathcal{A}$. Using the fact that we have no contributions from assignments $a\in\mathcal{A}\setminus\tilde{A}$, we can rewrite the second part of (7.17) as

$$\sum_{\substack{\{a_1,a_2\}\in\mathcal{N}\\ a_1,a_2\in\tilde{A}}} V(a_1,a_2)\cdot T(f(a_1)\neq f(a_2)) + \sum_{\substack{\{a_1,a_2\}\in\mathcal{N}\\ a_1\in\tilde{A},a_2\notin\tilde{A}}} V(a_1,a_2)\cdot T(f(a_1)\neq f(a_2)) \tag{7.18}$$

On the other hand, considering the cost of the cut $\mathcal{C}$, and ignoring the costs of the c-links and t-links, we can state

$$\begin{aligned} &\textstyle\sum_{a\in\mathcal{A}^0}(D(a)+D_{\text{smooth}}(a))\cdot T(a\in\mathcal{V}^s) \\ &\textstyle+\sum_{a\in\mathcal{A}^\alpha} D(a)\cdot T(a\in\mathcal{V}^t) \\ +&\textstyle\sum_{\substack{\{a_1,a_2\}\in\mathcal{N}\\ a_1,a_2\in\tilde{A}}} V(a_1,a_2)\cdot T\left((a_1\in\mathcal{V}^s,a_2\in\mathcal{V}^t)\vee(a_1\in\mathcal{V}^t,a_2\in\mathcal{V}^s)\right) \end{aligned} \tag{7.19}$$

For the first two terms of latter equation we can get rid of the indicator function by rewriting them as

$$\sum_{a \in \mathcal{A}^0 \cap A(f^{\mathcal{C}})} (D(a) + D_{\text{smooth}}(a)) + \sum_{a \in \mathcal{A}^\alpha \cap A(f^{\mathcal{C}})} D(a)$$

The argument of the indicator function in the third term of (7.19) can also be simplified significantly by

$$\sum_{\substack{\{a_1,a_2\} \in \mathcal{N} \\ a_1,a_2 \in \tilde{A}}} V(a_1, a_2) \cdot T\left(f^{\mathcal{C}}_{a_1} \neq f^{\mathcal{C}}_{a_2}\right)$$

The data term contributions just fit together and finally coincide with what we computed in (7.17). In the same way we see that the last equation coincides with the first part of (7.18). Hence, what remains is to show that the smoothness penalties are equal. Thus we want to show that

$$\sum_{\substack{\{a_1,a_2\} \in \mathcal{N} \\ a_1 \in \tilde{A}, a_2 \notin \tilde{A}}} V(a_1, a_2) \cdot T(f(a_1) \neq f(a_2)) = \sum_{a \in \mathcal{A}^0 \cap A(f^{\mathcal{C}})} D_{\text{smooth}}(a)$$

We can assume for the summation of the first term, that $a_1 \in \mathcal{A}^0$. Because if this was not the case, i.e. if $a_1 \in \mathcal{A}^\alpha$, then it would follow that a_2 is element of $\mathcal{A}^\alpha$ too, because the assignments are neighbours, and we imposed a neighbourhood system that only contains assignments with equal disparity. In this case, we would violate the condition that $a_2 \notin \tilde{A}$, thus we know that $a_1 \in \mathcal{A}^0$.

With this detail and the definition of D_{smooth} from (7.9) we get their equality immediately. □

Thus we have proven that a minimum cut on the given graph construction finds the labelling within one α-expansion with least energy.

Dense disparity In many cases, it is not desired to have occlusion *holes* in the disparity map, where the algorithm does not establish a pixel correspondence. As an example, when trying to perform backward registration [43] or when trying to compare the algorithms in the on-line evaluation form [47, 48, 49], 100% dense disparity maps are required.

What one can do to get a 100% dense disparity also in occluded regions is a trick: Assume a pixel that is marked as occluded. Usually such pixels

are detected in positions where the depth of the scene changes abruptly. Hence to one side of this occlusion, we will find a foreground pixel, and to the other side a background pixel. The idea now is to fill-in the disparity of the background, because the pixels that have no correspondence in the other frame can just belong to the background. The foreground that caused the occlusion must still be visible. At which side foreground and background must be located just depends on whether we are computing the left-to-right or the right-to-left setting. For left-to-right computations, the background is always to the left of the occlusion. Thus what we can do is just searching along the same scanline in left direction until the first non-occluded pixel is found and insert its disparity to all traversed occluded pixels.

Again we want to refer to the original publications of this approach [38, 41] where all ideas we presented in this section are taken from. Interestingly there is a revised version of [38], unfortunately not officially published, that re-motivates the graph construction systematically following the generic construction in [40]. But this new motivation does not alter the effective graph construction, the generic graph from [40] only generalises this construction to a wider formal class of energy functions.

7.5 Results

For our results we started with a configuration f where all assignments are inactive at the beginning and we travelled twice through all labels α in ascending order.

Still a problem to be discussed are quantitative error statistics. As mentioned, we do not output a disparity value for each pixel. This gives rise to several possible ways how to evaluate the bad pixels measure.

- **Standard bad pixels:** $BP^{\mathbf{standard}}$ does only get a ground truth disparity map but no ground truth occlusion map. Hence we get a contribution wherever the disparity does not identify an occluded pixel.

- **Strict non-occluded bad pixels:** $BP^{\mathbf{strict}}$ For this measure, we fill in disparities from the background first and then compute the standard measure which will then give a contribution in every pixel. No occlusion maps are considered.

- **Strict bad pixels (with occlusion map):** $BP^{\mathbf{strict\ non\text{-}occ}}$ Given a ground truth, we proceed as follows: If the ground truth votes for occlusion but the computed disparity not, then this is taken as a

bad pixel. Vice versa, if the computed disparity gives a false positive occlusion, then this also is taken as a bad pixel. But, in this setting, if both ground truth and disparity detect an occlusion in a pixel, then this must count as a correctly detected occlusion and no bad pixel.

- **Relaxed bad pixels (with occlusion map):** $BP^{\textbf{relaxed non-occ}}$ only takes a contribution from a pixel if both ground truth and computed disparity are not occluded.

All necessary images and occlusion maps to evaluate our results were taken from [49, 48, 47]. For an overview over the performance of this method we give table 8 and figure 45. We found out that the algorithm is extraordinarily stable with respect to the main model parameter λ. This behaviour is illustrated in table 7.

	$BP^{\textbf{relaxed non-occ}}$
$\lambda = 1$	1.768
$\lambda = 3$	1.057
$\lambda = 10$	1.395
$\lambda = 50$	6.545

Table 7: Error evaluation for variations of the model parameter λ of the method by Kolmogorov and Zabih [38] on the Tsukuba image set. The static cues factor was $\delta = 3$ in all cases.

$\lambda = 3$	$BP^{\mathbf{standard}}$	$BP^{\mathbf{strict}}$	$BP^{\mathbf{strict\ non\text{-}occ}}$	$BP^{\mathbf{relaxed\ non\text{-}occ}}$	$AADE$
Tsukuba	1.587	2.124	2.518	1.057	0.145
Venus	1.203	1.298	2.076	0.956	0.320
Teddy	7.274	9.876	7.234	5.948	0.722
Cones	14.448	19.095	16.513	11.725	1.303

Table 8: Error statistics for the method by Kolmogorov and Zabih [38]. The static cues factor was $\delta = 3$ in all cases.

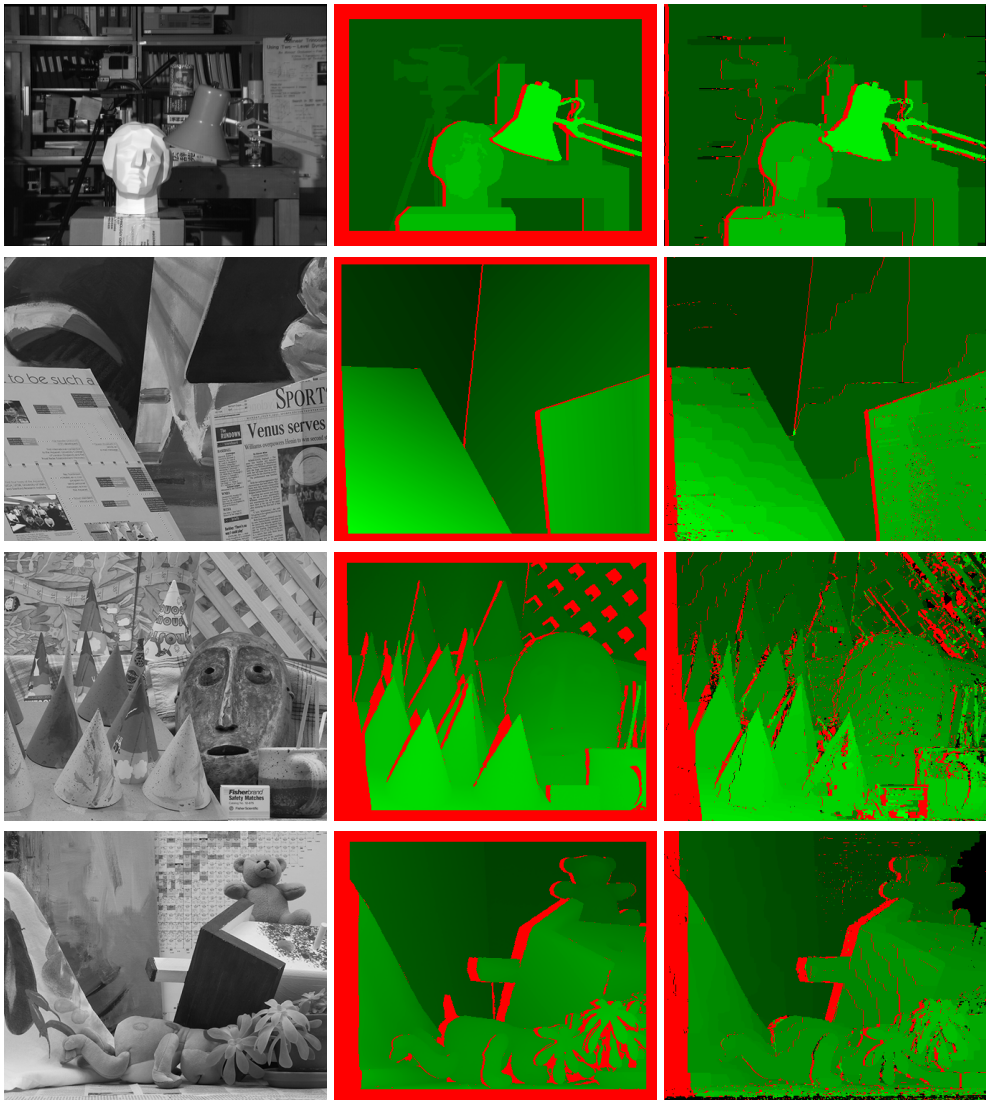


Figure 45: Results for the important test images from [49, 48, 47]. **Left column:** Left input frame. **Middle column:** Ground truth with occlusions in red. **Right column:** Computation results for the presented algorithm. **Top to bottom:** Tsukuba, Venus, Cones and Teddy sequence. Quantitative error statistics can be looked up in table 8

8 Extensions

In sections 6 and 7 we have discussed and reimplemented two successful graph cut approaches to the stereo vision problem. So far we have not supplemented new ideas. This shall be the topic of this part of this document.

8.1 Advanced static cues

It is possible to exchange the hard thresholding for the image driven term in [13] by the application of a Perona-Malik type decreasing function. Instead of the formulation from equation (6.7)

$$V_{pq}(f_p, f_q) = T(f_p \neq f_q) \cdot \begin{cases} K & , \text{ if } |I_p - I_q| \geq 5 \\ \delta \cdot K & , \text{ else} \end{cases}$$

we decided to use instead the continuous function [44]

$$g(s^2) = \frac{1}{1 + \frac{s^2}{\lambda^2}}. \tag{8.1}$$

This decreasing function was then applied to suppress the influence of the smoothness prior at edge positions

$$V_{pq}(f_p, f_q) = g(|I_p - I_q|^2) \cdot K \cdot T(f_p \neq f_q). \tag{8.2}$$

Note that this formulation gives at most the value K as smoothness penalty, instead of adding more weight to the prior as proposed in [13]. This means that the smoothness term can become arbitrarily close to zero in this formulation. The used function is plotted in figure 46. For computation results we refer to table 9 and figure 47.

	$BP^{\textbf{non-occ}}$ with static cues	$BP^{\textbf{non-occ}}$ without static cues
$V^{\textbf{quadr}}$	3.358	4.491
$V^{\textbf{linear}}$	3.45	4.06
$V^{\textbf{Potts}}$	2.467	2.967

Table 9: Error evaluation for the expansion move algorithm on the Tsukuba sequence.

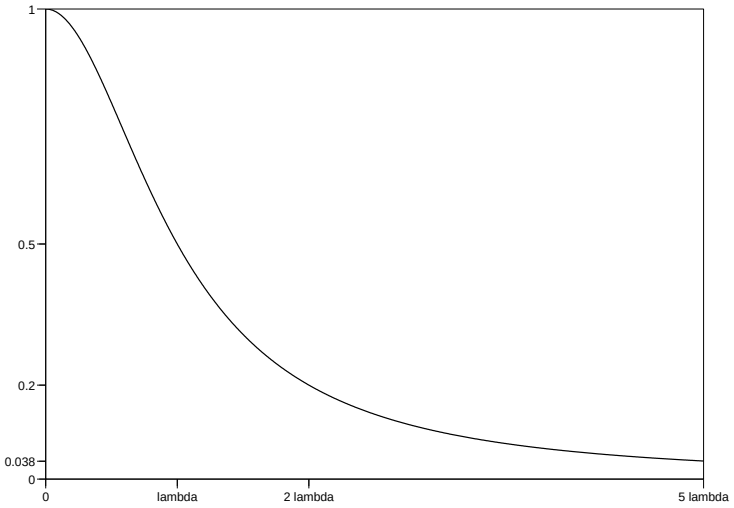


Figure 46: Penalisation function proposed by Perona and Malik [44]. In practice we can choose the parameter λ in such a way, that the smoothness penalty is never set to zero completely, but still decreased significantly.

We only applied this modified image driven term to the classical approach from section 6 and did not extend the method from section 7, because the specialised graph structure of the latter methods limits the degrees of freedom for changes and extensions considerably.

8.2 Extended Neighbourhood System

Inspired by literature on graph cut based image segmentation [11], we have extended the underlying neighbourhood model from a 4-connected to an 8-connected neighbourhood system. This extension is relatively simple, since we only have to consider 4 additional neighbour pixels in all terms. Moreover, assuming a horizontal and vertical grid size of 1, we have to approximate the derivatives in diagonal direction correctly: If the grid has quadratic and non-skewed shape, we get a diagonal grid size of $\sqrt{2}$. Thus, besides the sum over all smoothness contributions from neighbouring pixels from equation (3.19), we also have to carefully adjust the weighting of the image driven parts of the energy function. Instead of just thresholding the absolute value of the grey value difference of neighbouring pixels, we also have to respect the diagonal grid size. In practice, we found that the minimisation process converges slightly slowlier. Instead two cycles, we used three cycles of the expansion algorithm. This lead to the results listed in

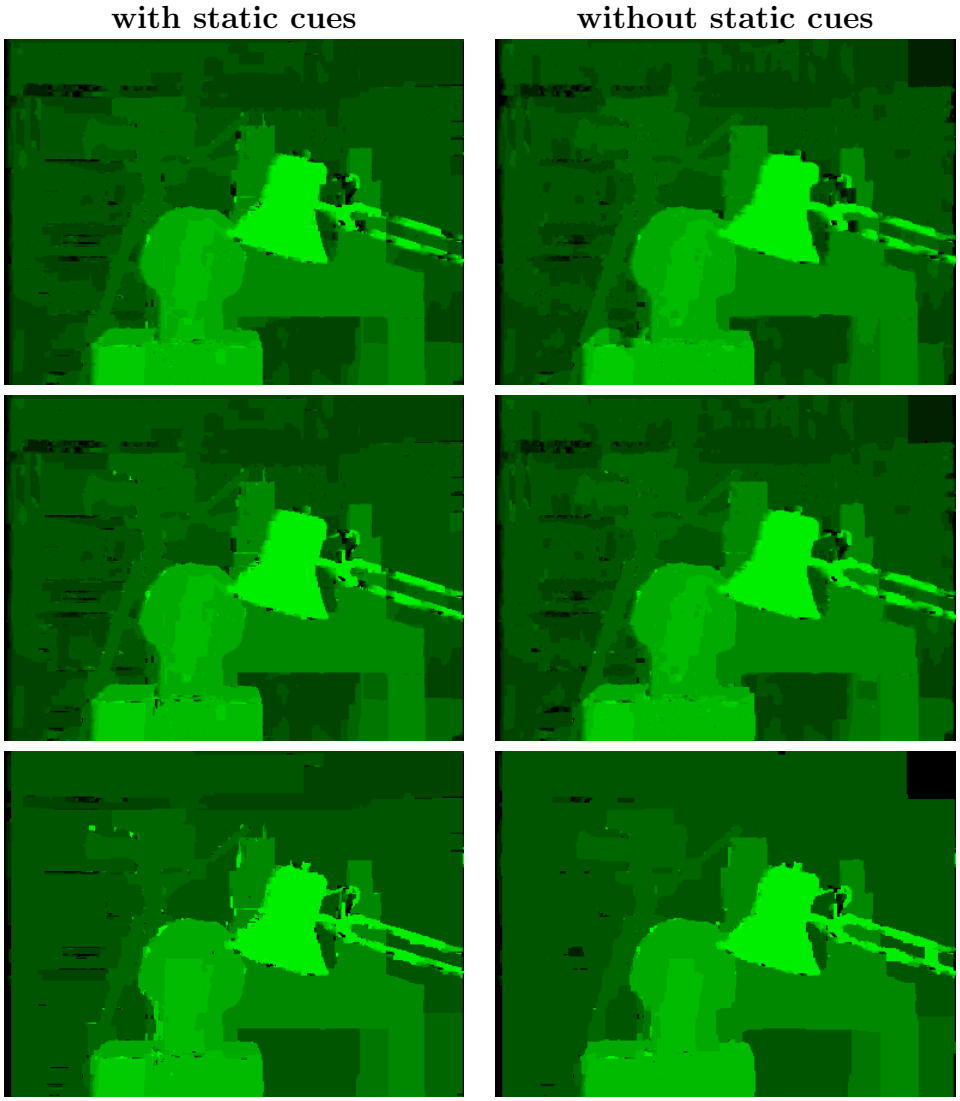


Figure 47: Advanced static cues results computed on the Tsukuba dataset [49] using the expansion move algorithm from section 6. **Top row:** $V^{\mathbf{quadr}}$. **Middle:** $V^{\mathbf{linear}}$. **Bottom:** $V^{\mathbf{Potts}}$. Quantitative error statistics are given in table 9.

table 10.

Image pair	$BP^{\text{non-occ}}$ 4 neighbours	$BP^{\text{non-occ}}$ 8 neighbours
Tsukuba	2.48	2.171
Venus	2.155	1.872
Cones	11.114	6.499

Table 10: Comparison of statistical error measures for the extended neighbourhood set. Expansion moves and Potts model used. The truncation level has to be adopted when using 8 neighbours, because otherwise the smoothness term attains too high weight. Images taken from [49, 48].

8.3 Semi-symmetric Occlusion Handling

Occlusions are a crucial problem for stereo methods, since on the one hand they occur naturally and on the other hand the usually imposed colour consistency term is not resembling the physical reality of our scene. Thus by enforcing our data term to be contributing to our model globally, we introduce a systematic error. This is an obvious drawback, because we minimise an energy whose global minimum does not naturally coincide with the true disparity field.

In section 7, we tried to tackle this problem by allowing the solution to not establish a correspondence in some pixels and by that we excluded systematically incorrect contributions of the data term. The downside of this realisation was a complex and inflexible graph construction, as well as a not 100% dense disparity field.

On the other hand the method we presented in section 6 had a comparably straightforward and flexible construction, but completely disregarded occlusions as well as treated the images asymmetrically.

Inspired by a variational method by Ben-Ari and Sochen [3], we now want to combine the advantages of both methods, i.e. a straight forward and simple construction, with the symmetric treatment of both input images and the incorporation of occlusions. Unfortunately we found out in the meantime that Chang et. al. presented a similar method in 2006 [18].

Consistency checking Instead of the fully symmetric and equitable handling of both images and the enforced uniqueness constraint for pixel assignments, we will use another strategy to detect occlusions. This technique is known under several names as *cross-checking*, *consistency check* or *forward-backward check* [20, 30, 8].

The idea is simple: We compute both, left-to-right and right-to-left

stereo disparity fields d_l and d_r, respectively. This computation is carried out using the classical approach from section 6. Since the computation of d_l and d_r has basically been done on the same images, we can assume that the two fields should be *consistent*. But if this is not the case somewhere, then we will assume, that this pixel is occluded, because no consistent corresponding pixel could be determined in the other frame.

Formally speaking, we compute two disparity fields

$$d_l : \Omega_l \to \mathcal{L} \quad \text{and} \quad d_r : \Omega_l \to \mathcal{L}.$$

As introduced in section 5, these disparity fields establish a correspondence between the two frames. Explicitly we say that for $i \in \{l, r\}$ a point $\mathbf{x}$ in one frame corresponds to the point $\mathbf{x}'$ in the other frame and can be computed via

$$\forall \mathbf{x} = (x, y) \in \Omega_i : \qquad \mathbf{x}' = (x + d_i(\mathbf{x}), y)\,.$$

Since we are equipped with both disparities we can now perform a consistency check as follows: For each pixel at position $\mathbf{x} = (x, y) \in \Omega_l$ in the left frame, we compute its corresponding pixel in the right frame according to the last equation $\mathbf{x}' = (x + d_l(\mathbf{x}), y) \in \Omega_r$. Since we are in the ortho-parallel setting, no vertical disparities have to be considered. We can now look up the right-to-left disparity in the right frame, which should lead us into the left frame to position

$$\mathbf{x}'' = (x + d_l(\mathbf{x}) + d_r(\mathbf{x}'), y) \in \Omega_l$$

again. The notion of consistency can now be defined as the case where

$$\mathbf{x} = \mathbf{x}''.$$

This process is illustrated in figure 48 again. Obviously, it is equivalent to test whether the values of the disparities of corresponding pixels fit:

$$d_l(\mathbf{x}) + d_r(\mathbf{x} + d_l(\mathbf{x})) \stackrel{!}{=} 0 \tag{8.3}$$

This procure can be defined analogously for the right frame. Note that it would not make sense to use the advanced approach from section 7 for the disparity computation here, because the left-to-right and right-to-left field being produced by this method must fulfil this consistency assumption already by the underlying model of the method.

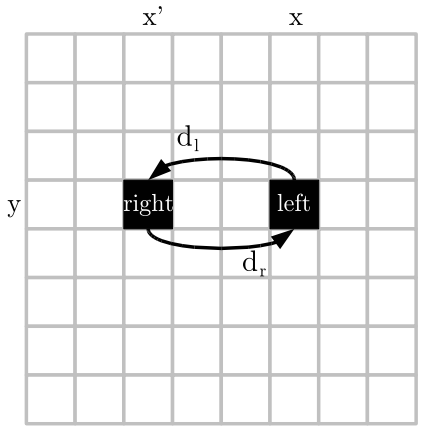


Figure 48: Consistency checking. Starting from one frame, we try to go to the other frame and from there back to the first frame again.

Mistrusting inconsistent pixels We cannot be sure to have an occlusion whenever the consistency check detects irregularities, but at least we know that we are confronted with a *problematic* pixel. But in case that we really have an occlusion in this pixel, we know definitely that the colour consistency assumption that our data term imposes does not reflect the reality. Thus, we will *switch off the data term* whenever the consistency check fails.

To this end we compute two consistency fields $C_l : \Omega_l \to [0,1]$ and $C_r : \Omega_r \to [0,1]$ as

$$C_l(\mathbf{x}) = g\left((d_l(\mathbf{x}) + d_r(\mathbf{x} + d_l(\mathbf{x}))\right) \tag{8.4}$$

where the function g is again the function from (8.1) due to Perona and Malik [44]. For our tests we took the value $\lambda = 1$ for the parameter of this function.

The right consistency field can be computed analogously to (8.4). We can then realise the disabling of the data term by multiplying it with the corresponding consistency field. Hence we will reuse the graph construction from section 6, with the new data terms $\tilde{D}_p^l(f_p)$ and $\tilde{D}_p^r(f_p)$

$$\tilde{D}_p^i(f_p) = C_i(p) \cdot D_p(f_p) \qquad \text{for } i \in \{l, r\} \tag{8.5}$$

where D_p is the usual data term for stereo vision from equation (3.18). Chang et al. [18] did not use a Perona Malik type function that only partly

deactivate their data term, they completely disabled the data term as soon as there is any inconsistency detected during the cross-checking.

We may not forget to enable the image driven part of the smoothness prior if we disable partially the data term. This is very important, because by that the discontinuities in the disparity field will be aligned with discontinuities in the image.

The general proceeding is obvious now: We will repeatedly compute both left-to-right and right-to-left disparity fields with the new data term from (8.5). After each iteration we will update the consistency fields with the fresh disparity information. As output, we give the disparity field as computed. It would be possible as well to hand out the occlusion map, because we compute them anyway after each iteration.

As initialisation for C_l and C_r we just use fields that have value 1 everywhere, i.e. the standard stereo computation as introduced in section 6. We let our algorithm perform 4 iterations. The disparity computations in each iteration can easily be performed in parallel on machines with more than one processing unit, because the computations do not need to exchange any information that might get changed at runtime. By that the computation time just increases by the number of iterations.

Figure 49 and table 11 present visual and quantitative results of this simple algorithm. Figure 50 illustrates the first few iterations of our algorithm on the Tsukuba sequence. One can see that the occlusion maps improve in the first few iterations, and worsen bit then. This is a behaviour one still has to investigate on, maybe a relaxation of the binary decision for occlusions might help here. Usually we would expect a convergence of the resulting disparities and occlusion maps after some iterations.

Concluding this method, we can state that we have used simplest means to improve the performance of the classical graph cut method by Boykov et al. significantly. This might be interesting, because the intuitive graph construction can be used to get really good results. Unfortunately we have not been the first to combine these two concepts, [18] proposed a very similar idea already at the Asian Conference on Computer Vision in 2006.

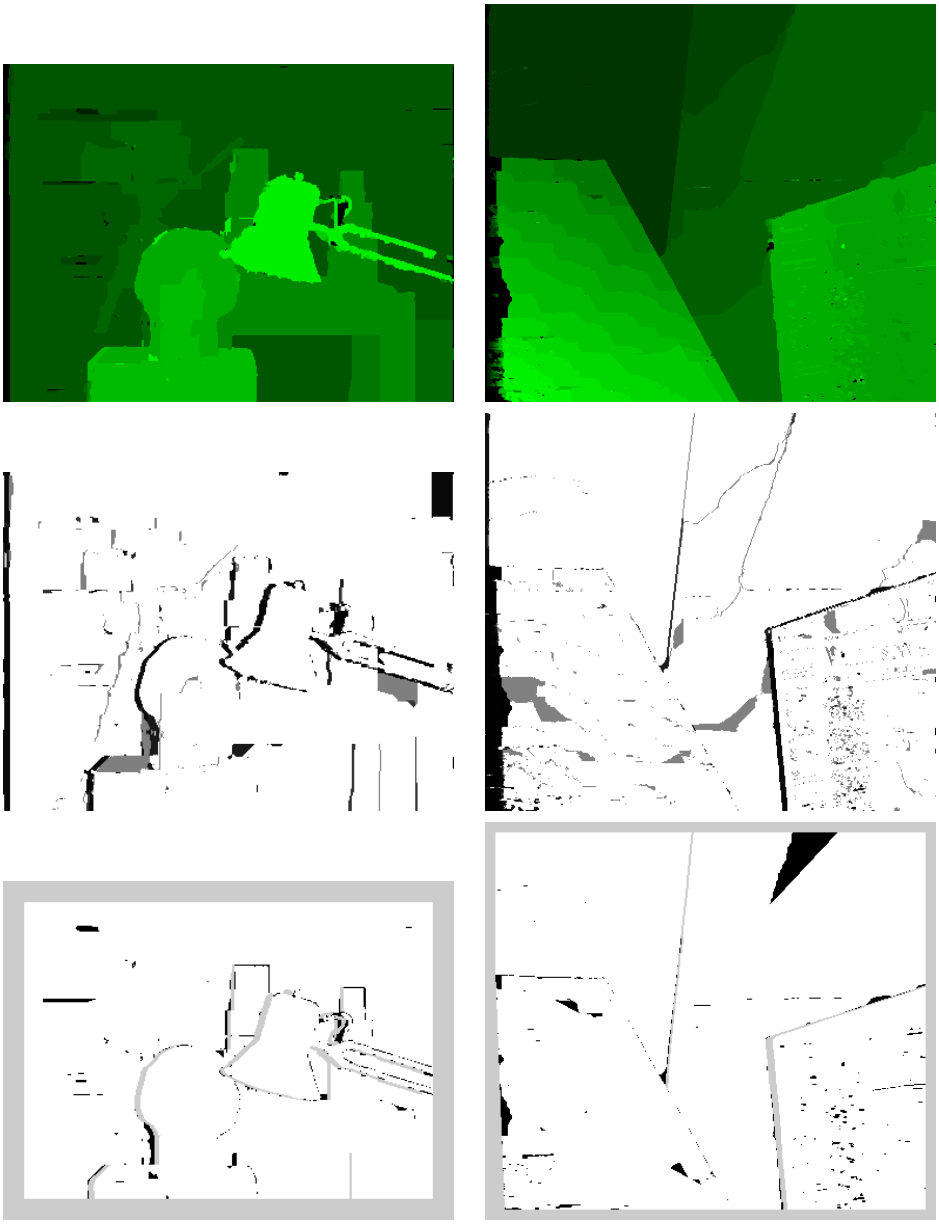


Figure 49: Computed disparities (top row), consistency fields (middle) and error images (bottom) for the Tsukuba and the Venus image sequence [49, 48]. For the consistency fields, dark colours indicate inconsistent areas. The error image is shaded in grey in all areas where the ground truth occlusion map is marked as occluded, these areas do not count for the $BP^{\textbf{strict non-occ}}$ and $BP^{\textbf{relaxed non-occ}}$ error measures.

Sequence	our approach	Boykov et al. [13]	Kolmogorov [38]
Tsukuba	1.66	2.171	1.057
Teddy	8.839	9.21	5.948
Cones	5.533	8.525	11.725
Venus	1.805	1.872	0.956

Table 11: Comparison of quantitative error statistics results between the discussed algorithms and our proposed algorithm. Numbers given present the $BP^{\text{relaxed non-occ}}$ measure. Note that the third column gives the error statistics of the base method by [13], but with all our extensions already enabled.

8.4 Other extensions

Besides the just presented extensions, we also tried to adapt several successful methods and ideas from the field of variational methods to graph cuts. Among those, e.g. we tried to impose a gradient constancy assumption [56, 14] additionally to the colour or grey value consistency assumption. This idea assumes that two pixels correspond, if their derivative of the image intensities is similar. The advantage of this assumption is that it is invariant under additive illumination changes, which can be very helpful in practice.

There are several ways how to implement and realise such an additional gradient constancy term. Especially the question at which point to introduce the robust penalisation is interesting. Either, one can sum together the gradient and grey value constancy terms where each term itself is robustified [15], e.g. in a TV fashion, or one can sum the squared terms together and apply the robust penaliser to the result [14]. In contrast to their variational counterparts, we found out that for graph cuts the latter proceeding according to the data term in the method of Brox et al. [14] leads to better results. But still we have to note that the quality of the computed disparities does not increase by using a gradient constancy assumption – at least not for the common test sequences that we used [49, 48, 47]. One resulting disparity obtained with the combined robustification [14] is given in figure 51.

The reason why generally the idea of incorporating derivative information unfortunately does not enhance the results of our graph cut method seems to be the averaging that necessarily happens when estimating the partial derivatives of the images. By this averaging, the previously precisely localised image information is blurred. This blur is unfortunately not

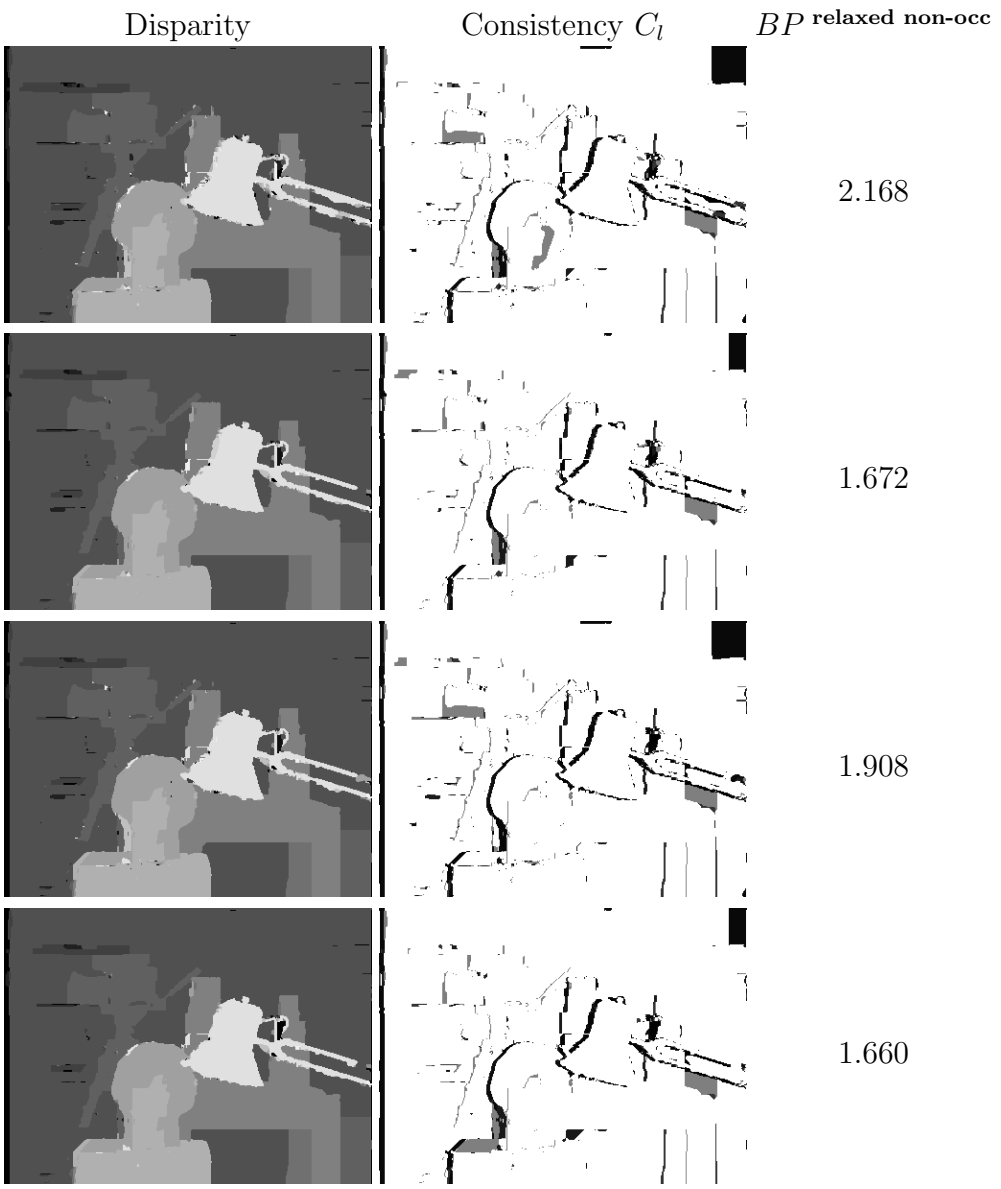


Figure 50: Behaviour of the algorithm over the first 4 iterations. **Top row to bottom:** Disparity and consistency fields after first, second, third and fourth iteration.

eliminated in the optimisation procedure and is still present in the resulting disparity field.

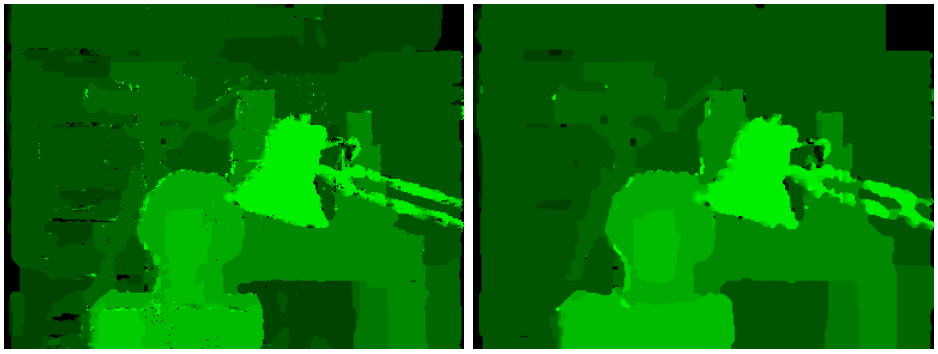


Figure 51: **Left:** Computation result using enabled gradient constancy assumption for $\beta = 5.4$. **Right:** Result for the CLG idea [17]. $BP^{\text{relaxed non-occ}} = 3.84$

Another unfortunately unsuccessful extension was the incorporation of block matching ideas, such as proposed by [17]. The underlying idea is to not only compare single pixels, but to compare patches of the images. In practice, the data term is evaluated on all pixels in a certain neighbourhood of the currently considered central pixel and their result is averaged. The big disadvantage of this idea in practice is that the assumption that the disparity is constant in this neighbourhood is not true if we consider a pixel near a depth discontinuity. Thus, such a CLG data term necessarily has to lead to blurring effects in the disparity. Like for the previously mentioned gradient constancy assumption, we found out that the order in which to sum up and penalise is very important. As before, a separately robustified version that sums up the already penalised parts of the convolution showed to give worse results as the opposite order. What we did was formally to use a data term

$$E_{\text{data}}(f) = \sum_{p \in \mathcal{P}} \Psi \left(h * D(f_p) \right) \tag{8.6}$$

In this formulation, the convolution of the data term with the kernel h inside the robust penaliser function can be written as

$$\Psi \left(h * D(f_p) \right) = \Psi \left(\sum_{\mathbf{x} \in \Gamma(h)} h(\mathbf{x}) \cdot D_{p+\mathbf{x}}(f_p) \right) \tag{8.7}$$

where $\Gamma(h)$ denotes the set of relative pixel positions where h does not vanish. Note that the data term is evaluated with the same constant disparity

value in all shifted positions. It is important here that the term $D_p(f_p)$ in (8.7) is e.g. the squared grey value difference. The penalising function was chosen as $\Psi(x) = \sqrt{x}$ which lead to an L_1 penalisation finally. For the convolution kernel h we used in practice binomial kernels as e.g.

$$h = \frac{1}{256} \cdot \begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$

where the centre position denotes pixel position $(0, 0)$. Increasing their size lead in the same way to comparably growing blurring effects. But as already mentioned in the original publication [17], this methodology should render the estimation of the disparity robust against noise in the input imagery.

9 Conclusion

In this thesis we have explored the world of graph cuts, a powerful method for discrete optimisation. Starting from different optimisation problems, we implemented and evaluated graph cut based optimisation processes.

After presenting graph theoretical fundamentals in section 2, we dived into the field of graph cuts in section 3 for the first time. There we tried to compare the computed results with those of related variational methods.

In section 4 we have discussed important properties of the penaliser functions. In particular, we reasoned why subadditivity is of such importance for the shape of discontinuities in the solution. Fortunately, we considered graph constructions being especially well suited for subadditive functions. A very important instance of such functions was the Potts model, which lead to piecewise constant solutions with perfectly sharp discontinuities.

Sections 5, 6 and 7 finally brought us to the main topic this thesis: Stereo Vision. We saw two successful methods, the first model only incorporated basic and straight forward model assumptions. The next method presented in section 7 was more advanced and made use of a fully symmetric treatment of both frames. This is also reflected in the quantitative error statistics: While the basic approach by Boykov et al. lead to moderate results, we saw that the symmetric method by Kolmogorov and Zabih really produced disparity fields of remarkable quality.

In the final part of this thesis we introduced several smaller contributions known from other related methods such as image driven smoothness terms also for other than the Potts models, gradient constancy assumptions and robust penalisation for the data term. We also extended the neighbourhood system, which lead to slightly improved results. Finally we adopted an occlusion treatment technique inspired from a variational stereo method [3, 18], which also lead us to improved results.

It seems that the main advantages of graph cut based optimisation are the extended move spaces, that can find local minima of the cost function by changing *multiple* pixels at a time. The expansion moves even considered all pixels at a time and found a labelling that had the least energy among all possible labellings being reachable by one such expansion move. This is a very strong capability compared to standard moves, e.g. like simulated annealing which can only change one pixel in one move. Moreover, we saw that the optimisation process, although it is not initialisation independent in theory, converged to almost the same solutions for any initialisations. The same was true for the order in which the labels were processed.

An obvious drawback of graph cuts is their fully discrete nature. While this has a strongly regularising effect in practice, it still constricts their usage considerably. E.g. the inherent sub-pixel accuracy of methods using partial differential equations can only be partly overcome by increasing the number of labels drastically. This in turn blows up the computational effort to a point where other methods might become equally suitable again. Thus we have to trade off between the problem size and quality requirements.

While considering all the proposed methods and their results, one should keep in mind that there is an important difference between the model and its optimisation. The perfect optimisation strategy will not help, if one tries to minimise the wrong model. In the same way, if our minimiser cannot find a good minimum of the perfect model assumption, then can still not get great results. Practically speaking it seems that a model for stereo vision that does not take occlusions explicitly into account seemingly cannot get beyond a certain quality threshold.

9.1 Further work

Besides the two main publications on stereo vision that we have discussed in detail in this thesis, there are a lot more highly interesting and promising publications on the topic of graph cut methods. The range of possible fields of application is wide spread, because we just need to have an appropriate discrete energy formulation at hand for a certain problem, in order be able to apply graph cut methods. There is a successful publication by Kolmogorov and Zabih [40] on exactly this topic of generalising and formalising the applicability of graph cut methods. Several general function classes are given there, with corresponding generic graph constructions and optimality arguments.

We also want to mention a paper by Kim et al. [36] trying to incorporate the maximisation of mutual information [50, 60] instead of the classical grey value constancy into the energy formulation for stereo vision.

Optical Flow Another big topic in computer vision, which we have not treated at all so far is optical flow estimation, which can be seen as a generalisation of the 1-D correspondence problem of ortho-parallel stereo vision to the 2-D case. There, our search for corresponding pixels is not limited to the actual scan line, motions may occur in all directions. As a consequence, our label set must become two-dimensional in order to express horizontal and vertical motions at a time. While this obviously increases the computa-

tional effort drastically, it definitely would be interesting to have such sharp discontinuities as the stereo algorithms have shown us. Moreover the fact that graph cuts do not have to work on multiple scales of the input images, which is the case for today's state of the art methods for optic flow, would be a great feature of graph cut based optic flow methods.

FusionFlow In the context of the just mentioned optic flow methods and besides a straightforward extension of already known and existing methods, e.g. by Boykov et al. [13], there is a very interesting publication by Lempitsky et al. [42] going in a different direction. Especially this work comes partly across the problem of a fully discretised two dimensional label space. Instead of trying to estimate the displacement vectors, this approach needs a number of candidate solutions, which may be computed with any optical flow method, e.g. variational methods like [35, 17, 14]. Also constant proposal solutions are used. The method then builds a graph for all pairs of two such proposals and performs a binary optimisation always selecting for each pixel from one of the two currently processed proposals. In this way, this method has shown to produce high quality flow fields, although the proposal solutions used by Lempitsky et al. were themselves of comparably low quality.

Incorporation of colour information In many cases, the incorporation of colour information has shown to give a respectable performance gain. Usually using modern imaging techniques for stereo vision produce colour images naturally, so this information is available in most cases. There are several ways how to realise the usage of vector valued image material. A straight forward idea would be sum up the data terms arising from each channel. Note that the smoothness prior would not be affected directly, we would just have to think about how to determine the local image contrast for colour images if we use the additional image driven term.

Adaptive Disparity Resolution Refinement We have seen that although graph cuts provide powerful means for the energy minimisation process in stereo vision, the inherent discretisation of the label space is a clear drawback. The arising disparity fields show stair casing artefacts, which are especially in slanted surface regions far away from the perfect solution. There are techniques to overcome this problem by fitting affine functions into the solution [5]. This means that after the disparity computation, the labels are replaced by affine function which have a continuous co-domain.

Another idea, staying completely in the discrete domain, could be to develop a method that adaptively detects consecutive labels of interest and refines the label set $\mathcal{L}$ between them. Simple α-β-swap moves could then be used to relabel the affected pixels. In this way the problem of stair casing cannot be solved, but at least the height of the steps could be adaptively reduced and disparity fields of higher quality could be a result.

Software optimisation It has become obvious, that while the software realisation used for this thesis was very handy and well understandable, there are inherently large performance drawbacks coming from the extensive use of pointers. One should note here, that due to its straight-forwardness, the chosen design was perfect for this kind of exploration of techniques in the field of graph cuts. But of course, the performance in reality could and should be improved drastically by other implementations. Moreover, there is no direct reason why it should not become possible some day to perform such computations in real time, or at least to compute several frames per second. The topology of the graphs we use is always similar and specialised algorithms like [10] can definitely speed up the calculations by orders of magnitudes.

Acknowledgements

At this point I would like to use the opportunity to express my gratitude to several people that supported me in writing this thesis.

First of all I want to thank Professor Joachim Weickert for offering me the possibility to write this master's thesis under his supervision in the Mathematical Image Analysis Group and for giving me the opportunity to work in this inspiring scientific environment. Also of great importance for this thesis was Dr. Andrés Bruhn, whom I would like to express my special gratitude to for advising my thesis, providing me with deep insights and great new ideas for this work and for long fruitful discussions on the topic. He also allowed me to use his visualisation frontend which simplified the testing process significantly. Also, I do not want to forget all other members of the MIA group for their kind support and interesting discussions, espcially Henning Zimmer for stimulating discussions on the topic of modern computer vision topics and state of the art ideas and techniques. Further I would like to express my gratitude to Markus Mainberger for allowing me to reuse his great illustrations on the stereo geometry and to Swen Grewenig for his unhesistant support in mathematical problems.

I also want to thank Vicky for always being at my side througout my studies.

Last but not least, I want to thank my parents for always being there when I needed them and for giving me every support they could. My family always meant a lot to me and I am glad to have such a good relationship to them.

Thank you!

Oliver Demetz
February 2009

References

[1] R. Acar and C. R. Vogel. Analysis of bounded variation penalty methods for ill–posed problems. *Inverse Problems*, 10:1217–1229, 1994.

[2] L. Alvarez, J. Esclarín, M. Lefébure, and J. Sánchez. A PDE model for computing the optical flow. In *Proc. XVI Congreso de Ecuaciones Diferenciales y Aplicaciones*, pages 1349–1356. Las Palmas de Gran Canaria, Spain, September 1999.

[3] Rami Ben-Ari and Nir A. Sochen. Variational stereo vision with sharp discontinuities and occlusion handling. In *Proc. Eleventh International Conference on Computer Vision*. IEEE, 2007.

[4] Stan Birchfield and Carlo Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406, April 1998.

[5] Stan Birchfield and Carlo Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *In International Conference on Computer Vision*, pages 489–495, 1999.

[6] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proc. 18th International Conference on Machine Learning*, pages 19–26. Morgan Kaufmann, San Francisco, CA, 2001.

[7] Avrim Blum, John Lafferty, Mugizi Robert Rwebangira, and Rajashekar Reddy. Semi-supervised learning using randomized mincuts. In *In Proceedings of the 21st International Conference on Machine Learning*, pages 97–104, 2004.

[8] R. C. Bolles and J. Woodfill. Spatiotemporal consistency checking of passive range data. In *Proc. 1993 International Symposium on Robotics Research*, Pittsburg, PA, 1993.

[9] Yuri Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *ICCV*, pages 105–112, 2001.

[10] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In

Energy Minimization Methods in Computer Vision and Pattern Recognition, pages 359–374, 2001.

[11] Yuri Boykov and Vladimir Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. *IEEE Ninth International Conference on Computer Vision*, 1:26–33, 2003.

[12] Yuri Boykov, Olga Veksler, and Ramin Zabih. Markov random fields with efficient approximations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–655, 1998.

[13] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[14] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optic flow estimation based on a theory for warping. In T. Pajdla and J. Matas, editors, *Computer Vision – ECCV 2004*, volume 3024 of *Lecture Notes in Computer Science*, pages 25–36. Springer, Berlin, 2004.

[15] A. Bruhn and J. Weickert. Towards ultimate motion estimation: Combining highest accuracy with real-time performance. In *Proc. Tenth International Conference on Computer Vision*, volume 1, pages 749–755, Beijing, China, June 2005. IEEE Computer Society Press.

[16] Andrés Bruhn. Correspondence problems in compuer vision. Website available at `http://www.mia.uni-saarland.de/Teaching/copcv08.shtml`, 2008. Lecture.

[17] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61:211–231, 2005.

[18] Ju Yong Chang, Kyoung Mu Lee, and Sang Uk Lee. Stereo matching using iterated graph cuts and mean shift filtering. In *7th Asian Conference on Computer Vision*, volume 3851/2006 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006.

[19] Boris V. Cherkassky and Andrew V. Goldberg. On implementing push-relabel method for the maximum flow problem. *Algorithmica*, 19:390–410, 1997.

[20] S.D. Cochran and G.G. Medioni. 3-D surface description from binocular stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):981–994, October 1992.

[21] I. Cohen. Nonlinear variational method for optical flow computation. In *Proc. Eighth Scandinavian Conference on Image Analysis*, volume 1, pages 523–530, Tromsø, Norway, May 1993.

[22] Scilab Consortium. Scilab – the open source platform for numerical computation. Website available at `http://www.scilab.org`.

[23] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. Mit Press, 2001.

[24] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiway cuts (extended abstract). In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 241–251, New York, NY, USA, 1992. ACM.

[25] Camil Demetrescu. What do we learn from experimental algorithmics. In *In Mathematical Foundations of Computer Science*, pages 36–51, 2000.

[26] E A Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Mathematics Doklady*, 11, 1970.

[27] P Elias, A Feinstein, and C E Shannon. Note on maximum flow through a network. In *IRE Transactions on Information Theory IT-2*, pages 117–199, 1956.

[28] Olivier Faugeras, Quang-Tuan Luong, and T. Papadopoulou. *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications*. MIT Press, Cambridge, MA, USA, 2001.

[29] L. R. Ford Jr and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, June 1956.

[30] P.V. Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *International Journal of Machine Vision and Applications*, 6(1):35–49, 1993.

[31] A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. In *Machine Vision and Applications*, volume 12, pages 16–22. Springer, 2000.

[32] A V Goldberg and R E Tarjan. A new approach to the maximum flow problem. In *STOC '86: Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 136–146, New York, NY, USA, 1986. ACM.

[33] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 35:921–940, 1988.

[34] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, April 2004.

[35] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[36] Junhwan Kim, Vladimir Kolmogorov, and Ramin Zabih. Visual correspondence using energy minimization and mutual information. In *International Conference on Computer Vision*, pages 1033–1040. IEEE Computer Society Press, 2003.

[37] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[38] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *In International Conference on Computer Vision*, pages 508–515, 2001.

[39] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *European Conference on Computer Vision*, pages 82–96, 2002.

[40] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 26, pages 147–159. IEEE, 2004.

[41] Vladimir Kolmogorov and Ramin Zabih. *Graph Cut Algorithms for Binocular Stereo with Occlusions*. 2005.

[42] Victor Lempitsky, Stefan Roth, and Carsten Rother. FusionFlow: Discrete-continuous optimization for optical flow estimation. In *IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, June 2008.

[43] E. Mémin and P. Pérez. Hierarchical estimation and segmentation of dense motion fields. *International Journal of Computer Vision*, 46(2):129–155, 2002.

[44] P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:629–639, 1990.

[45] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23:309–314, 2004.

[46] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[47] Daniel Scharstein and Richard Szeliski. Stereo vision research page.

[48] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, volume 1, pages 195–202. IEEE, June 2003.

[49] Daniel Scharstein, Richard Szeliski, and Ramin Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 2002.

[50] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, (27):379–423, 1948.

[51] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.

[52] S. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 349–356, 2005.

[53] N. Slesareva, A. Bruhn, and J. Weickert. Optic flow goes stereo: a variational approach for estimating discontinuity-preserving dense dispartiy maps. In W. Kropatsch, R. Sablatnig, and A. Hanbury, editors, *Pattern Recognition*, volume 3663 of *Lecture Notes in Computer Science*, pages 33–40. Springer, Berlin, 2005.

[54] Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *J. ACM*, 44(4):585–591, 1997.

[55] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill–Posed Problems*. Wiley, Washington, DC, 1977.

[56] S. Uras, F. Girosi, A. Verri, and V. Torre. A computational approach to motion perception. *Biological Cybernetics*, 60:79–87, 1988.

[57] Olga Veksler. *Efficient Graph-Based Energy Minimization Methods in Computer Vision*. PhD thesis, Graduate School of Cornell University, August 1999.

[58] Olga Veksler. Image segmentation by nested cuts. In *In IEEE Conference on Computer Vision and Pattern Recognition*, pages 339–344, 2000.

[59] Olga Veksler. Graph cut based optimization for mrfs with truncated convex priors. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR '07.*, 2007.

[60] Paul A. Viola and William M. Wells. Alignment by maximization of mutual information. In *International Journal of Computer Vision*, pages 16–23, 1995.

[61] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395 – 416, 2007.

[62] E T Whittaker. On a new method of graduation. *Proc. Edinburgh Math. Assoc*, 78:81–89, 1923.

[63] Wikipedia. Edmonds-karp algorithm — wikipedia, the free encyclopedia, 2008. [Online; accessed 11-December-2008].

[64] Herbert S. Wilf. *Algorithms and Complexity*. AK Peters, Ltd., 2 edition, December 2002.

[65] O. J. Woodford, P. H. S. T. Torr, I. D. Reid, and A. W. Fitzgibbon. Global stereo reconstruction under second order smoothness priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage*, 2008.

[66] Ramin Zabih, Olga Veksler, and Yuri Boykov. US Patent 6744923 - system and method for fast approximate energy minimization via graph cuts. United States Patent, June 2004.