TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

Dissertation

Segmentation-based Stereo and Motion with Occlusions

ausgeführt
zum Zwecke der Erlangung des akademischen Grades eines
Doktors der technischen Wissenschaften

unter der Leitung von
ao. Univ.-Prof. Mag. Dipl.-Ing. Dr. Margrit Gelautz
Institut 188 für Softwaretechnologie und Interaktive Systeme

eingereicht
an der Technischen Universität Wien
Fakultät für Informatik

von
Dipl.-Ing. Michael Bleyer
Matr.-Nr. 9525639
Gröhrmühlgasse 38
2700 Wiener Neustadt
bleyer@ims.tuwien.ac.at

Wien, im Februar 2006

Summary

Given two images recorded from slightly different perspectives, a shape-from-stereo approach identifies corresponding points in both images that are projections of the same point in the scene. In a standard stereo setup, such corresponding pixels are known to lie on the same horizontal scanline, so that this correspondence problem is reduced to a one-dimensional search task. The offset between x-coordinates in the left and right images is then referred to as disparity, and a pixel's disparity is inversely proportional to the pixel's depth. However, assigning each point to its correct disparity is a fundamental problem in computer vision. Although there is a large body of literature, common stereo methods still show poor performance in some image areas. Firstly, matching often fails in the absence of discriminative image features that can be uniquely matched in the other view (untextured regions). Secondly, some pixels' matching points are occluded in the second image. Since occlusions occur at disparity discontinuities, it is specifically challenging to precisely outline object boundaries. A large number of stereo algorithms fail in this respect, since the fact that there are occlusions is simply ignored.

In this thesis, we propose two novel stereo algorithms that tackle the inherent problems in stereo matching by dividing the reference image into segments of homogeneous colour. We assume that disparity inside such segments varies smoothly, while disparity discontinuities coincide with the segment borders. Both algorithms make use of a layered representation and model the stereo task as a two step problem. In the first (layer extraction) step, we answer the question: What are the dominant disparity planes (which we refer to as layers) likely to occur in the scene? These layers are extracted by clustering a set of initial disparity segments. In the second (layer assignment) step, we then focus on the more difficult question: Which part of the image is covered by which layer and where do occlusions occur? For the first algorithm presented in this thesis, we develop a novel global cost function that measures the goodness of an assignment of segments to layers by image warping. We show that image warping can as well be used to detect the occlusions in both images. This cost function is then optimized by a greedy algorithm, which is computationally efficient, but can get trapped in a local optimum. In order to overcome this weakness, we present a second algorithm for the layer assignment task that makes use of a recent robust optimization scheme, namely graph-cuts. The novelty of this approach lies in that we show how segmentation-based stereo matching can be formulated in a graph-cut approach with explicitly modelling occlusions. Both methods are then extended to the closely related optical flow (or motion) problem. As opposed to stereo, the displacement vector for this problem is a two-dimensional one.

In the experimental results, we demonstrate that our methods produce good-quality results, especially in regions of low texture and close to disparity/motion boundaries. Moreover, our stereo algorithms show excellent results on the Middlebury stereo evaluation website.

Kurzfassung

In einem Shape-from-Stereo-Ansatz werden zwei Bilder aus leicht unterschiedlichen Perspektiven aufgenommen. Die Aufgabe eines Stereoalgorithmus ist es dann, korrespondierende Pixel in beiden Bildern zu identifizieren, welche Projektionen des gleichen Punktes in der Szene darstellen. In einem Standard-Stereo-Setup ist es bekannt, dass korrespondierende Punkte auf derselben horizontalen Scanlinie zu finden sind, sodass das Korrespondenzproblem auf eine eindimensionale Suche beschränkt werden kann. Der Offset zwischen den X-Koordinaten im linken und dem rechten Bild wird dann als Disparität bezeichnet, und diese Disparität ist invers proportional zum Tiefenwert eines Punktes. Die Zuordnung eines Pixels zu seiner korrekten Disparität stellt jedoch ein fundamentales Problem des Maschinellen Sehens dar. Obwohl umfassende Literatur vorhanden ist, liefern herkömmliche Methoden schlechte Ergebnisse in gewissen Bildbereichen. So versagt der Korrespondenzfindungsprozess oft aufgrund des Nichtvorhandenseins von Bildmerkmalen, die zum eindeutigen Auffinden der gesuchten Pixelkorrespondenz im anderen Bild nötig wären (untexturierte Regionen). Darüber hinaus sind die Korrespondenzen mancher Pixel im anderen Bild verdeckt. Da Verdeckungen vor allem in der Nähe von Disparitätsunstetigkeiten auftreten, ist es besonders schwierig Objektgrenzen präzise zu rekonstruieren. Eine große Anzahl von Stereoalgorithmen scheitert in diesem Sinne, da sie die Tatsache, dass Verdeckungen vorhanden sind, einfach ignorieren.

In dieser Arbeit präsentieren wir zwei neue Stereoalgorithmen, welche die inhärenten Schwierigkeiten in der Stereokorrespondenzbildung mittels einer Zerteilung des Referenzbildes in Segmente gleichartiger Farbe bewältigen. Unsere Annahme ist, dass die Disparität innerhalb eines derartigen Segmentes kontinuierlich variiert, während Disparitätsunstetigkeiten mit den Segmentgrenzen zusammenfallen. Beide Algorithmen repräsentieren die Disparitäten anhand von Layern und modellieren das Stereoproblem in zwei Schritten. Im ersten (Layer Extraction) Schritt beantworten wir die Frage: Was sind jene ebenen Oberflächen (wir bezeichnen diese als Layer), die in der Szene dominant vorhanden sind? Diese Layer werden durch das Clustern anfänglicher Disparitätssegmente gefunden. Im zweiten (Layer Assignment) Schritt beschäftigen wir uns mit der schwierigeren Frage: Welche Teile des Bildes sollen welchem Layer zugeordnet werden und wo treten Verdeckungen auf? Für den ersten Algorithmus dieser Arbeit entwickeln wir eine neuartige Kostenfunktion, welche die Qualität einer Zuordnung von Segmenten zu Layern mittels einer Bildwarpingoperation misst. Wir zeigen, dass diese Warpingoperation auch dazu verwendet werden kann, um Verdeckungen in beiden Bildern zu erkennen. Ein gieriger Algorithmus wird dann zur Optimierung dieser Kostenfunktion verwendet. Dieser Optimierungsalgorithmus ist vom Rechenaufwand her effizient, läuft jedoch Gefahr, in einem lokalen Optimum stecken zu bleiben. Um diese Schwäche zu bewältigen, beschreiben wir einen zweiten Algorithmus zum Lösen des Layer Assignment-Problems, welcher von einer kürzlich publizierten, ro-

busten Optimierungstechnik Verwendung macht, nämlich Graph-Cuts. Die Neuheit unseres Ansatzes liegt darin, dass wir zeigen, wie segmentierungsbasiertes Stereo in einem Graph-Cut-Ansatz formuliert werden kann, sodass Verdeckungen explizit modelliert werden. Beide Methoden werden dann auf das nahe verwandte Problem der Berechnung des optischen Flusses (oder Bewegungsberechnung) erweitert. Im Unterschied zu Stereo ist der Verschiebungsvektor in diesem Problem ein zweidimensionaler.

In unseren Experimenten demonstrieren wir, dass unsere Methoden Resultate von guter Qualität generieren, vor allem in Regionen mit schwacher Textur und nahe an Disparitäts-/Bewegungsunstetigkeiten. Darüber hinaus erzielen die vorgestellten Stereoalgorithmen exzellente Ergebnisse auf der Middlebury-Stereoevaluierungswebseite.

Acknowledgements

First of all, I want to thank my supervisor Margrit Gelautz for her constant support and encouragement throughout the years that I spent as a PhD student. She brought me into the field of computer vision and taught me the principles of scientific work and writing. Without her guidance, this work would not have been possible. I want to express my gratitude to Christian Breiteneder for giving me the privilege to work in his group. He was always there to lend support (not only in financial concerns), which I do not see as a matter of course. He also proofread this thesis. I am grateful to Markus Vincze who kindly agreed to be the second reviewer of this thesis.

During my studies, I had the pleasure to supervise the master thesis of Christoph Rhemann. His work proved to be of enormous value for my research and contributed to the results shown in Chapters 6 and 8. I would further like to thank Danijela Markovic and Stathis Stavrakis with whom I share the office. Our discussions on work- and non-work-related topics were always a great source of inspiration. I would also like to acknowledge the other current and former members of IMS, especially Horst Eidenberger and Dieter Schmalstieg, who financed my research during the first half-year of my studies. Furthermore, I want to thank Daniel Scharstein and Richard Szeliski, who are running the Middlebury Stereo Vision Research Page, for including our results on their website.

I would like to thank all of my friends and people that supported me. (I do not put any names, since I am afraid to forget someone.) I would also like to thank my family, especially my mother and father, who made it possible for me to study. Finally, I thank Sissi Lackner for being at my side throughout all these years.

Financial support for this work was obtained from the Austrian Science Fund (FWF) under project P15663.

Contents

Chapter 1

Introduction

1.1 Stereo vision

The task of stereo vision is the computation of three-dimensional data from two-dimensional input images. This is exactly what the human visual system is doing when we perceive depth. Stereo vision therefore tries to imitate the ability of the human brain to infer depth from a scene and consequently uses the same principle. This principle is intuitively explained as follows.[1] Apart from interpreting monocular depth cues, i.e. cues that are present when only one eye is used, the human visual system relies on the fact that our eyes capture *two* images of the world. Since these images are obtained from slightly different perspectives, the position of a scene point in one view is horizontally displaced in the other view. The amount of this displacement allows reasoning about the point's depth, as we illustrate in Figure 1.1. The length of the horizontal displacement vector is commonly referred to as *disparity*, and a pixel's disparity is inversely proportional to the pixel's distance from the cameras. Using this principle, the human brain converts the disparity information into a three-dimensional impression of the world. Although the process seems to be very simple and the stereo task is permanently solved by the human visual system without us even noticing the effort, the same problem turns out to become very difficult when it needs to be solved by a computer. The major challenge that one faces when using a shape-from-stereo approach is that of solving the correspondence problem. The correspondence problem denotes the task of automatically computing the correct disparity value at each pixel. This is one of the oldest, but still most challenging problems in low-level computer vision and represents the topic of this thesis.

[1] We will give a more technical description at a later point in this thesis.

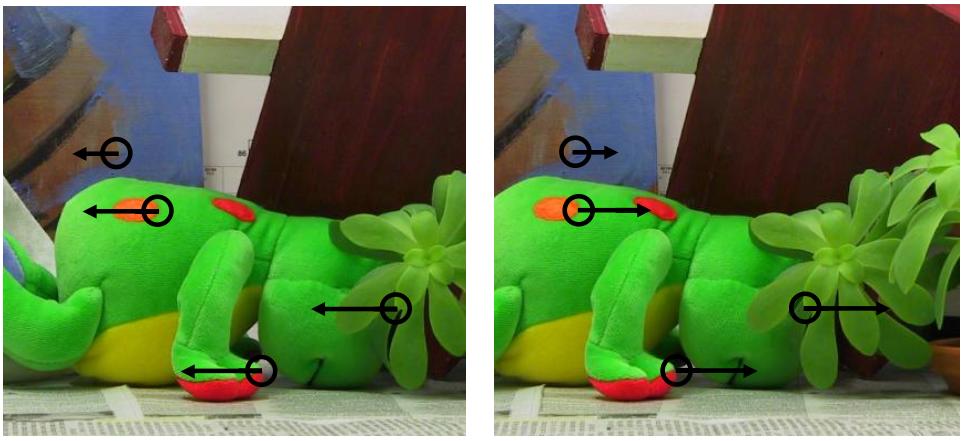


Figure 1.1: Binocular depth perception. Two images are obtained from different viewpoints. Due to the different perspectives, corresponding image points are displaced in horizontal direction as indicated by the arrows. The amount of displacement is inversely proportional to the depth of a point.

1.2 Contributions

The major goal of this thesis is to develop novel algorithms for the dense correspondence problem that are specifically designed to generate accurate results in image areas which are traditionally challenging in stereo matching. These image areas are regions of low texture and close to disparity discontinuities, where standard approaches often fail to compute the correct disparities due to the lack of discriminative image features, suboptimal smoothness assumptions or improper treatment of occlusions. In this work, the key step in overcoming these weaknesses of common methods is to divide the reference image into segments of homogeneous colour. Based on the assumption that disparity discontinuities go along with discontinuities in the intensity image, the resulting segments then form our algorithms' matching primitives. Therefore, instead of assigning each individual pixel to a discrete disparity value, we assign complete segments to continuous disparity models. The major advantages of this segmentation-based concept are twofold. First, disparities inside a segment are constrained to follow the same disparity model. This allows the assignment of smooth disparity values in regions of poor texture. Second, disparity discontinuities are enforced to coincide with segment borders. This is advantageous, since we believe that disparity boundaries can be more accurately identified by the use of monocular cues than this would be possible using disparity only.

In this thesis, we formulate segmentation-based stereo as a global cost optimization problem. We therefore develop two novel cost functions that measure the quality of an assignment of segments to disparity models. These cost functions are then subject to minimization. One advantage of our cost functions lies in that they explicitly model the occlusion problem in both views. Occlusions denote pixels which are visible in only one of the input images. These occlusions usually go along with discontinuities in depth, and if ignored, they corrupt the matching results in areas close to object boundaries. The capability of our algorithms to handle occlusions, together with the segmentation information, leads to an improved performance in regions close to disparity boundaries.

In our first approach, the cost function measures the quality of a disparity solution using an image warping mechanism. The basic idea is to transform the reference image into the geometry of the second view using the current disparity assignments. If the disparity solution was the correct one, the resulting warped image should show high similarity with the real second view. One novelty (among others) of our method is that we do not only apply image warping to measure the agreement of pixel values between the reference and the second views, but as well use the warping results to detect the occlusions in both images simultaneously. The cost function is then optimized by a greedy special purpose optimization scheme. We describe various ways to speed up the optimization process so that the resulting algorithm is computationally efficient.

The novelty of our second approach is that we show how to set up a cost function for segmentation-based stereo with treatment of occlusions that can be optimized via robust graph-cut-based optimization. Our idea is motivated by the observation that occlusions cannot be dealt with when modelling the stereo problem solely in the domain of segments. We therefore include two levels into our problem formulation, one representing the extracted segments and the other representing pixels. The link between these levels is then built by the following constraint: "Each pixel inside a segment must be assigned to the same disparity model as every other visible pixel of this particular segment. The pixel might, however, also be declared as occluded." Identification of occluded pixels is then modelled by enforcing the uniqueness of a match in both views. We prove that our cost function belongs to the class of those cost functions that can be optimized via graph-cuts and show how to construct the graph that is used in the optimization process.

The second problem that we address in this thesis is the optical flow or motion problem. A moving scene is thereby recorded by a single camera, and the task is to find a dense field of displacement vectors that transform one frame into a subsequent one. In contrast to the stereo correspondence

problem, such a displacement vector is, in general, a two-dimensional one. Since this results in a larger search range, optical flow computation can be regarded as a more difficult task than stereo. Nevertheless, due to the closely related natures of these problems, the challenges remain the same. In this thesis, we reformulate both of our algorithms in order to model the motion problem. Moreover, for the graph-cut-based approach, we show how this method is extended to compute the optical flow field considering more than two input images. We demonstrate that both algorithms can as well be applied to the task of motion segmentation.

1.3 Resulting publications

The work presented in this thesis has appeared in the following publications:

- M. Bleyer and M. Gelautz. A Layered Stereo Matching Algorithm Using Image Segmentation and Global Visibility Constraints. *ISPRS Journal of Photogrammetry and Remote Sensing*, 59(3):128–150, 2005.
- M. Bleyer and M. Gelautz. A Layered Stereo Algorithm Using Image Segmentation and Global Visibility Constraints. *IEEE International Conference on Image Processing (ICIP)*, Singapore, pages 2997–3000, 2004. (oral presentation)
- M. Bleyer, M. Gelautz and C. Rhemann. Colour Segmentation-based Computation of Dense Optical Flow with Application to Video Object Segmentation. *ÖGAI Journal (Special Issue on Multimedia Information Retrieval)*, 24(1):11–15, 2005.
- M. Bleyer, M. Gelautz and C. Rhemann. Region-based Optical Flow Estimation with Treatment of Occlusions. *Joint Hungarian-Austrian Conference on Image Processing and Pattern Recognition (HACIPPR)*, Veszprem, Hungary, pages 235–242, 2005. (oral presentation)
- M. Bleyer and M. Gelautz. Graph-based Surface Reconstruction from Stereo Pairs Using Image Segmentation. *SPIE Symposium on Electronic Imaging 2005 (Videometrics VIII)*, volume 5665, San Jose, CA, USA, pages 288–299, 2005. (oral presentation)
- C. Rhemann. Region-based Optical Flow Estimation with Treatment of Occlusions. Master's thesis, Vienna University of Technology, 2005.

Details on the camera system used in recording the test images as well as the application of the proposed algorithms to the task of automatic generation of artistic stereo paintings have been published in the following works:

- M. Bleyer and M. Gelautz. Video-based 3D Reconstruction of Moving Scenes Using Multiple Stationary Cameras. *27th Workshop of the Austrian Association for Pattern Recognition (Vision in a Dynamic World)*, Vienna, Austria, pages 181–187, 2003. (poster presentation)
- M. Gelautz, E. Stavrakis and M. Bleyer, Stereo-based Image and Video Analysis for Multimedia Applications. *XXth ISPRS Congress 2004*, Istanbul, Turkey, pages 998–1004, 2004. (oral presentation)
- E. Stavrakis, M. Bleyer, D. Markovic and M. Gelautz. Image-based Stereoscopic Stylization. *IEEE International Conference on Image Processing (ICIP)*, volume 3, Genova, Italy, pages 5–8, 2005. (oral presentation)

1.4 Organization

The remainder of this thesis is organized as follows. Chapter 2 describes the principles of depth reconstruction from stereo images from a technical point of view. We identify the matching task as the key step in stereo vision and point out the factors that make this problem challenging. Chapter 3 then reviews prior methods that compute dense correspondences from stereo and motion pairs. We describe the Middlebury stereo vision benchmark that is used in order to evaluate the performance of some of those algorithms on a set of standard images. The concept of segmentation-based matching, which builds the core for those algorithms presented in this thesis, is then introduced in Chapter 4. We discuss advantages and disadvantages that go along with a stereo method of this type. Chapter 5 presents the first novel algorithm for the stereo problem. This method shows moderate computational demands, but produces results of good quality. The algorithm uses a greedy search strategy to optimize a global cost function. We then modify this technique in order to compute the optical flow between two consecutive frames of a video sequence in Chapter 6. The second novel algorithm of this thesis is then described in Chapter 7. We show how to set up a cost function that models segmentation-based matching with explicitly accounting for occlusions. The cost function is designed to be optimizable via graph-cuts. This method is then extended to motion computation in Chapter 8. Finally, we give our conclusions in Chapter 9.

Chapter 2

Fundamentals of stereo vision

2.1 Stereo from a technical point of view

In the previous chapter, we have informally presented the concept of depth reconstruction from two-dimensional images in order to provide an intuitive understanding of "what is meant by disparity". In the following, we will focus on a more technical problem description.

2.1.1 Inferring depth from a two camera setup

Looking at the stereo problem from a technical point of view, we obtain the configuration illustrated in Figure 2.1. There are two pinhole cameras with C_l and C_r being their focal centers and L and R denoting the corresponding image planes[1]. We assume that the camera system is fully calibrated, i.e. the camera parameters as well as the positions and orientations of the cameras are known[2]. Both cameras capture the scene point P. The projections of P onto the left and right image planes p_l and p_r are given by the intersection of the lines $\overrightarrow{C_lP}$ and $\overrightarrow{C_rP}$ with the corresponding image planes. As a consequence of the projection, the z-coordinate of P is lost in each image and cannot be recovered when only a single camera is available. However, if we know the two corresponding points p_l and p_r, we can reconstruct P, which lies at the intersection of the rays $\overrightarrow{C_lp_l}$ and $\overrightarrow{C_rp_r}$. Unfortunately, these correspondences are not known apriori, and this leads to the difficult part of stereo reconstruction. The question that we have to answer is: Given the projection p_l of the scene point P onto the left image, where can we find

[1] We use capital letters to denote points in the world coordinate system (measured in meters) and small letters for points in the image (measured in pixels).

[2] Camera calibration can be achieved using standard methods [91, 105].

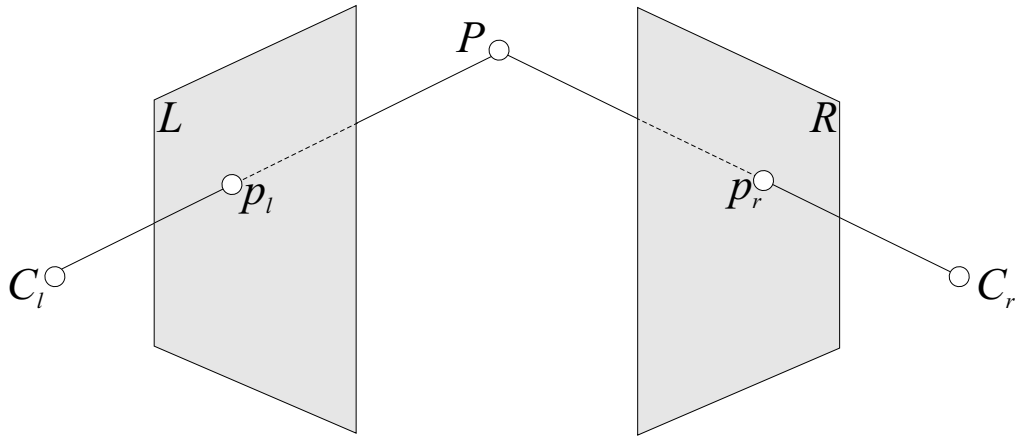


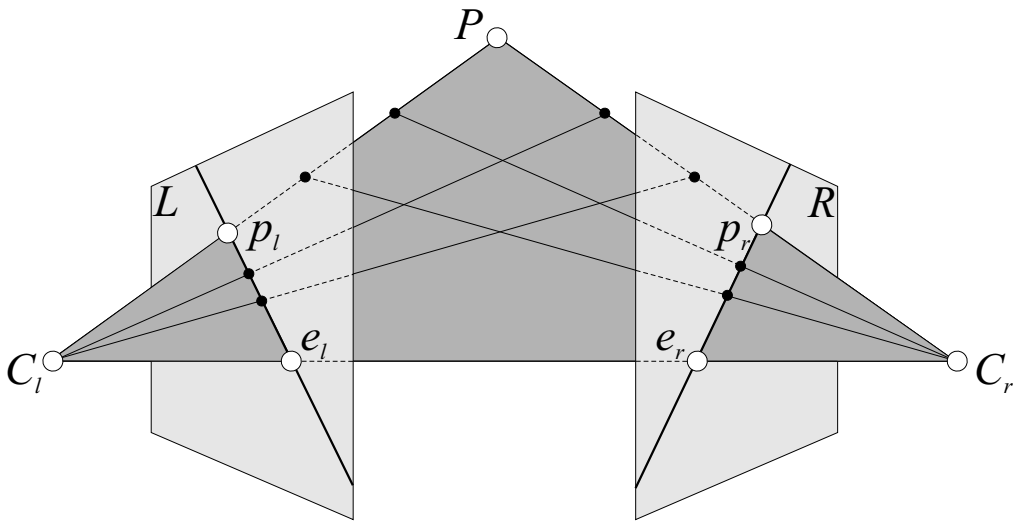
Figure 2.1: Two camera stereo setup.

p_r, which is the projection of the same scene point P onto the right image plane? This problem is known as the *correspondence problem* and represents the key step in stereo vision.

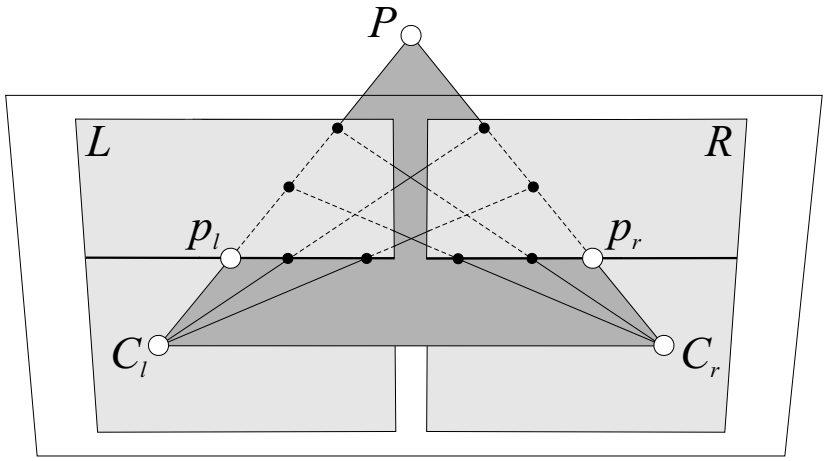
2.1.2 Epipolar geometry and rectification

To simplify the search for correspondences, we can use the observation illustrated in Figure 2.2a. Let us assume that we want to find the matching point of p_l in the right view. We can then observe that any scene point projecting to p_l is constrained to lie on a line, which is the projection ray $\overrightarrow{C_l p_l}$. Consequently, each of those scene points must also lie on a line in the right view. This line is given by the projection of the ray $\overrightarrow{C_l p_l}$ onto the second image. We refer to this line as *epipolar line* of p_l. It is interesting to note that each epipolar line in the right image must pass through the point e_r. This point is called *epipole* and lies at the intersection of the baseline, which is the line connecting the focal centers C_l and C_r, with the right image plane R. In other words, the epipole is the point where the left camera is seen from the right camera. Since the stereo problem is symmetrical, the same observations can be made when searching the matching point of p_r in the left view, as we show in the figure. Using the knowledge about epipolar lines, the correspondence problem can be reduced to a one-dimensional search task.

A specifically interesting case arises if both image planes L and P lie in a



(a)



(b)

Figure 2.2: Epipolar geometry. (a) Epipolar lines for cameras in general positions and orientations. The matching point in one view must lie on the corresponding epipolar line in the other view. (b) Epipolar lines after epipolar rectification. Image planes lie in the same plane and x-axes are parallel to the baseline. The matching point in one view lies on the same horizontal scanline in the other image.

common plane and their x-axes are parallel to the baseline, which is shown in Figure 2.2b. In this configuration, the epipoles move to infinity and the epipolar lines coincide with horizontal scanlines. The matching point of a pixel in one view can then be found on the same scanline in the other view, i.e. $y_l = y_r$ where y_l and y_r refer to the y-coordinates of a scene point in the left and right images, respectively. The horizontal offset between corresponding pixels $x_l - x_r$ is referred to as *disparity*. To take benefit of this simple geometry, the images of two cameras in general positions can be reprojected onto a plane that is parallel to the baseline. This process is known as *rectification* or *epipolar rectification*. The rectification step involves resampling of the image, and therefore some precision in the 3D-reconstruction is lost. However, since it is more convenient to search for correspondences along horizontal scanlines than to trace general epipolar lines, this transformation is commonly applied. Throughout this thesis, we will assume that this simple geometry is used whenever we deal with stereo pairs.

2.1.3 3D reconstruction via triangulation

Let us for now assume that the correspondence problem is solved. The reconstruction of a point's depth can then be accomplished via triangulation as shown in Figure 2.3. We assume that the cameras are epipolar rectified as described in Section 2.1.2. This means that the cameras are parallel to each other and have identical focal lengths. The focal length f, i.e. the distance between the camera's focal point and its optical center, and the baseline length B, i.e. the length of the line connecting the two focal points, are known from the calibration procedure. From similar triangles we derive that

$$\frac{X}{Z} = \frac{x_l}{f} \qquad \text{and} \qquad \frac{X - B}{Z} = \frac{x_r}{f} \tag{2.1}$$

and finally derive

$$Z = \frac{B \cdot f}{x_l - x_r} = \frac{B \cdot f}{d} \tag{2.2}$$

with B and f being constant values and d denoting the disparity according to our previous definition. From equation (2.2) we conclude that disparity is inversely proportional to depth. A *disparity map* that records the disparity for each image point is therefore sufficient for a complete three-dimensional reconstruction of the scene. This relationship is also the reason why disparity is commonly used synonymously with inverse depth.

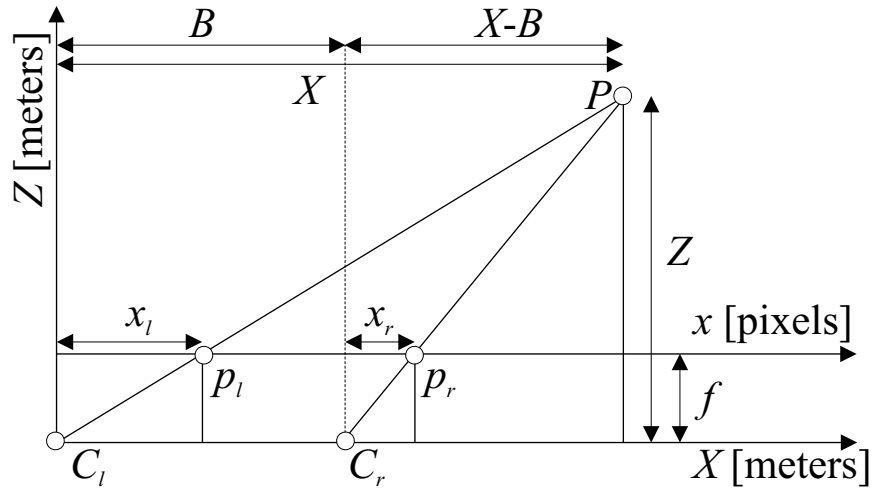


Figure 2.3: Reconstruction of the depth coordinate via triangulation.

2.2 The correspondence problem

In order to compute a three-dimensional scene reconstruction, we need to identify points in both images that are projections of the same scene point. What sounds easy in theory, turns out to be extremely difficult in practise. A good indicator for the problem complexity is the huge amount of papers that have been published on the correspondence problem ever since the early days of computer vision. Although there has been steady progress, the stereo problem still cannot be considered as being solved. We try to identify the factors that make this task so challenging in the following.

2.2.1 Challenges

It is quite reasonable to assume that pixels originating from the same scene point show constant intensity values in both images. This is referred to as the assumption of *photo consistency* and builds the core of almost every matching algorithm. Unfortunately, this assumption only holds true for *Lambertian surfaces*, i.e. perfectly diffuse surfaces that reflect the same luminance regardless of the viewing angle. In practise, the assumption of Lambertian surfaces is violated by *specular reflections*, whose positions and colours change substantially depending on the viewpoint. Furthermore, varying intensity values for the same scene point can as well be the consequence of *different sensor characteristics* of the two cameras. Image acquisition is also commonly af-

fected by a certain amount of *noise* originating from the camera electronics. However, photo consistency is probably the most effective assumption for solving the correspondence problem. Deviations from photo consistency are usually small enough to allow for the application of this constraint.

A second problem in stereo matching is identified as follows. Throughout the process of establishing correspondences, one has to rely on the presence of *discriminative image features* that can be *uniquely* matched in the other view. Unfortunately, such features do not necessarily exist in every part of the image. This is, for example, the case in *untextured regions*. Matching in an untextured area becomes ambiguous, since there are many potential matching points of very similar intensity patterns. When searching along horizontal epipolar lines, this situation obviously does not improve if there is continuous texture with only horizontal orientation. This phenomena is known as the *aperture problem*. Another source of ambiguity arises from the presence of a *repetitive pattern* (e.g. a wallpaper). Also, in this case, the image feature is found multiple times in the second view.

A third and final complicating factor is that a matching point might not even exist. The reason for this is that there are pixels that are only visible in one image. We refer to such pixels as *occluded*, or more correctly, *half-occluded*. The occlusion problem is illustrated in Figure 2.4. Since occlusions often occur at depth discontinuities, it is specifically challenging to precisely outline object boundaries, which is, nevertheless, often required for applications such as depth segmentation [61] or novel view generation [77, 109].

As becomes obvious from our problem analysis above, a strategy that matches single pixels by simple comparison of intensity values along epipolar lines would give very poor results. This indicates that the photo consistency assumption and the epipolar constraint alone are not sufficient to solve the problem. From a mathematical viewpoint, the stereo problem is known to be *ill-posed* [72] in the sense of Hadamard [38]. That is, (1) a solution may not exist (occlusions), (2) a solution may not be unique (untextured regions) and (3) small variations in the input data show large effects on the solution (noise). Consequently, additional assumptions are needed in order to make the problem tractable.

2.2.2 Assumptions

In the following, we review assumptions that have been commonly used in the literature to overcome the problem of ambiguity in stereo matching. The

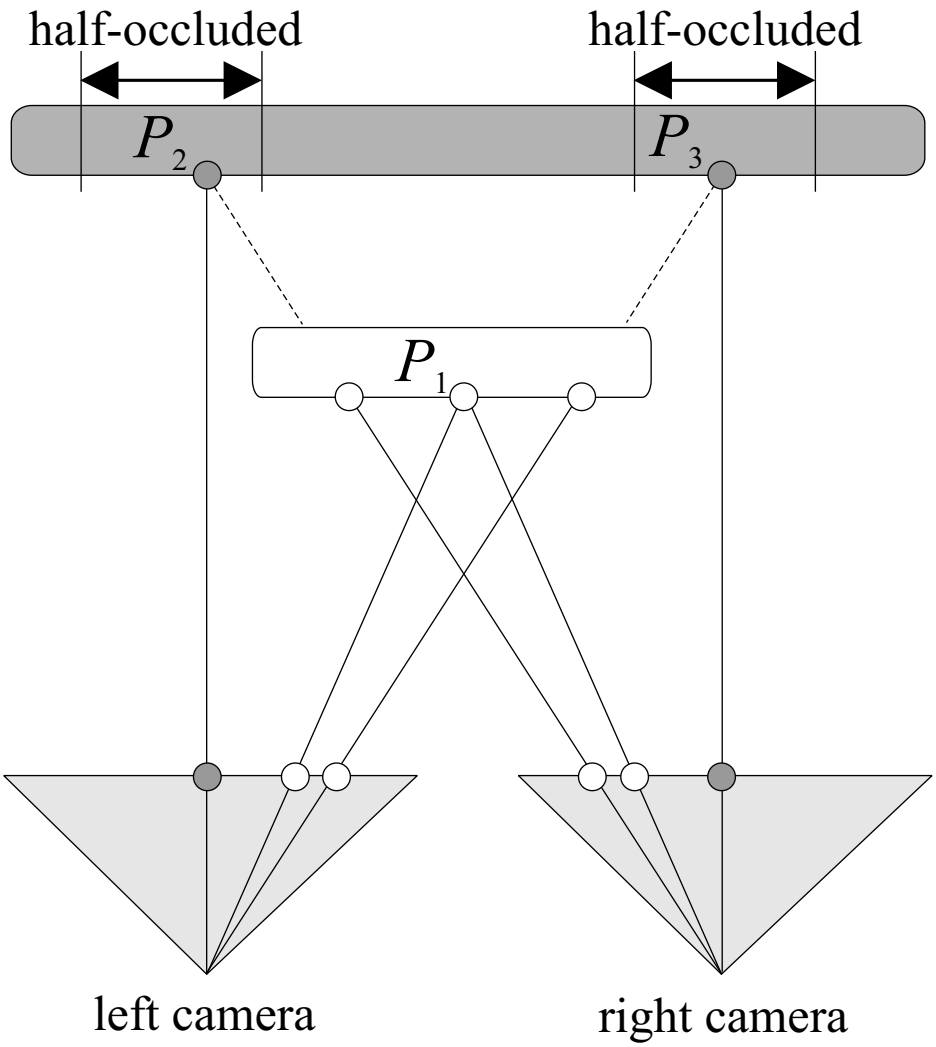


Figure 2.4: The occlusion problem. While it is possible to reconstruct P_1, which can be seen from both views, P_2 and P_3 are occluded in one view and can therefore not be matched. Oftentimes, occlusions are the consequence of discontinuities in depth.

epipolar constraint[3] and the photo consistency assumption are not listed, since they have been discussed earlier in this chapter.

Smoothness assumption

The smoothness assumption [62, 63] is motivated by the observation that natural scenes consist of objects with smooth surfaces. This assumption states that disparity varies smoothly almost everywhere (except at depth boundaries). That means we can expect that our disparity map is *piecewise smooth*. Smoothness is assumed by almost every correspondence algorithm, either in an implicit or explicit way. Whether this assumption holds true obviously depends on the scene content. While it is applicable on scenes consisting of compact objects (e.g. a flat box in front of a flat background), the assumption fails if there are thin fine-structured shapes (e.g. branches of a tree, hairs).

Uniqueness assumption

The uniqueness assumption [62, 63] states that a pixel of one view corresponds to at most one pixel of the other view. This constraint is often used to identify occlusions by enforcing one-to-one correspondences for visible pixels across images (e.g. [50, 108]). The uniqueness assumption is valid for scenes composed of opaque surfaces, but is broken if there are transparencies. An often used example is an image of a fish in a transparent fish bowl. In this example, a single pixel is the projection of both the fish and the bowl. Consequently, the same pixel has two different depth values. Another aspect that has been overlooked until recently is that for horizontally slanted surfaces the uniqueness assumption is violated as well [69]. Let us consider a slanted plane recorded with two cameras. The projections of such a plane will, due to the slant, show different widths in the two images. As a consequence of this different sampling, there are points in one view that correspond to more than actually one pixel of the other view.

Ordering assumption

The ordering assumption [103] states that the order in which points occur is preserved in both images. More precisely, let p_l be a point of the left view that corresponds to p_r of the right image. Moreover, another point q_l of the left image corresponds to q_r of the second view. The ordering assumption

[3] In contrast to the other assumptions, the epipolar constraint will indeed always hold true unless the calibration data is erroneous.

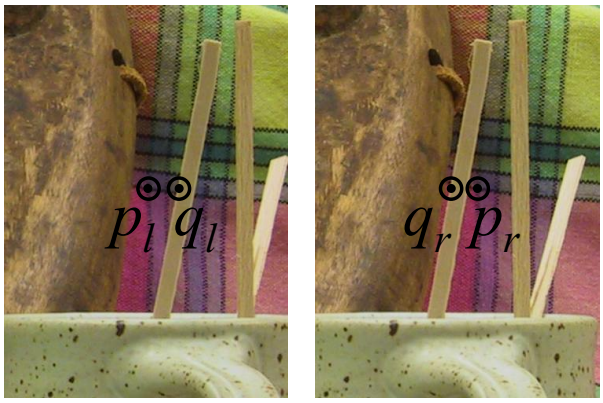


Figure 2.5: Violation of the ordering constraint at thin foreground objects. While p_l is left to q_l in the left view, p_r is right to q_r in the right image.

then states: If p_l is to the left of q_l then also p_r has to be to the left of q_r. The advantage of using the ordering assumption is that its application allows for the explicit detection of occlusions. This will become obvious in our discussion of dynamic programming-based stereo algorithms (for which the ordering constraint builds the basis) in Section 3.1.4. The limitation of the ordering assumption is that it indeed does not always hold true. It is known to fail for scenes containing thin foreground objects as we show in Figure 2.5.

Disparity gradient limit

The disparity gradient limit is interesting in the sense that it originates from psychological studies [21]. These studies have provided evidence that the human visual system fails to fuse two nearby dots of a stereogram if the points' disparities vary by a sufficiently large value. The disparity gradient is measured in the cyclopean view. This is the virtual view that lies exactly in the middle of the two real views. The disparity gradient of two points is then defined as the ratio of their disparity difference and their spatial separation in the cyclopean view. It has been shown in those studies that the human visual system fails in stereo fusion if the disparity gradient exceeds a value of one. For a stereo algorithm, this means that two points cannot be accepted simultaneously if their disparity gradient is above this limit. Since the disparity gradient limit arises from studies on the human visual system, it is not clear whether it can be used for general camera setups that are very different from the geometry of the human eyes.

2.3 Summary

In this chapter, the basics of stereo vision have been presented. Focusing on a technical problem description, we have introduced the epipolar geometry of two calibrated cameras and presented a transformation that aligns epipolar lines with the horizontal scanlines of the images. While the discussion of the matching problem has been postponed, we have established the relationship between a point's disparity and depth. The second part of this chapter has then been exclusively devoted to the stereo correspondence problem. We have pointed out the complicating factors in stereo matching, which are roughly summarized as: image noise, untextured regions and the occlusion problem. Since the epipolar constraint combined with the assumption of photo consistency have been found to be insufficient to solve the problem, we have finally reviewed the most frequently used assumptions in stereo matching.

Chapter 3

Solving the correspondence problem

3.1 Prior work on stereo

There is a considerable amount of literature on the stereo correspondence problem and giving an all-embracing review is hardly possible. We therefore focus our summary on a few techniques that we consider as important. An extensive review on recent stereo algorithms that produce a dense disparity field is given by Scharstein and Szeliski [78]. The authors identify four steps that are usually performed by a stereo method. These are: (1) matching cost computation, (2) cost aggregation, (3) optimization and (4) disparity refinement. The four steps build the basis for their taxonomy. In our analysis, we focus on the optimization component and divide algorithms between *local* and *global* methods with the major difference between these two approaches being the way how the smoothness assumption is employed.

Local methods assume that small image patches show similar intensity patterns across views. They typically operate on windows that are shifted on the corresponding scanline in the second view to find the point of maximum correspondence. Since pixels inside the search window are assumed to have constant disparities, those methods apply an *implicit* smoothness assumption. However, the final disparity assignment is obtained by selecting the point of highest matching score (*winner-takes-all principle*). This strategy is purely local in the sense that a pixel's matching score is not influenced by the disparity assignments of neighbouring pixels. In contrast to local approaches, global methods *explicitly* model the smoothness assumption. The correspondence problem is stated in terms of a cost function, which is subject to minimization. Although global approaches have shown to give the

strongest results at the current state-of-the-art [78], the optimization step usually involves a high computational effort. Real-time implementations using nowadays hardware are therefore mostly only possible for local methods[1].

3.1.1 Window-based correlation

Window-based (or local) approaches are popular in stereo vision for their simplicity and high efficiency. Since matching single pixels based on the winner-takes-all principle is very prone to matching ambiguities, those methods choose small image areas as their matching primitives instead. For this reason, those approaches are often also referred to as *area-based* methods. Window-based correlation exploits the concept of a support region. That is, each pixel receives support from its surrounding points. It is assumed that points inside this support region are likely to have the same disparity and can therefore help to resolve matching ambiguities. Usually, rectangular or square windows centered on the point for which a correspondence needs to be established implement this concept.

The first step of a window-based approach is the computation of matching scores, which represent the likelihood that two points correspond to each other. Using a predefined search range, the search window is therefore horizontally displaced in the second view for each allowed disparity. More precisely, the matching score for a pixel (x, y) at disparity d is derived by comparing the intensity values of the window centered at (x, y) of the first view against the window centered at the position $(x + d, y)$ of the second image. Commonly used matching measures include sum-of-absolute-differences (SAD), sum-of-squared-differences (SSD), normalized cross-correlation (NCC), as well as the sampling insensitive measurement of Birchfield and Tomasi [9]. The three-dimensional structure that is obtained by determining the matching score for each pixel (x, y) at each disparity d is commonly referred to as *disparity space image* (DSI) [15, 78]. In a naive implementation, the efficiency of calculating the DSI depends on the chosen window size, which is computationally expensive. However, for rectangular search windows of fixed size, a "sliding window" technique, as described by Faugeras et al. [29] and Mühlmann et al. [67], can be employed to make the complexity linear with respect to the number of pixels and disparities. This gives rise to real-time implementations such as the two most popular commercial stereo systems of SRI [53] and Point Grey Research [2].

Once the matching costs are determined, the matching point is derived

[1] An exception to this are some global techniques that use dynamic programming for the optimization of the objective function.

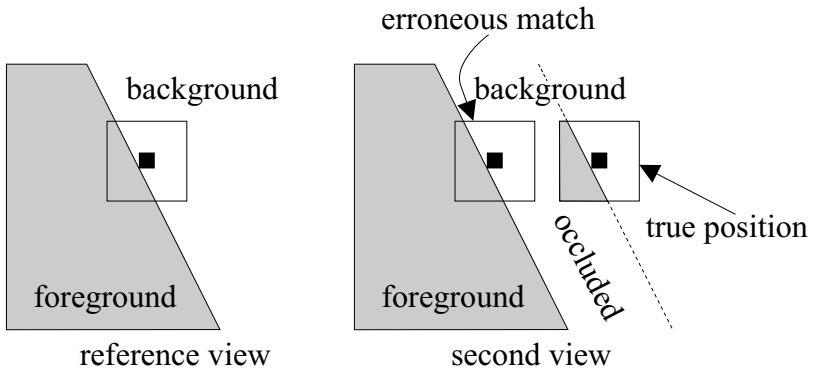


Figure 3.1: Systematic error of window-based methods near depth discontinuities.

based on the winner-takes-all principle. That is, the point of highest matching score is selected from the DSI[2]. Unfortunately, it is not guaranteed that the matching costs exhibit a clear and sharp extremum at the correct disparity. This is the case in untextured regions or in the presence of a repetitive pattern. Multiple disparities then show very similar matching values and image noise will strongly influence the selection of the final disparity. To avoid this problem, the search window has to be large enough to capture sufficient intensity variation.

A problem that arises when using image patches as matching primitives is that the implicit assumption of constant disparity within a search window is not guaranteed to hold true. Looking at this assumption in more detail, it will indeed only hold true if the scene is entirely composed of planes fronto-parallel to the camera. Using this strategy for matching slanted surfaces leads to undesired effects due to perspective distortions. Even more severe, the assumption of constant disparity is systematically broken at depth discontinuities where the search window captures pixels of completely different disparities. Matching in those regions becomes ambiguous. In Figure 3.1, we try to match a background point whose search window partially overlaps a depth discontinuity. The correct solution would be to match the window at the disparity of the background. However, the intensity patterns at this position are quite different, whereas the similar patterns at the disparity of the foreground indicate a good match. In general, the decision which disparity is chosen at a depth discontinuity depends on the texture of the fore- and background as well as on the intensity pattern of the occluded region. Since

[2]Window-based cost aggregation can as well be combined with different optimisation strategies. This is preferable over point-wise matching if there is a high noise level.

in natural images, the background often continues similarly (as is the case in Figure 3.1) the point is matched at the disparity of the foreground. The resulting elongation of foreground objects in the disparity map is commonly referred to as the *foreground fattening effect*. Problems associated with the implicit smoothness assumption obviously become more severe with larger window sizes, since the chances of capturing pixels of different disparities are increased.

Generally, the choice of an appropriate window size is a crucial decision. Small windows do not capture enough intensity variation to give correct results in less-textured regions, whereas large windows tend to blur the depth boundaries and do not capture well small details and thin objects. This motivates the use of *adaptive windows* that vary the shapes and sizes of support windows [15, 33, 39, 46, 47, 88, 93, 102]. Kanade and Okutomi [46] iteratively change the size and position of a rectangular window based on the local variation of the intensity and current disparity estimates. Their method requires an initial disparity map to start from. Fusiello et al. [33] anchor nine equally sized windows at different positions. They assume that at least at one of those positions the window does not overlap a depth discontinuity. Consequently, the matching score is chosen to be the minimum of the matching costs of all nine windows. Using a similar idea, Hirschmüller et al. [39] divide the search window into a set of subwindows. The subset of those windows that show the highest correlation values is used for the matching cost computation. Finally, Veksler [93] estimates an arbitrarily sized and shaped window at each pixel by optimizing over a large class of compact windows.

However, in practise, even adaptive windows only partially represent a remedy to the above illustrated problems. *Feature-based methods* therefore estimate correspondences only for points that can be matched unambiguously (e.g. [60, 76, 92]). This results in a *sparse* disparity map (as opposed to a *dense* one that gives a correspondence at each point). A commonly applied method to filter out unreliable correspondences in window-based correlation is to enforce the uniqueness of matches across views. Let us assume that the point p_r of the right view was computed to be the matching point of a pixel p_l. This match is then valid only if p_l also gives the highest matching score when searching for the matching point of p_r in the left view. In other words, the matched point in the right image has to point back to the pixel in the left view. This check is often referred to as *cross-validation* or *left-right consistency checking*. As Fua [32] points out, cross-validation is known to fail if the areas to be correlated have little texture or in the presence of an occlusion.

3.1.2 Progressive approaches

Progressive methods [55, 86, 97, 106] aim at resolving the problem of ambiguity in stereo matching by first establishing correspondences between points that can be matched unambiguously. These "reliable" points are then used to disambiguate subsequent matches by imposing constraints on them. This strategy originates from the *least commitment principle* used in Artificial Intelligence [106]. The principle states that unreliable decisions should be postponed until enough contextual information is gathered. However, although using disparity information from surrounding pixels, progressive approaches still belong to the class of local methods, since they do not explicitly optimize a cost function.

Zhang and Shan [106] measure the reliability of a point based on its local variation of intensity. Moreover, they require that for a reliable point the matching scores show a strong unique peak at a particular disparity. To resolve ambiguity for subsequent matches, Zhang and Shan then use the disparity gradient limit. Szeliski and Scharstein [86] present a progressive method that applies cross validation to filter out unreliable points. Once a set of reliable points is known, they rule out potential subsequent matches using the uniqueness constraint. Furthermore, the search window size is iteratively increased to reduce matching ambiguity.

The major advantage of progressive approaches lies in their computational efficiency, since they avoid computational expensive global optimization. On the other hand, they cannot recover from early wrong decisions, which corrupt subsequent matches. Recently, Wei and Quan [97] proposed to overcome this problem by matching segments that are derived from colour segmentation. They argue that since regions contain richer information than individual pixels, the likelihood of early wrong matches is reduced.

3.1.3 Cooperative approaches

Cooperative approaches [62, 63, 64, 104, 108] explicitly model the assumptions of smoothness and uniqueness that *cooperatively* solve the stereo problem. Those methods operate directly on the DSI whose matching scores are iteratively updated exploiting the two assumptions.

In the first step, cooperative approaches calculate initial DSI values that are derived from a local method (window-based correlation). Those initial matching scores are then refined by an iterative update function that applies two basic concepts. Firstly, the smoothness assumption is exploited by the use of support regions. In contrast to window-based correlation, these support regions do not operate on the image space, but on the space

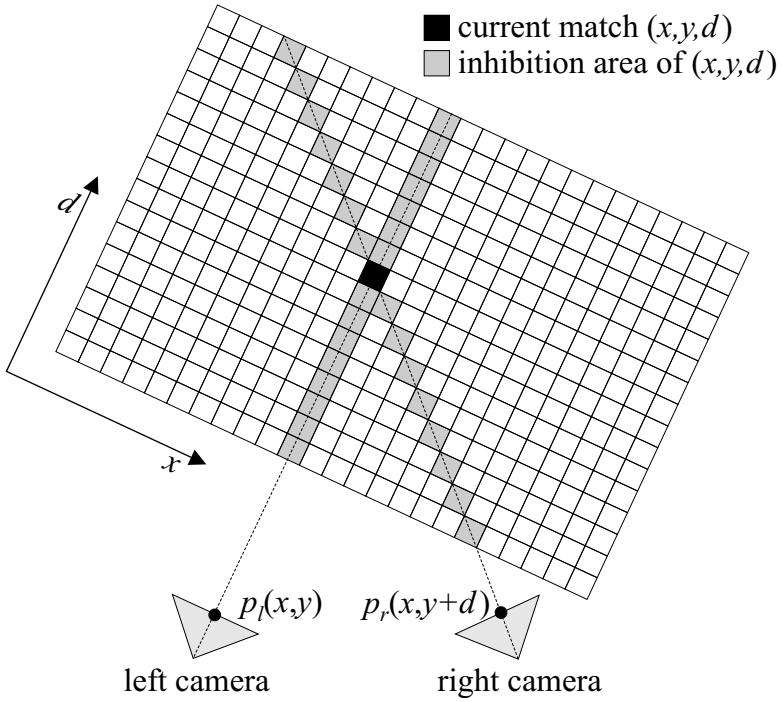


Figure 3.2: The concept of an inhibition area in the DSI. Assuming the validity of the uniqueness constraint, an established match (x, y, d) inhibits all other matches at the lines of sight of both cameras.

of correspondences, i.e. on the DSI. While Marr and Poggio [62, 63] employ two-dimensional support regions, Zitnick and Kanade [108] make use of three-dimensional ones in order to overcome perspective distortions caused by slanted surfaces. The matching values within a support region are then aggregated (e.g. by computing the average). This serves to propagate disparities having a high matching score.

Secondly, the uniqueness assumption is applied by using the concept of an inhibition area, which we illustrate in Figure 3.2. This concept is implemented by increasing the matching score of the strongest match, while the scores of all other potential matches on the corresponding lines of sight are reduced. A function that exploits the above ideas is used to simultaneously update all matching scores in the DSI. This process is then iterated until convergence. Upon convergence, the matching scores of each pixel in the left image are supposed to exhibit a strong unique peak at the correct disparity. In the final step, a pixel is therefore assigned to the disparity of maximum matching score. Since for occluded pixels no correct matching point exists,

they show low matching values at each allowed disparity. Occluded pixels can therefore be identified by comparing a pixel's highest matching score against a predefined threshold.

Cooperative approaches lie in the middle between local and global methods. On the one hand, they show global behaviour by propagating good disparities to ambiguous regions. On the other hand, they do not explicitly state a cost function that is subject to minimization. In practise, cooperative methods have shown to give strong results in various publications [104, 108]. Limitations include the higher computational effort compared to local methods. Furthermore, depth boundaries tend to be blurred, since rectangular support regions are employed. Zhang and Kambhamettu [104] try to overcome this problem by using image segmentation to estimate an appropriate support window. Cooperative stereo also depends on good initialization. While many incorrect matches may have large initial matching values, correct matches must have high initial matching scores [108].

3.1.4 Dynamic programming

Global approaches based on dynamic programming [10, 15, 25, 34, 70] match two horizontal scanlines individually, assuming the validity of the ordering constraint.[3] Their principle is illustrated in Figure 3.3. The figure shows two scanlines of the left and right images along with their colour values. Each cell in the two-dimensional array corresponds to a potential match of two pixels. A lot of those matches are prohibited, since they would result in a negative disparity or exceed a predefined limit on the maximum allowed disparity (set to 10 in the figure). Those matches are coloured grey. Each cell holds its associated matching costs. The illustrated array of cells therefore represents a shifted version of a DSI slice at a particular y-coordinate. The task of a stereo method based on dynamic programming is now to compute the minimum cost path that connects the two opposite corners C_s and C_e of the array.

As a consequence of the ordering assumption, there are only three directions in which a path is allowed to continue, i.e. diagonally, horizontally and vertically. A cell that is joint with its predecessor by a diagonal move represents a match at the same disparity with its predecessor. The costs of including a diagonal move into the path are then given by the corresponding matching costs. On the other hand, if a cell is connected with its predecessor by a horizontal or vertical move, this represents an occlusion. In this case,

[3] An implementation of a dynamic programming-based algorithm can also be found in Intel's Open Source Computer Vision Library [1].

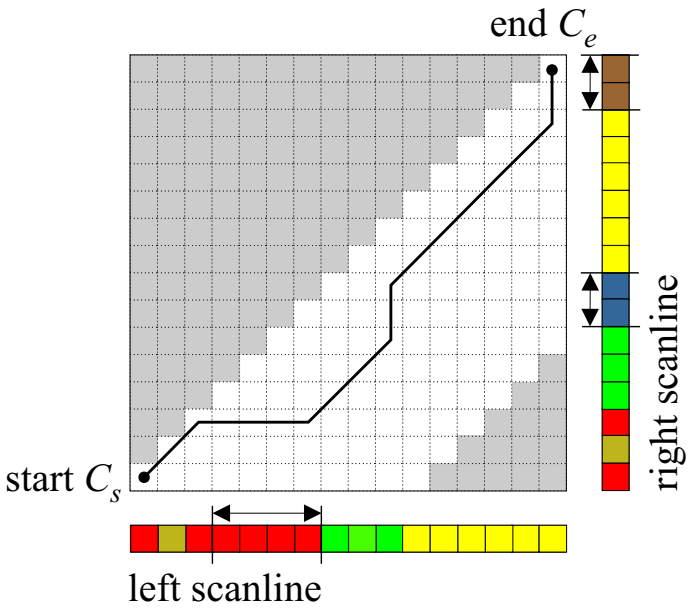


Figure 3.3: Stereo based on dynamic programming. The minimum cost path through a DSI slice is computed. Pixels that are declared as occluded (according to the computed path) are marked in both scanlines.

more than one pixel of one view projects to a single pixel of the other view. The costs for including such an occlusion move are given by a predefined constant value that serves to penalize occlusions.

The path that yields the global optimum of costs is efficiently computed using dynamic programming. The basic idea of dynamic programming is: If P is the optimal path between the starting cell C_s and the ending cell C_e then any subpath P' from C_s to an intermediate cell $C_i \in P$ must as well be the one of lowest costs. This idea is used to recursively compute optimal subpaths to each cell in the array. Knowing the diagonal, horizontal and vertical neighbours' costs, the optimal predecessor of a cell is calculated as the one whose costs plus the costs of the move joining these two cells are lowest. Each cell then records these costs as well as a link to its optimal predecessor. Once this computation is done for each cell, the final path is recovered by tracing back the stored links starting from the ending cell C_e and leading to the starting cell C_s. The final path represents the best set of matches that fulfil the ordering assumption.

Advantages of stereo algorithms based on dynamic programming include their capability to explicitly identify occlusions in both views and their low computational costs. Some implementations even allow for the computation

of dense disparity maps close to real-time [31, 36]. On the other hand, optimization via dynamic programming only works for one-dimensional cost functions, since pairs of scanlines have to be matched individually, one after the other[4]. Smoothness is therefore enforced within, but not across scanlines. As a consequence, the computed disparity maps suffer from horizontal streaks, which is especially true in regions of low texture. To eliminate horizontal streaks, a post-processing step can optionally be carried out [10]. This might, however, destroy correctly determined disparity values. A second problem of stereo via dynamic programming is the dependency on the validity of the ordering constraint. Since this assumption does not hold for thin foreground objects (see Section 2.2.2), disparity computation fails in those areas.

3.1.5 Graph-cut-based approaches

Graph-cut-based methods were originally introduced to overcome the weakness of dynamic programming, which lies in its one-dimensional smoothness term. To enforce two-dimensional smoothness, those approaches convert the stereo correspondence task into a maximum flow/minimum cut problem. The maximum flow problem is explained as follows. Let us consider a weighted directed graph $\mathcal{G}$ such as that of Figure 3.4. The graph contains two special vertices, called the source src and the sink snk. Directed edges of this graph are interpreted as water pipes and the edge weights represent the capacities of pipes that limit the amount of water that can flow through them. The maximum flow problem then refers to the question: Given that the source src pumps water into the network $\mathcal{G}$, what is the maximum flow of water that can reach the sink snk? We show a maximum flow solution for the specified graph by plotting the flow value at each edge (pipe) in Figure 3.4. It is well known that computing the maximum flow is equivalent to solving the minimum cut problem. Formally expressed, a cut denotes a partition of the vertices in $\mathcal{G}$ into two disjoint sets SRC and SNK with $src \in SRC$ and $snk \in SNK$. The costs of the cut are thereby defined as the summed-up capacities of those edges that go from SRC to SNK, and the minimum cut is the one of minimum costs. According to the theorem of Ford and Fulkerson [30], the maximum flow in the network is equal to the costs of the minimum cut. Once the maximum flow was computed, a minimum cut consists of a set of edges whose maximum capacities were reached in the maximum flow solution (Figure 3.4). Roughly spoken, the minimum cut problem is: What

[4]Recently, Veksler [95] has shown how to approximate the behaviour of a two-dimensional cost function by applying dynamic programming on a tree structure.

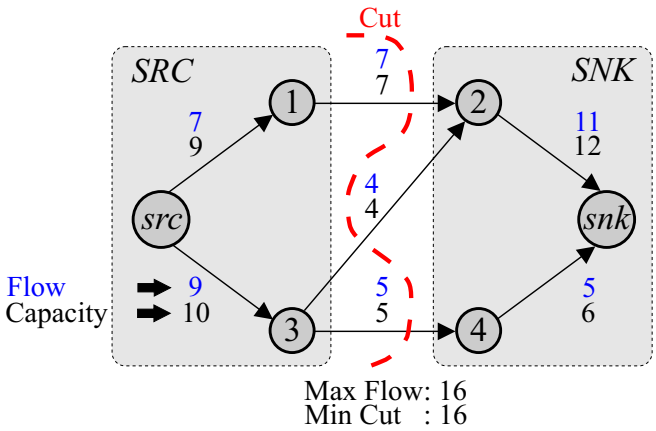


Figure 3.4: Duality of maximum flow and minimum cut. The maximum flow saturates a set of edges that build a minimum cut on the graph. The maximum flow value is equal to the costs of the minimum cut.

is the bottleneck in the network that limits the water flow? The maximum flow/minimum cut problem has been extensively studied in combinatorial optimization literature [16], and a variety of algorithms that compute optimal solutions in polynomial time exist.

To establish the connection between a minimum cut in a graph and the stereo correspondence problem, let us consider a simple example. In this example, the disparity of minimum costs for a single pixel p_1 is derived by computing the minimum cut in a special purpose graph. Given a disparity range from zero to three, we construct the graph illustrated in Figure 3.5a. Edges to one of the terminal nodes *src* and *snk* are given infinite weights, while edges between non-terminal nodes are weighted by the corresponding pixel dissimilarity at a particular disparity d. The thickness of the edges is thereby proportional to the computed dissimilarity in the figure. It is quite obvious that the minimum cut will identify the edge at disparity one as being the bottleneck in the illustrated graph. Moreover, it is possible to construct a graph so that the locally best disparity assignments for all pixels of the reference image are computed simultaneously by the minimum cut. We show such a construction for six pixels $p_1 - p_6$ of a scanline in Figure 3.5b. In fact, what we present here is just a parallel implementation of the winner-takes-all principle. That is, each pixel's disparity is selected based purely on the data and independent of neighbouring pixels' disparity assignments. However, we can include edges that model interactions between neighbouring pixels as shown in Figure 3.5c. In addition to the "disparity edges" (vertical ones),

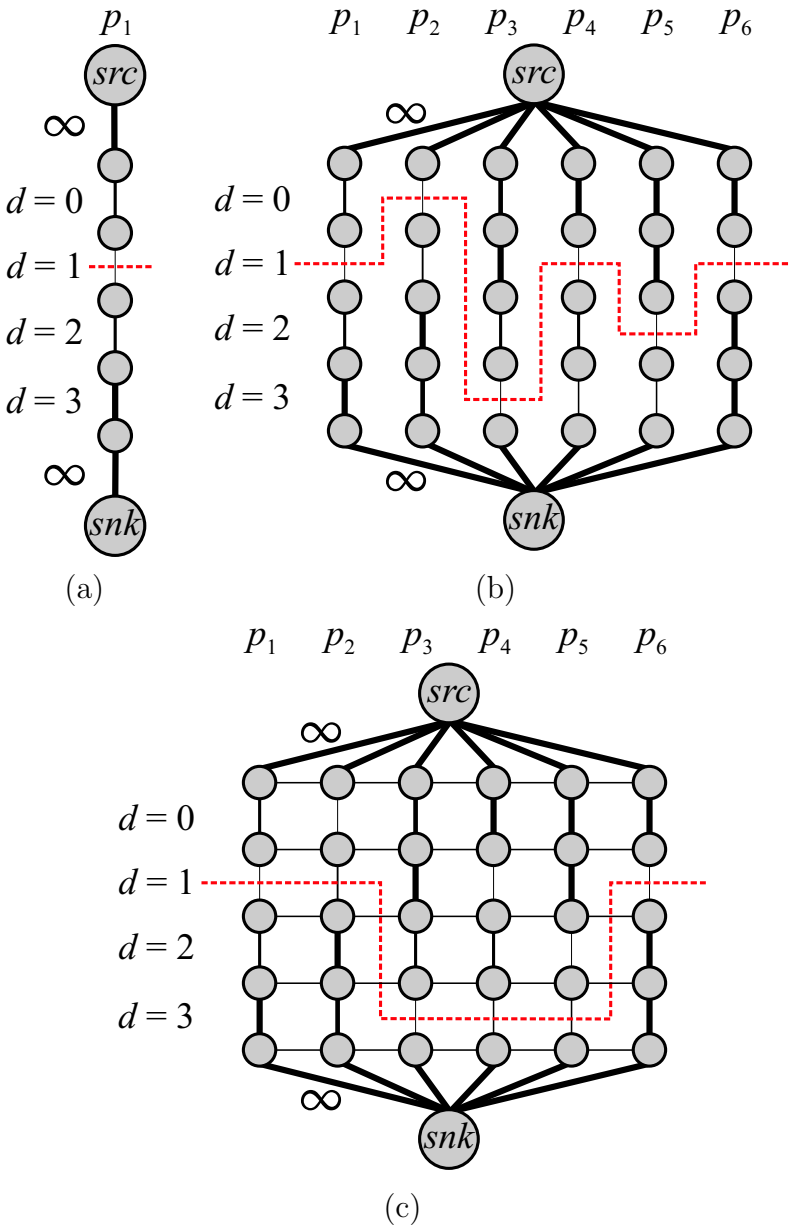
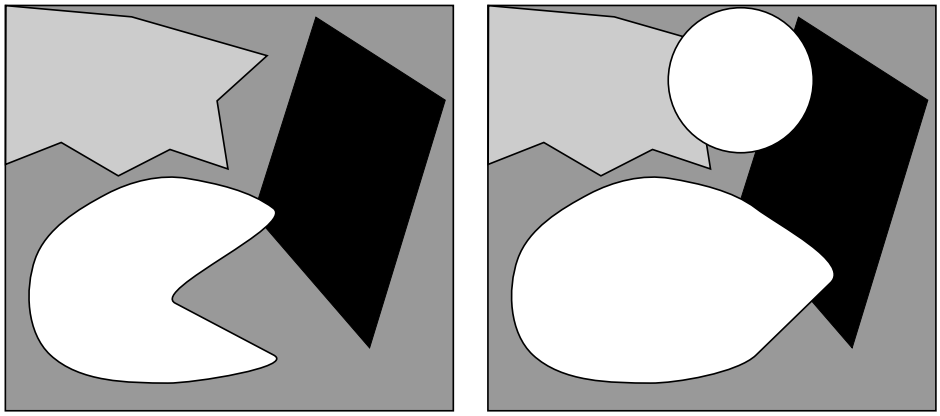


Figure 3.5: Graph-cuts in stereo vision. (a) A single pixel's disparity of lowest costs is computed by the minimum cut (red dotted line) in the illustrated graph. (b) The locally best disparities for multiple pixels are derived simultaneously. (c) Smoothness edges are inserted into the graph of (b) to bias the minimum cut towards smooth disparity solutions.

the graph now also contains a set of "smoothness edges" (horizontal ones) of a constant user-defined weight λ. The basic idea is that if the disparities d_1 and d_2 of neighbouring points change by $d_1 - d_2$ pixels, the minimum cut must include $|d_1 - d_2|$ smoothness edges. Since the smoothness edges contribute to the overall cut costs, a minimum cut will more likely be one that results in a smooth disparity reconstruction (such as the cut in Figure 3.5c). Note that, as opposed to the two-dimensional graph of the figure, the structure of the graph is in general a three-dimensional one. Smoothness edges do not only link pixels within the same horizontal scanline, but also neighbouring pixels of different scanlines. Graph-cut-based approaches are therefore capable of minimizing cost functions that involve a two-dimensional smoothness term.

The concept of computing a global optimal solution for the stereo problem with a single minimum cut in a special purpose graph was originally introduced by Roy and Cox [75]. In their work, they construct a graph that is very similar to the one that we illustrated in Figure 3.5c. Roy and Cox regard their algorithm as an extension of dynamic programming without explicitly stating a cost function. However, taking a closer look at the graph of Figure 3.5c, we can identify the cost function that is optimized by the minimum cut as being the sum of a data and a smoothness term. The data term sums up the pixel dissimilarities over all pixels at their current disparities, while the smoothness term imposes a penalty for neighbouring pixels of different disparity values. More precisely, for two neighbouring pixels assigned to disparities d_1 and d_2, the smoothness penalty is computed by $|d_1 - d_2| \cdot \lambda$ with λ being a user-defined value. This smoothness term is a convex function of the size of the jump in disparity. For cost functions involving a convex smoothness term, Ishikawa [42, 43] proves that a global minimum of costs is indeed reachable via graph-cuts in polynomial time. Exact optimization of such cost functions is the concept behind the graph-cut-based stereo methods of Ishikawa and Geiger [44] and Buehler et al. [20]. Although global minimal solutions are of theoretical interest, convex smoothness terms do not represent an optimal choice for the stereo problem. The costs of two small jumps in disparity are not higher than that of one large jump. The effect of this is that disparity values which lie inbetween the disparities of fore- and background are computed in the proximity of depth discontinuities. Depth boundaries therefore tend to be blurred.

To avoid the problem of overpenalizing large jumps in disparity, it is preferable to use a non-convex smoothness function. In its simplest form, such a function imposes a constant penalty λ whenever two neighbouring pixels show different disparities and returns 0 otherwise. This particular smoothness function is commonly referred to as the Potts model [73]. Since the smoothness penalty is given independently of the size of the disparity dis-



(a) (b)

Figure 3.6: The α-expansion move (after [52]). (a) Original label configuration. (b) α-expansion of the white label. The assignments of some pixels are changed to the white label, while the other pixels keep their old labels.

continuity, it is clear that two small disparity jumps now generate larger costs than a single large one. Therefore, non-convex functions are better suited to correctly reconstruct depth discontinuities. However, even the simplest discontinuity preserving smoothness function, i.e. the Potts model, results in a problem formulation whose optimization is proven to be $\mathcal{NP}$-complete [52]. However, Boykov et al. [17] show that a strong local optimum, which is guaranteed to lie within a known factor of the real optimum, can be calculated for non-convex smoothness terms by iterative application of their α-expansion move.

The α-expansion framework will be described very roughly in the following. More details are found in Chapter 7 where we present a stereo algorithm that uses the α-expansion algorithm for optimization of a global cost function. An α-expansion move basically changes the disparity assignments of a subset of pixels to the disparity α and leaves the other pixels assigned to their old disparities. Formally expressed, let f be the current configuration that labels each pixel of the reference image by a disparity value. The configuration f' is within one α-expansion move from f if for each pixel p, $f'(p) = \alpha$ or $f'(p) = f(p)$. We give an example of an α-expansion move in Figure 3.6. The optimal α-expansion move is the one that results in the largest improvement of costs. It is derived by computing the minimum cut on a weighted graph. Since the α-expansion move changes a large number of labels simultaneously, its application allows to compute a strong local minimum. Boykov et al. embed the α-expansion move into a greedy algorithm. Starting from

an arbitrary configuration, the algorithm computes the optimal α-expansion move for each allowed disparity in fixed or random order. If a move decreases the costs then this is the new label configuration. This procedure is iterated until there is no disparity that further decreases the costs by application of the α-expansion move.

The α-expansion framework is successfully applied on different problem formulations. Kolmogorov and Zabih [50] identify occlusions by enforcing the uniqueness constraint. Since the Potts model aims at generating piecewise constant disparities, the reconstruction of slanted surfaces is not optimal. Birchfield and Tomasi [11] therefore represent the scene by a set of planar layers, so that each pixel is assigned to a plane model instead of a discrete disparity value. Lin and Tomasi [57] extend this approach to handle occlusions in a symmetrical way. Furthermore, they apply a spline model to describe surfaces' disparities. Kolmogorov and Zabih [51] use the α-expansion algorithm to approximate the minimum of a cost function for multi-camera scene reconstruction. They explicitly account for occlusions. Recently, Hong and Chen [40] and Deng et al. [27] combined image segmentation with graph-cut-based optimization.

At the current state-of-the-art in stereo matching, global approaches using optimization via graph-cuts or belief propagation [84, 85] give the strongest experimental results. Belief propagation thereby represents an optimization scheme that can be used as an alternative to the α-expansion algorithm in order to approximately minimize global cost functions of the discussed type. We refer the reader to the paper of Tappen and Freeman [90] who compare graph-cuts against belief propagation. They approximate the solution to the same Markov Random Field using both methods. They conclude that graph-cuts produce solutions of lower costs, while belief propagation is potentially faster. However, from a computational viewpoint, both methods are quite efficient for what they do, but not yet suited for real time applications.

3.2 Evaluation of stereo methods

The absence of ground truth data often represents a limiting factor in computer vision. Fortunately, for the stereo correspondence problem, there exist ground truth disparities for a number of test image pairs of real scenes. Over the last couple of years, the Middlebury stereo vision benchmark of Scharstein and Szeliski [78] has become a commonly agreed way to measure the performance of a stereo algorithm. To evaluate a method, the authors provide a set of four test pairs along with the corresponding ground truth. These data sets are shown in Figure 3.7. Researchers who want to participate in the

test are asked to run their stereo algorithms on these image pairs using constant parameter settings. The resulting disparity maps are then compared against the corresponding ground truth.

For quantitative evaluation, Scharstein and Szeliski measure the percentage of wrong pixels, i.e. pixels whose absolute disparity error is larger than one. Error percentages are computed in three different image regions. First, they determine the percentage of wrong pixels over the complete image (*all*). Then they estimate the percentages of bad pixels in untextured regions (*untex.*) and regions close to disparity discontinuities (*disc.*), since those image areas are specifically challenging in stereo computation. In all three cases, only *unoccluded* pixels are considered. This is due to the reason that most stereo methods are not able to produce meaningful depth estimates for points affected by occlusion. The evaluated stereo algorithms are then ranked according to these error metrics. Table 3.1 shows the online version of the Middlebury stereo benchmark[5] at the time of writing this thesis. Methods are listed in descending order of their overall performance. To allow for a comparison between algorithms of different categories (according to our taxonomy of Section 3.1), we as well tabulate the optimization schemes applied by the approaches. These are: winner-takes-all (WTA), progressive optimization (PG), cooperative stereo (CO), dynamic programming (DP), graph-cuts (GC), belief propagation (BP) and others (O) for approaches that do not fit into any of those categories.

To demonstrate the characteristics of individual techniques, we pick out some results for the Tsukuba test set. As a representative of window-based matching using winner-takes-all optimization, we choose the fast correlation method of Mühlmann et al. [67] and show the computed disparity map in Figure 3.8a. Artefacts caused by untextured regions are clearly visible (e.g. right upper corner of the image). Moreover, depth discontinuities are poorly captured due to the edge fattening effect of window-based techniques. The results of the cooperative approach of Zitnick and Kanade [108] (Figure 3.8b) appear much smoother, but still suffer from imprecisely reconstructed depth boundaries, since rectangular support windows are used. This problem is overcome by the dynamic programming algorithm implemented by Scharstein and Szeliski [78] that uses pixels as matching primitives. However, as seen from Figure 3.8c, the scanline streaking effect represents a severe limitation, which is also visible in the bad quantitative results of this technique. In Figure 3.8d, we present the results of the graph-cut approach of Roy and Cox [75] that optimizes a two-dimensional smoothness term. Since this smoothness term is a convex function, the technique generates wrong "inbe-

[5] http://www.middlebury.edu/stereo/


Tsukuba left image Tsukuba right image Ground truth


Sawtooth left image Sawtooth right image Ground-truth

Venus left image Venus right image Ground truth

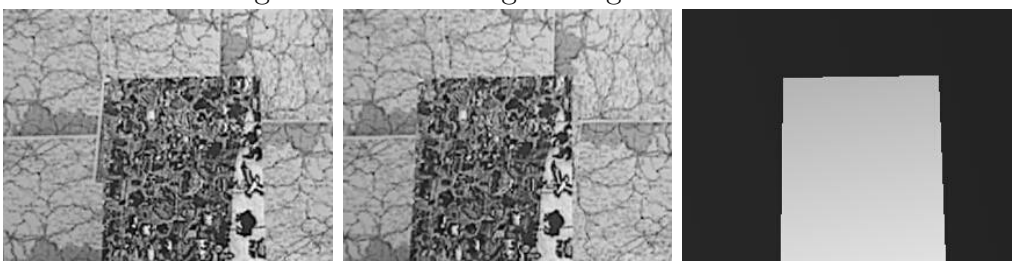

Map left image Map right image Ground truth

Figure 3.7: The Middlebury data sets with ground truth used in the study of Scharstein and Szeliski [78]. Disparities are encoded with bright pixels representing large disparity values and dark pixels corresponding to low ones.

Algorithm - Optimization		Tsukuba			Sawtooth			Venus			Map	
		all	*untex.*	*disc.*	*all*	*untex.*	*disc.*	*all*	*untex.*	*disc.*	*all*	*disc.*
Sym.BP+occl. [84]	BP	0.97$_{2}$	0.28$_{3}$	5.45$_{2}$	0.19$_{1}$	0.00$_{1}$	2.09$_{1}$	0.16$_{4}$	0.02$_{3}$	2.77$_{5}$	0.16$_{1}$	2.20$_{1}$
Patch-based [27]	GC	0.88$_{1}$	0.19$_{1}$	4.95$_{1}$	0.29$_{5}$	0.00$_{1}$	3.23$_{5}$	0.09$_{2}$	0.02$_{3}$	1.50$_{2}$	0.30$_{8}$	4.08$_{10}$
Segm.-based GC [40]	GC	1.23$_{5}$	0.29$_{5}$	6.94$_{5}$	0.30$_{6}$	0.00$_{1}$	3.24$_{6}$	0.08$_{1}$	0.01$_{1}$	1.39$_{1}$	1.49$_{27}$	15.4$_{31}$
Graph+segm.	GC	1.39$_{8}$	0.28$_{3}$	7.17$_{7}$	0.25$_{4}$	0.00$_{1}$	2.56$_{3}$	0.11$_{3}$	0.02$_{2}$	2.04$_{3}$	2.35$_{31}$	20.8$_{35}$
Segm.+glob.vis.	O	1.30$_{6}$	0.48$_{8}$	7.50$_{10}$	0.20$_{2}$	0.00$_{1}$	2.30$_{2}$	0.79$_{7}$	0.81$_{10}$	6.37$_{12}$	1.63$_{29}$	16.0$_{33}$
Belief prop. [85]	BP	1.15$_{3}$	0.42$_{7}$	6.31$_{3}$	0.98$_{13}$	0.30$_{20}$	4.83$_{11}$	1.00$_{10}$	0.76$_{9}$	9.13$_{18}$	0.84$_{22}$	5.27$_{14}$
Layered [57]	GC	1.58$_{12}$	1.06$_{15}$	8.82$_{13}$	0.34$_{7}$	0.00$_{1}$	3.35$_{7}$	1.52$_{18}$	2.96$_{28}$	2.62$_{4}$	0.37$_{13}$	5.24$_{13}$
2-pass DP [49]	DP	1.53$_{11}$	0.66$_{11}$	8.25$_{12}$	0.61$_{9}$	0.02$_{9}$	5.25$_{12}$	0.94$_{8}$	0.95$_{11}$	5.72$_{11}$	0.70$_{20}$	9.32$_{21}$
Region-Progress. [36]	PG	1.44$_{9}$	0.55$_{9}$	8.18$_{11}$	0.24$_{3}$	0.00$_{1}$	2.64$_{4}$	0.99$_{9}$	1.37$_{15}$	6.40$_{13}$	1.49$_{28}$	17.1$_{34}$
GC+occl. [50]	GC	1.19$_{4}$	0.23$_{2}$	6.71$_{4}$	0.73$_{12}$	0.11$_{12}$	5.71$_{15}$	1.64$_{21}$	2.75$_{26}$	5.41$_{9}$	0.61$_{18}$	6.05$_{16}$
MultiCam GC [51]	GC	1.85$_{16}$	1.94$_{21}$	6.99$_{6}$	0.62$_{11}$	0.00$_{1}$	6.86$_{17}$	1.21$_{15}$	1.96$_{18}$	5.71$_{10}$	0.31$_{10}$	4.34$_{12}$
Improved Coop. [64]	CO	1.67$_{13}$	0.77$_{12}$	9.67$_{17}$	1.21$_{18}$	0.17$_{15}$	6.90$_{19}$	1.04$_{11}$	1.07$_{12}$	13.6$_{24}$	0.29$_{6}$	3.65$_{7}$
Adapt. weights [102]	WTA	1.51$_{10}$	0.65$_{10}$	7.24$_{8}$	1.14$_{16}$	0.27$_{18}$	5.48$_{13}$	1.14$_{12}$	0.61$_{8}$	4.49$_{6}$	1.47$_{26}$	13.5$_{29}$
Symbiotic [37]	O	2.87$_{21}$	1.71$_{20}$	11.9$_{19}$	1.04$_{14}$	0.13$_{13}$	7.32$_{21}$	0.51$_{5}$	0.23$_{5}$	7.88$_{15}$	0.50$_{16}$	6.54$_{18}$
Disc. pres. [3]	O	1.78$_{15}$	1.22$_{17}$	9.71$_{18}$	1.17$_{17}$	0.08$_{11}$	5.55$_{14}$	1.61$_{20}$	2.25$_{21}$	9.06$_{17}$	0.32$_{11}$	3.33$_{6}$
Var. win. [94]	WTA	2.35$_{19}$	1.65$_{19}$	12.1$_{21}$	1.28$_{19}$	0.23$_{17}$	7.09$_{20}$	1.23$_{16}$	1.16$_{13}$	13.3$_{22}$	0.24$_{4}$	2.98$_{4}$
GC α-β-swap [78]	GC	1.94$_{18}$	1.09$_{16}$	9.49$_{16}$	1.30$_{20}$	0.06$_{10}$	6.34$_{16}$	1.79$_{24}$	2.61$_{25}$	6.91$_{14}$	0.31$_{9}$	3.88$_{8}$
Reliability-DP [36]	DP	1.36$_{7}$	0.81$_{13}$	7.35$_{9}$	1.09$_{15}$	0.44$_{22}$	4.13$_{9}$	2.35$_{26}$	2.37$_{23}$	13.5$_{23}$	0.55$_{17}$	6.14$_{17}$
Multiw. cut [11]	GC	8.08$_{35}$	6.53$_{31}$	25.3$_{36}$	0.61$_{10}$	0.46$_{24}$	4.60$_{10}$	0.53$_{6}$	0.31$_{6}$	8.06$_{16}$	0.26$_{5}$	3.27$_{5}$
GC α-expansion [17]	GC	1.86$_{17}$	1.00$_{14}$	9.35$_{14}$	0.42$_{8}$	0.14$_{14}$	3.76$_{8}$	1.69$_{23}$	2.30$_{22}$	5.40$_{8}$	2.39$_{33}$	9.35$_{22}$
Tree DP [95]	DP	1.77$_{14}$	0.38$_{6}$	9.48$_{15}$	1.44$_{23}$	0.84$_{27}$	6.87$_{18}$	1.21$_{14}$	1.41$_{16}$	5.04$_{7}$	1.45$_{25}$	13.0$_{28}$
4-State DP [26]	DP	4.70$_{29}$	3.68$_{26}$	21.0$_{32}$	1.43$_{22}$	0.17$_{15}$	13.9$_{29}$	1.18$_{13}$	0.59$_{7}$	17.9$_{27}$	0.30$_{7}$	4.23$_{11}$
Comp. win. [93]	WTA	3.36$_{24}$	3.54$_{24}$	12.9$_{24}$	1.61$_{26}$	0.45$_{23}$	7.87$_{22}$	1.67$_{22}$	2.18$_{19}$	13.2$_{21}$	0.33$_{12}$	3.94$_{9}$
Realtime [39]	WTA	4.25$_{28}$	4.47$_{29}$	15.0$_{28}$	1.32$_{21}$	0.35$_{21}$	9.21$_{23}$	1.53$_{19}$	1.80$_{17}$	12.3$_{19}$	0.81$_{21}$	11.3$_{26}$
Cooperative [108]	CO	3.49$_{25}$	3.65$_{25}$	14.7$_{26}$	2.03$_{27}$	2.29$_{31}$	13.4$_{28}$	2.57$_{29}$	3.52$_{29}$	26.3$_{35}$	0.22$_{3}$	2.37$_{2}$
Relax+occl. [18]	O	6.33$_{33}$	6.63$_{32}$	22.9$_{33}$	1.51$_{25}$	0.29$_{19}$	15.0$_{34}$	1.44$_{17}$	1.24$_{14}$	19.1$_{30}$	0.43$_{14}$	5.99$_{15}$
Bay. diff. [78]	O	6.49$_{34}$	11.6$_{37}$	12.2$_{22}$	1.45$_{24}$	0.72$_{25}$	9.29$_{24}$	4.00$_{31}$	7.21$_{33}$	18.3$_{29}$	0.20$_{2}$	2.49$_{3}$
Stoch. diff. [54]	O	3.95$_{26}$	4.08$_{28}$	15.4$_{30}$	2.45$_{31}$	0.90$_{28}$	10.5$_{25}$	2.45$_{27}$	2.41$_{24}$	21.8$_{32}$	1.31$_{24}$	7.79$_{20}$
Genetic [35]	O	2.96$_{22}$	2.66$_{23}$	14.9$_{27}$	2.21$_{29}$	2.76$_{33}$	13.9$_{30}$	2.49$_{28}$	2.89$_{27}$	23.0$_{33}$	1.04$_{23}$	10.9$_{25}$
SSD+MF [78]	WTA	5.23$_{32}$	3.80$_{27}$	24.6$_{35}$	2.21$_{28}$	0.72$_{26}$	13.9$_{31}$	3.74$_{30}$	6.82$_{32}$	12.9$_{20}$	0.66$_{19}$	9.35$_{22}$
Pix-to-pix [10]	DP	5.12$_{31}$	7.06$_{35}$	14.6$_{25}$	2.31$_{30}$	1.79$_{29}$	14.9$_{33}$	6.30$_{34}$	11.3$_{36}$	14.5$_{25}$	0.50$_{15}$	6.83$_{19}$
Max flow [75]	GC	2.98$_{23}$	2.00$_{22}$	15.1$_{29}$	3.47$_{32}$	3.00$_{34}$	14.1$_{32}$	2.16$_{25}$	2.24$_{20}$	21.7$_{31}$	3.13$_{34}$	15.9$_{32}$
Scanl. opt. [78]	O	5.08$_{30}$	6.78$_{33}$	11.9$_{20}$	4.06$_{33}$	2.64$_{32}$	11.9$_{26}$	9.44$_{37}$	14.5$_{37}$	18.2$_{28}$	1.84$_{30}$	10.2$_{24}$
Dyn. prog. [78]	DP	4.12$_{27}$	4.63$_{30}$	12.3$_{23}$	4.84$_{36}$	3.71$_{36}$	13.2$_{27}$	10.1$_{38}$	15.0$_{38}$	17.1$_{26}$	3.33$_{35}$	14.0$_{30}$
Realtime DP [31]	DP	2.85$_{20}$	1.33$_{18}$	15.6$_{31}$	6.25$_{38}$	3.98$_{37}$	25.1$_{36}$	6.42$_{35}$	8.14$_{34}$	25.3$_{34}$	6.45$_{37}$	25.1$_{36}$
MMHM [67]	WTA	9.76$_{37}$	13.8$_{38}$	24.3$_{34}$	4.76$_{35}$	1.87$_{30}$	22.4$_{35}$	6.48$_{36}$	10.3$_{35}$	31.2$_{36}$	8.42$_{38}$	12.6$_{27}$
Shao [80]	O	9.67$_{36}$	7.04$_{34}$	35.6$_{37}$	4.25$_{34}$	3.19$_{35}$	30.1$_{38}$	6.01$_{33}$	6.70$_{31}$	43.9$_{38}$	2.36$_{32}$	33.0$_{38}$
Max. surf. [83]	DP	11.1$_{38}$	10.7$_{36}$	41.9$_{38}$	5.51$_{37}$	5.56$_{38}$	27.3$_{37}$	4.36$_{32}$	4.78$_{30}$	41.1$_{37}$	4.17$_{36}$	27.8$_{37}$

Table 3.1: The Middlebury stereo test bed [78]. The algorithms are listed roughly in order of their overall performance. Large numbers give the percentage of wrong pixels, while small numbers indicate the rank. *Graph+segm.* and *Segm.+glob.vis.* are those methods presented in this thesis.

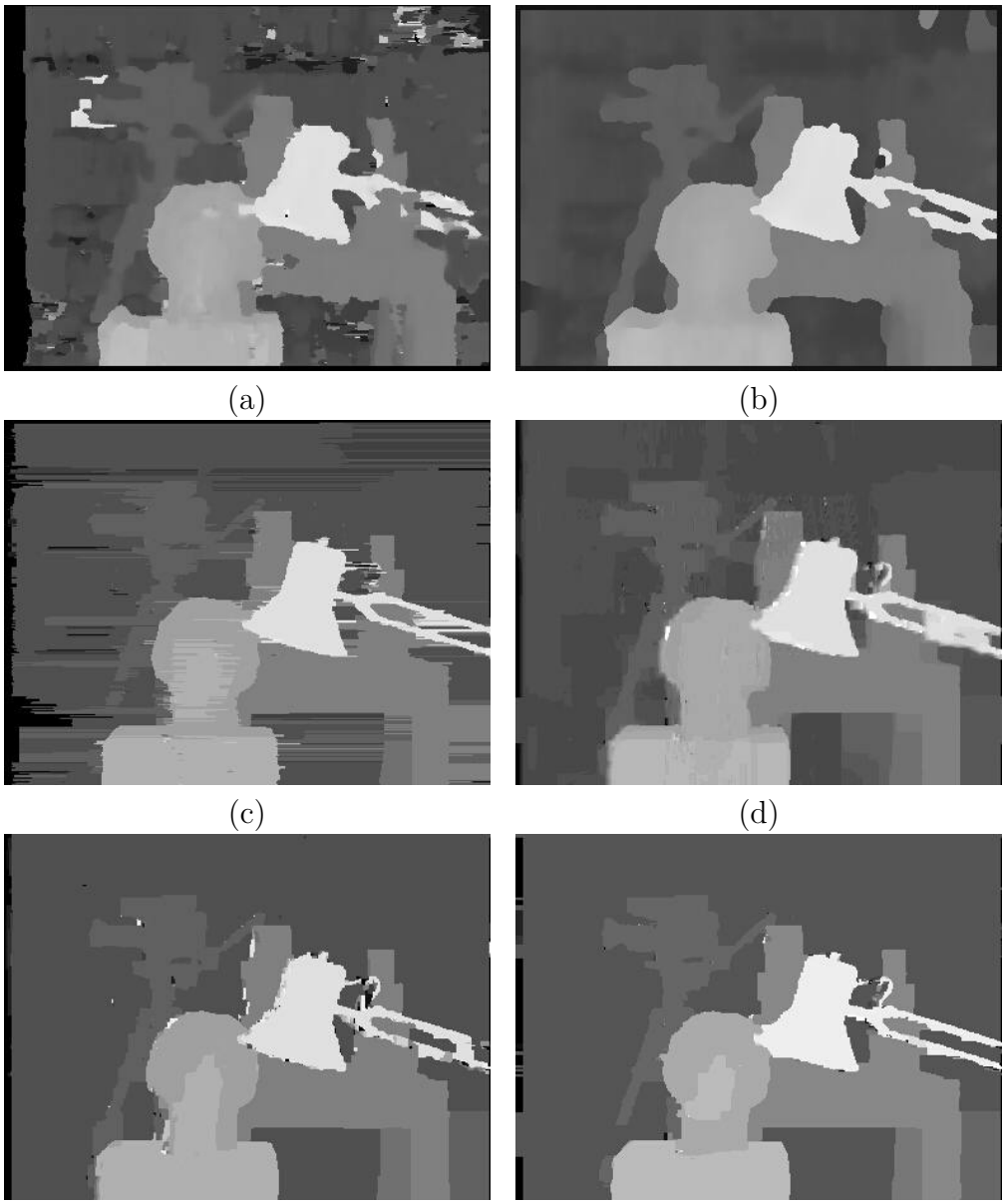


Figure 3.8: Results of different methods for the Tsukuba data set. (a) Window-based correlation [67]. (b) Cooperative approach [108]. (c) Dynamic programming [78]. (d) Maximum flow formulation [75]. (e) α-expansion algorithm [17]. (f) Graph-cuts with occlusions [50].

tween disparities" in the proximity of disparity borders (e.g. contour of the lamp). This can also be seen in the large error percentages in regions close to depth discontinuities (see Table 3.1). The situation is improved in Figure 3.8e. This figure shows the disparity map computed by the α-expansion algorithm of Boykov et al. [17] that optimizes a non-convex smoothness function. However, occlusions are not dealt with, and, therefore, pixels of occluded areas are assigned to more or less random disparity values. In contrast to this, Kolmogorov and Zabih [50] explicitly identify occlusions in their graph-cut-based algorithm. Occluded areas are then filled in with disparities from surrounding regions in a postprocessing step. The resulting disparity map is shown in Figure 3.8f.

3.3 Prior work on motion

Although the major focus of this thesis lies on the stereo correspondence problem, we will show how to extend our stereo algorithms to the task of optical flow (or motion) computation in later chapters of this work. For completeness, we give a brief overview on the motion correspondence problem and on standard methods used to solve this task in the following.

In the case of motion analysis, a single camera records subsequent frames of a moving scene. In this setup, the motion correspondence problem can roughly be stated as: "What went where?" In a more formal problem description, we define the task of a motion algorithm as the automatic reconstruction of a dense field of displacement vectors that transform one image into the next image of a sequence [41]. This array of vectors is commonly referred to as *optical flow*, and as opposed to stereo, such vectors are, in general, *two-dimensional*. Accurate estimation of the optical flow field plays a key-role in various computer vision applications, including motion detection and segmentation, frame interpolation, three-dimensional scene reconstruction, robot navigation, video shot detection, mosaicking and video compression.

From its definition, optical flow is computed as the displacement of brightness patterns over time. As a consequence, the optical flow field does not necessarily correspond to the real motion within the sequence. An often used example in this context is that of a rotating sphere of uniform colour [12]. Although there is motion on the sphere, its projections onto the images of a video sequence show constant brightness patterns. This results in zero optical flow. What is therefore computed by an optical flow algorithm represents the *apparent* motion. However, natural scenes usually have enough structure, so that the optical flow field represents a good approximation of the real displacements.

From a computational point of view, the optical flow problem can be regarded as more difficult than stereo. Note that the larger search range (due to the two-dimensional displacement vector) does not only lead to a higher computational effort, but also increases the chances of finding an incorrect match. However, due to the closely related nature of the two problems, the methods used to solve the motion correspondence problem show high similarity with those algorithms applied in stereo computation. For example, the concept of window-based matching, which we have discussed in the context of stereo in Section 3.1.1, is widely adopted in optical flow computation as well (e.g., [5, 45, 58]). In principle, every motion algorithm can be used to compute the disparities of a stereo pair (while not every stereo method can be extended to compute motion). In the following, we give a review of differential approaches, which represent the most popular way to tackle the optical flow problem. We then focus on parametric motion algorithms and motion segmentation methods that are most closely related to the algorithms of this thesis. Our review is not intended to be exhaustive, and for a more elaborate summary, the reader is referred to several review papers [7, 65, 66, 82]. Note, however, that it is difficult to give a quantitative comparison of optical flow methods, since ground truth data for real image sequences is normally not available.

3.3.1 Differential approaches

The basic assumption of differential approaches is that the intensity value of a pixel is approximately constant under motion. Let $I(x, y, t)$ denote the continuous intensity function of a point with x and y being its spatial coordinates and t being the time. The assumption of constant intensity can then be expressed by $I(x, y, t) = I(x + u, y + v, t + 1)$ with (u, v) being the pixel's displacement at time $t + 1$. The vector (u, v) therefore represents the optical flow at image point (x, y). After expansion in a Taylor series[6] we derive: $I_x \cdot u + I_y \cdot v + I_t = 0$ where I_x, I_y, I_t denote the partial derivatives. This equation is known as the *optical flow constraint equation* [41].

The optical flow constraint equation is underdetermined, since it has two unknowns: u and v. This issue is commonly referred to as *aperture problem*. As a consequence of the aperture problem, it is, in general, only possible to compute the flow vector that is normal to the image edges. In order to overcome this problem, an additional constraint is required. This constraint is usually formed by the smoothness assumption. Similar to our categorization of stereo algorithms in Section 3.1, differential optical flow methods

[6]Details are given in [7].

are divided between *local* and *global* approaches[7] based on the way how this smoothness constraint is employed.

Local approaches [8, 59] assume that the flow vectors are constant within a support window of user-defined size. For example, the Lucas-Kanade algorithm [59] combines the optical flow constraint equations of points within the support region. The method then computes the flow vector of a pixel using least squared error regression. However, the problem of this methodology is that in untextured image areas the window does not capture feature points that would help to resolve matching ambiguities. Moreover, choosing the optimal window size represents a crucial decision (see Section 3.1.1).

As opposed to local techniques, global approaches explicitly state the smoothness assumption in terms of a global energy functional. Methods of this category include the well-known approach of Horn and Schunck [41] as well as various other approaches that optimize a *discontinuity preserving* smoothness term [13, 68, 74, 98]. Although global approaches show results of good quality, they pose complex optimization problems. Moreover, they are relatively sensitive to image noise [7], which is the reason why the input images are commonly preprocessed by a smoothing filter.

The major disadvantage of differential approaches lies in that they do, in general, not model the occlusion problem. Most global differential approaches enforce that every point is assigned to an optical flow vector. As a consequence, those methods cannot express the fact that a point's matching pixel does not exist (which would be required for occlusion handling).

3.3.2 Parametric methods and motion segmentation

Instead of assigning each pixel to a fixed-valued flow vector, parametric methods use models to describe the image motion. (An example of such a parametric model is the affine one.) Motion models enforce strong constraints on the motion present in the scene [14], and if the correct models are available, this results in a more accurate reconstruction of optical flow. The advantage of parametric techniques lies in that they can robustly describe large image areas by a single set of parameters. This is also the reason why there is a strong interrelationship between parametric approaches and motion segmentation algorithms.

Motion segmentation algorithms segment an image based on the optical flow information. Their basic assumption is that image regions which can be well represented by the same set of motion parameters originate from a

[7] Bruhn et al. [19] have recently presented a hybrid method that combines both strategies.

single moving object. The success of a motion segmentation algorithm therefore clearly depends on the quality of motion estimates. On the other hand, optical flow computation can be significantly improved, if an accurate segmentation of the image into regions of homogenous motion is available. As a consequence, it makes sense to compute optical flow and motion segmentation simultaneously [11, 22, 28]. Moreover, parametric optical flow techniques (such as those presented in this thesis) often generate motion segmentation as a by-product and vice versa. In the following, we focus on algorithms that we see as closest related to our work.

Black and Jepson [14] propose a parametric optical flow algorithm that divides the reference image into regions of homogenous colour. They compute an initial dense optical flow field and estimate a variable order parametric model for each individual colour segment using the initial flow estimates. To determine the number of each model's parameters (two, six or eight), the authors warp the segment to the second view and measure the registration error. In a subsequent step, each model is refined by applying standard area-based regression techniques. Note that each colour segment is described by its own parametric model. No attempt is taken to represent several segments by a single motion model, which might, however, increase the algorithms robustness, especially in low-textured image areas.

Wang and Adelson [96] popularized the concept of a layered representation in order to perform motion segmentation. The authors uniformly distribute a large number of seed blocks over the image. An affine motion model for each block is then fitted to a precomputed optical flow field. To extract those motion models that describe the different motions in the scene (these are called *layers*), the authors use a k-means clustering algorithm. Once the layers are known, each block is assigned to the layer model that shows the highest agreement with the initial motion estimates and a new image partition is computed. The algorithm is then iterated until convergence. In a similar spirit, Ke and Kanade [48] use a clustering approach for layer extraction. They compute an affine model for each segment that is derived by colour segmentation. Layers are then identified by mean-shift-based clustering in a low-dimensional linear subspace. However, the drawback of both methods is that they solely rely on the precomputed optical flow field (which is, more than likely, corrupted by erroneous matching results). More specifically, those methods do not impose spatial smoothness assumptions, and therefore the segmentation results contain isolated image regions. In an attempt to overcome this problem, Altunbasak et al. [4] present a colour segmentation-based algorithm that uses k-means clustering. This approach is similar to that of Wang and Adelson [96]. The authors, however, include an additional spatial smoothness constraint into the clustering process.

In the context of motion segmentation via global cost minimization, Ayer and Sawhney [6] employ the minimum description length (MDL) encoding principle in order to derive the smallest set of layers necessary to describe the image motion. They formulate statistical cost functions for the layer extraction and assignment tasks that are optimized by an expectation maximization algorithm. Several other techniques that use probabilistic cost formulations include [71, 89, 107]. Willis et al. [99] present a graph-cut-based approach to achieve a dense and piecewise smooth assignment of pixels to layers. Although the authors use their algorithm on image pairs of large inter-frame motion, they do not explicitly model the occlusion problem in the assignment step. In contrast to this, Xiao and Sha [101] show how to embed occlusion detection into a graph-cut-based method in a very recent work. In their paper, they claim to be the first ones to deal with the explicit identification of occluded pixels for the motion segmentation task.

3.4 Summary

In this chapter, we have presented prior work on the stereo correspondence problem. Although there are potentially many ways to build a taxonomy of stereo methods, we have decided to focus on the optimization component and divided algorithms into local and global approaches.

Starting with the local methods, we have presented the principle of window-based correlation. We have pointed out the problem of finding the correct search window sizes and shapes, which represents a trade-off between obtaining good disparities in low textured regions versus precisely outlining depth boundaries. We have then continued our review with progressive approaches. Those methods estimate a set of reliable points, which are then used to rule out potential subsequent matches. Between local and global methods, we have identified cooperative approaches that iteratively refine matching scores using the smoothness and uniqueness constraints.

As a first truly global technique, we have then presented dynamic programming. The principle of dynamic programming is to compute a lowest cost path in a DSI slice relying on the validity of the ordering constraint. Although dynamic programming is computationally efficient, it cannot be applied to optimize an objective function involving a two-dimensional smoothness term. In contrast to this, graph-cut-based approaches represent an effective mean to optimize cost functions of this type. While for convex smoothness functions even a global optimum of costs is reachable, it has turned out that those formulations are not well suited for the stereo problem. We have therefore presented the α-expansion algorithm that effectively computes a

local optimum of objective functions with non-convex smoothness terms.

To evaluate the performance of the different approaches, we have described the Middlebury stereo benchmark. Furthermore, we have presented some example results to show the characteristics of individual methods.

Since parts of this thesis will deal with the optical flow problem, we have then given a short introduction to this problem and summarized standard approaches that compute a dense optical flow field. The most popular algorithms in this respect are differential approaches. These methods make use of the optical flow constraint equation. Since this equation is underdetermined, the smoothness assumption is commonly added as an additional constraint. Based on the way how this assumption is applied, differential optical flow methods are divided between local and global ones. We have then focused on parametric optical flow and motion segmentation algorithms in order to provide an overview of methods that we consider as closest related to those motion algorithms presented in this thesis.

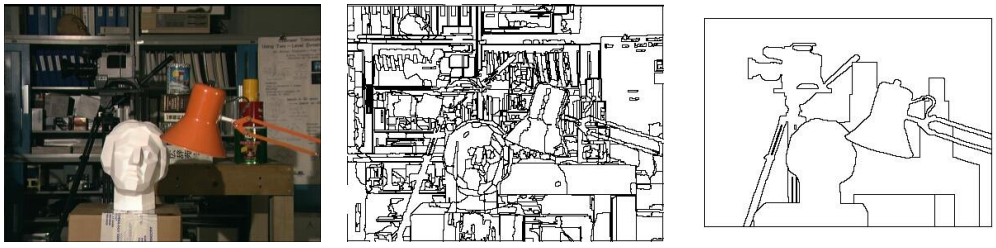
Chapter 4

Segmentation-based matching

4.1 Basic concept

As pointed out in the previous chapters, binocular depth computation is known to be difficult in some image regions (poor texture, occlusions, etc.). However, besides binocular cues (disparity), we can gain additional information from depth features that are present when only one image is available. This is backed by the fact that the human visual system is very skilled in the interpretation of such monocular cues. For example, when looking at the image in Figure 4.1a, the human observer can effortlessly get an impression of the scene's depth without having the second view at hand. An often employed monocular cue in computational stereo is based on the observation that discontinuities in depth commonly go along with discontinuities in the intensity image. Often, this observation is used to align disparity discontinuities with intensity edges (e.g., [10, 15, 56]). However, it also builds the foundation for segmentation-based methods that have recently gained attention for their strong experimental results.

Segmentation-based approaches divide one or sometimes both images into non-overlapping regions of homogeneous colour. Instead of computing a disparity for each individual pixel, those techniques assign a single disparity value (or model) to a complete segment. Therefore, those methods implicitly apply two basic assumptions: Firstly, inside a segment of homogeneous colour the disparity values are expected to follow some particular smooth disparity model (constant disparity, planar model, etc.). This is justified by the observation that pixels inside the same colour region most likely originate from the same object in the scene. Since, in general, real world objects are made up of smooth surfaces, also the disparity values inside the region are expected to vary smoothly. Secondly, disparity discontinuities are assumed



(a) (b) (c)

Figure 4.1: The segmentation assumption. (a) Left view of the Tsukuba image pair. (b) Result of a colour segmentation. Segment borders are shown. (c) Depth discontinuities computed from the ground truth.

to coincide with the boundaries of those colour regions. This assumption obviously reflects the above observation that depth discontinuities are located at intensity edges. Both assumptions usually hold true for images of natural scenes. To provide the reader with an idea of what such a segmentation may look like, we present the result of an off-the-self segmentation algorithm on the Tsukuba reference image in Figure 4.1b. In addition, we show the depth discontinuities computed from the ground truth image in Figure 4.1c in order to allow for an easier comparison against the extracted segment borders.

For the sake of clarity, throughout this thesis, a *segment* denotes an image region of homogeneous colour that was derived by colour segmentation. In principle, however, one could as well use different segmentation strategies (e.g., texture segmentation). The important point is that the computed segments do not overlap a disparity discontinuity.

4.2 Advantages and disadvantages

Recently, region-based algorithms have become popular in the stereo community. Although quite different from each other, all methods of this category take benefit of the segmentation information to increase their robustness in traditionally challenging areas in stereo computation. This is well reflected by the good experimental results of those techniques. For example, all of the five best ranked algorithms in the Middlebury database [78] (see Table 3.1) apply colour segmentation. We identify the advantages of region-based stereo as follows:

1. The probably most obvious advantage is that region-based stereo techniques constrain the disparity inside a region to follow a single disparity

model. In other words, smoothness within a segment is explicitly enforced. This is advantageous, since it allows the assignment of smooth disparity values in regions of poor texture.

2. Often, disparity boundaries can be more accurately identified by the use of monocular cues (such as the partition of the reference image into regions of homogeneous colour) than this would be possible using disparity only. We point out that a lot of stereo algorithms implicitly or explicitly aim at the reconstruction of simple object shapes. More precisely, they bias towards minimizing the length of object borders. Consequently, this decreases their performance in the presence of more complex outlines. Segmentation can represent a remedy to this problem.

3. The robustness in areas affected by occlusion is improved. In theory, matching might even succeed for a segment that is partially occluded, since it is still possible to match the segment's non-occluded pixels. However, this does not mean that occlusions can be ignored, as we will explain in more detail in the next section. Note that since a single disparity model is assigned to the *complete* segment, also those parts that are affected by occlusion are automatically filled in with some "meaningful" disparity.

4. The number of segments is usually significantly smaller than the number of pixels. This gives rise to potentially much faster stereo algorithms.

Nevertheless, using the segmentation assumption obviously also involves some disadvantages:

1. The most severe problem associated with region-based approaches is that the segmentation assumption is, in general, not guaranteed to hold true. More precisely, the success of such methods depends on the ability of the segmentation algorithm to accurately delineate the objects outlines. It is therefore safer to apply oversegmentation.

2. The disparity model can be inappropriate to represent the "real" disparity of a segment. This is, of course, rather a problem of using a model and not specifically bound to the segmentation aspect. However, choosing an appropriate model is a difficult task by itself. While simple models may oversimplify the real disparity, complex models may overfit the data and show undesired effects due to image noise.

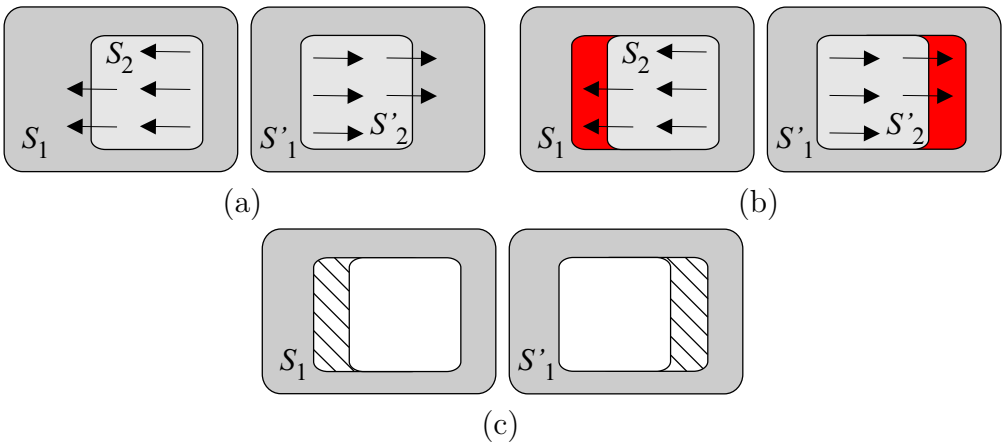


Figure 4.2: The occlusion problem in segmentation-based matching. Explanation is given in the text.

3. Modelling the stereo problem on the segment level exclusively is insufficient to handle occlusions. We will give more insights on this problem in the next section.

4.3 The role of occlusions

Treatment of occlusions is probably the most difficult part in stereo matching (which is also the reason why algorithms often ignore their existence). In region-based matching, a method that does not handle occlusions at all might indeed be partially successful on simple image pairs with only very small occlusions (see previous section). However, artefacts at depth boundaries will clearly become more pronounced with increasing size of occluded areas. We try to give an explanation for this in the following.

Let us consider the two views of a stereo pair illustrated in Figure 4.2a. The images show two segments S_1 and S_2 in the left image as well as the corresponding segments S'_1 and S'_2 in the right view. The segment S_2 is slightly displaced in the right image as indicated by the arrows. As a consequence of the displaced foreground object, there occur occlusions in both frames, which we colour red in Figure 4.2b. Note that S_1 and S'_1 are partially affected by occlusions. If we now match the complete segment S_1 of the left image in the right view using its correct disparity (zero in this case), this results in high matching costs in exactly those occluded regions. We mark those areas by diagonal hatches in the left view of Figure 4.2c. Since the stereo problem is symmetric, the same effect occurs if we match the complete segment S'_1

in the left view at disparity zero. We therefore also mark regions of high matching costs in the right view by diagonal hatches. Consequently, it is quite likely that S_1 and S_1' get erroneously assigned to a wrong disparity model that shows lower matching costs. Ignoring occlusions therefore does not only result in an incomplete problem formulation, but does as well show negative effects on the algorithm's results.

Unfortunately, occlusions cannot be dealt with in the domain of segments. For an explanation, let us again consider segment S_1 from Figure 4.2. When modelling the problem on the segment level only, we can simply state that the disparity of segment S_1 is zero. However, this is insufficient, since this would as well mean that the occluded parts of S_1 correctly match the second view at disparity zero. Therefore, the correct statement must be: S_1 has disparity zero (segment level), *but* contains a set of occluded pixels O_1 (pixel level). Consequently, occlusion detection requires the involvement of the pixel domain, which is also the basic idea behind the algorithm of Chapter 7.

4.4 Related segmentation-based algorithms

In this section, we give a brief overview on stereo techniques that employ image segmentation. Those methods that we see as closest related to the algorithms proposed in this thesis are discussed in more detail and differences to our techniques are pointed out.

Obviously, our algorithms are related to all those stereo methods that apply the segmentation assumption and some of which have already been mentioned in Chapter 3. In general, the principles of those techniques are quite different from each other, but they all share the advantages (and disadvantages) outlined in Section 4.2. From a viewpoint of the applied optimization scheme, the segmentation constraint is used in conjunction with progressive approaches [97], cooperative stereo [104], belief propagation [84], graph-cuts [27, 40] and special purpose optimization [88, 109]. Zitnick et al. [109] propose a region-based correspondence method for the task of novel viewpoint generation. The requirements for such an algorithm are clearly different, since the goal is not the computation of a disparity map that is as close as possible to the ground truth data, but the estimation of novel views that look natural. Zitnick et al. use colour segmentation to derive accurately outlined depth boundaries, which is of specific importance for this application. In a very recent work, Sun et al. [84] formulate the segmentation assumption as a soft constraint. More precisely, the authors bias the computed disparity solution towards consistency with the results of colour segmentation, but allow

deviations to increase the robustness against violations of the segmentation constraint.

We as well consider the algorithms of Birchfield and Tomasi [11] and Lin and Tomasi [57] as being similar to our methods in the sense that they formulate the correspondence problem in two steps. First, they estimate a set of layer models that correspond to different depth surfaces occurring in the scene. In the second step, they then assign each pixel to exactly one of those layers. Both algorithms are often also categorized as being segmentation-based, since the layer assignment step divides the image into regions of homogeneous disparity. However, they do not apply colour segmentation.

The closest related work to the algorithm presented in Chapter 5 is that of Tao and Sawhney [87]. The authors propose a region-based stereo algorithm that uses image warping to measure the quality of a disparity map. Each segment's disparity is modelled by a planar equation. Plane models are then propagated among neighbouring segments in order to generate a warped view that is as similar as possible to the real second view. In our work, we share the ideas of image warping for measuring the quality of a depth solution and hypothesizing depth from neighbouring segments. Most obvious differences in comparison to our method include our layered representation as well as our cost function that accounts for occlusions and aims at computing a smooth disparity map.

Among prior work, the closest related to the approach of Chapter 7 is the stereo method presented by Hong and Chen [40]. Similar to our technique, they combine region-based matching with graph-based optimization. They heuristically identify occlusions in a preprocessing step, which then allows them to optimize a very simple energy function. However, the results of their algorithm obviously depend on the success of this preprocessing step and it is not clear how well an apriori identification of occlusions can work, especially in the presence of large motion. In contrast to this, our cost function "knows" about the existence of occlusions. Disparities and occlusions are computed simultaneously, which we believe results in a more accurate reconstruction of both. Recently, our idea of developing a cost function that operates on a segment and a pixel level was extended by Deng et al. [27]. In contrast to our work, they segment both images, which allows them to split a segment into patches that are potentially occluded. The cost function no longer operates on a segment and a pixel level, but on a segment and a patch level. However, although their algorithm is faster than ours, it is not clear whether segmenting both images leads to a higher dependency on the segmentation assumption.

4.5 Summary

In this chapter, we have introduced the segmentation assumption, which builds the core for those algorithms presented in the subsequent chapters. Segmentation-based stereo relies on monocular cues that are present when only one image is observed. Approaches of this category divide the image into a set of homogeneously coloured regions. Within those regions, the disparity is expected to vary smoothly, while depth discontinuities are assumed to coincide with the regions' boundaries. Both assumptions are quite reasonable for natural images. We have described the advantages and disadvantages of using a segmentation-based approach. Major advantages are roughly summarized as: the handling of poorly textured areas, the accurate reconstruction of depth discontinuities and the increased robustness in areas close to occlusions. However, we have also pointed out that since this is the nature of an assumption, the segmentation constraint is not guaranteed to hold true. We have then focused on the role of occlusions in region-based matching. Although segmentation-based approaches are, in general, more robust against occlusions, it is, nevertheless, essential to take occlusions into account. Finally, we have presented region-based stereo algorithms that we consider as closest related to the proposed algorithms of this thesis.

Chapter 5

A greedy method using image warping

5.1 Introduction

This chapter describes the first segmentation-based stereo algorithm that is presented in this thesis. The algorithm was developed to overcome the inherent problems in stereo matching, which are: (1) untextured region, (2) the accurate reconstruction of disparity discontinuities and (3) the proper treatment of occlusions in both views. While the first two problems are, to a large extent, tackled due to the segmentation-based nature of the algorithm, occlusions are detected and handled by a novel mechanism that relies on image warping. Although image warping for stereo computation was originally proposed by Tao and Sawhney [87], they did not use it for treatment of occlusions. In this work, we choose a planar model to represent each segment's disparity. This model seems to be rather simple, but it works well enough in our experiments, even for more complex surfaces. The algorithm basically consists of two major steps. The first (layer extraction) step refers to the question: What are the dominant depth planes that occur in the scene? We try to give an answer by clustering a set of initial disparity segments. The second question (layer assignment step) then is: Which part of the image is covered by which dominant depth plane and where do occlusions occur? To find an answer to this problem, we formulate a global cost function that evaluates a disparity solution by image warping and detects occlusions simultaneously. A local minimum of costs is estimated by a greedy search strategy. The major argument for using this algorithm is that it has moderate computational demands (when using the optimization scheme from appendix A.1), but produces results of good quality.

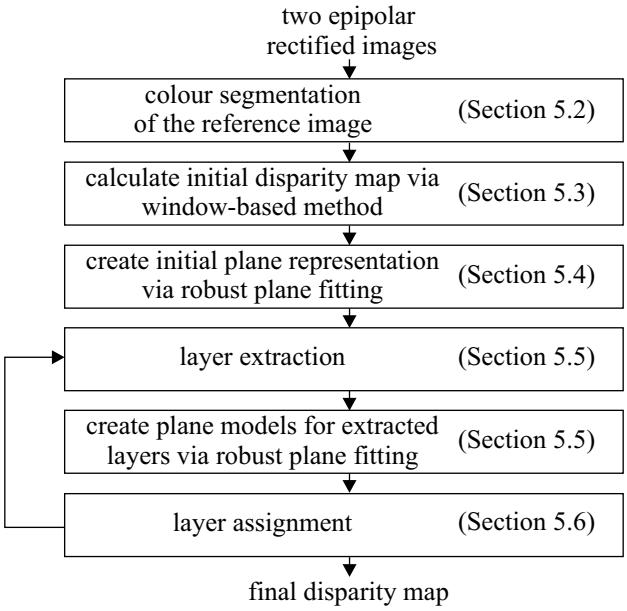
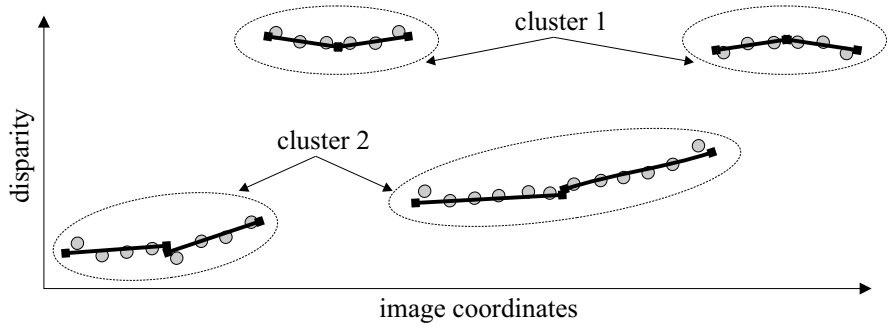


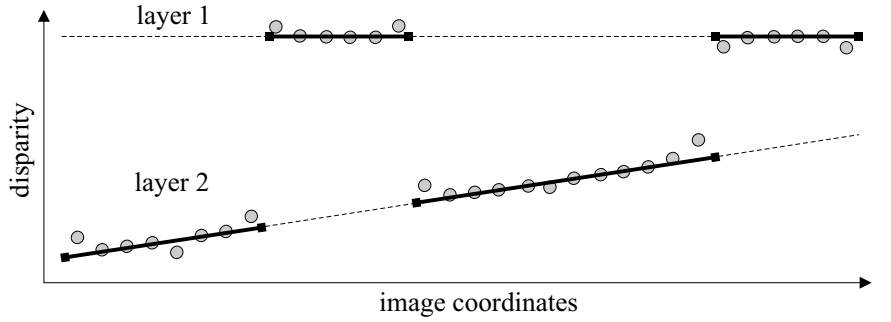
Figure 5.1: Overview of the algorithm.

The overall algorithm can be divided into several steps that are summarized in Figure 5.1. Input to our matching algorithm are two stereo images in epipolar geometry. The first processing step is the segmentation of the reference image into regions of homogeneous colour, as described in Section 5.2. Since discontinuities in the disparity map are usually reflected by discontinuities in the colour information, the borders of the segmented regions can be considered as a set of candidates for the boundaries of the disparity layers that we aim to compute. We calculate an initial disparity map using a window-based correlation technique. This process is explained in more detail in Section 5.3. In the next step, we create an initial plane representation for each extracted segment by robust fitting of a planar surface to the correlation-based disparity values inside each individual segment (Section 5.4).

The computed segments along with their plane description are the starting point for an iterative procedure in which segments are assigned to layers, which are groups of segments that can be approximated by one and the same planar equation. The iterative assignment starts with a *layer extraction* module (see Figure 5.1) based on mean-shift clustering, which we describe in Section 5.5. The advantage of the layered approach is illustrated in Figure 5.2. The planar models computed by fitting a plane to the disparity values derived from the initial disparity map are not very robust in small



(a)



(b)

Figure 5.2: Robustness of the layered representation. The less robust plane approximation of the individual segments illustrated in (a) is substituted by the more robust layer representation achieved by clustering as shown in (b).

segments as a consequence of the small spatial extent over which the plane was calculated. This is sketched in Figure 5.2a. A robust planar description of each layer is derived by fitting a plane over the larger region formed by all segments belonging to that particular layer, as shown in Figure 5.2b.

The last block in the iteration loop from Figure 5.1 is the *layer assignment* module, which we explain in Section 5.6. During this step, we try to improve the current solution based on a cost function that measures the quality of the current layer assignment. Based on the observation that erroneous assignments of segments to layers tend to arise more frequently along the layer borders rather than in their central regions, we test for each border segment whether a possible assignment to another layer might produce lower costs (i.e., a better solution). Based on the outcome of this hypothesis testing, new layers are formed by the layer extraction module during the next iteration step. The algorithm terminates if the costs could not be improved for a fixed number of iterations.

For the sake of clarity, we summarize the basic terms we use throughout this chapter. *Segments* are regions of homogeneous colour that are computed during the initial colour segmentation step. *Layers* are groups of segments (and therefore usually larger than individual segments) that can be approximated by one and the same planar equation. The *layer extraction* module computes new layers using mean-shift clustering. During the first iteration step, the individual segments are input to the clustering algorithm. In subsequent iteration steps, the clustering algorithm seeks to merge previously defined layers, which may have been modified in the layer assignment module, into larger layers. The *layer assignment* module tries to improve the current solution by assigning border segments to other neighbouring layers. During the iteration loop, the generated solution of lowest costs is recorded and returned as the final output of the algorithm.

5.2 Colour segmentation

We assume that for regions of homogeneous colour the disparity varies smoothly and depth discontinuities coincide with the boundaries of those regions. This segmentation assumption has been discussed in more detail in Chapter 4. It is incorporated into our approach by applying colour segmentation to the reference image and by using a planar model to represent the disparity inside the derived segments. In general, an oversegmentation of the image is preferable to ensure that this assumption is met. In principle, any algorithm that divides the reference image into regions of homogeneous colour can be used for the proposed stereo algorithm. Our current implementation uses a mean-shift-based segmentation algorithm that incorporates edge information as proposed by Christoudias et al. [23]. The resulting colour segmentation for a well-known stereo pair from the University of Tsukuba is shown in Figure 5.3. Pixels belonging to the same segment are assigned the same colour. To derive the desired plane models, we first compute an initial disparity map and use the computed disparity values to fit the plane for each segment.

5.3 Initial disparity map

We compute an initial disparity map using a local window-based method that exploits the results of the image segmentation and operates on different window sizes. We benefit from the image segmentation by exploiting the assumption of smoothly varying disparities inside a segment as introduced previously. Operating on different window sizes allows us to combine the

(a) (b)

Figure 5.3: Colour segmentation. (a) Left image. (b) Computed colour segmentation.

advantages of both small and large windows. The decision of which window size to use for which region is driven by the data.

Initially, we start with a small 3×3 window. The window centered on a pixel in the left image is shifted along the corresponding scanline in the right view to find the displacement of minimum dissimilarity. To measure the dissimilarity of two pixels, we compute the summed absolute differences of their RGB-values. We compute the disparity space image (DSI - see Section 3.1.1) using an efficient incremental approach described by Mühlmann et al. [67]. The disparity of a pixel $d_{x,y}$ at coordinates (x, y) is then derived from the DSI by using

$$d_{x,y} = \underset{D_{min} \leq d \leq D_{max}}{\operatorname{argmin}} \quad DSI(x, y, d) \tag{5.1}$$

with D_{min} and D_{max} denoting the minimum and maximum allowed disparity. This local optimization strategy is not able to produce correct disparity estimates in untextured or occluded regions. We filter out unreliable pixels by applying left-right consistency checking. An established match is only valid, if the matched point in the right image points back to the pixel in the left view. This check eliminates pixels of low texture or affected by occlusion (see Section 3.1.1). We further reject points with insufficient support by removing connected regions of equal disparity smaller than a predefined threshold.

We then reduce the search scope for each segment depending on a measurement of the segment's confidence. A similar approach was taken by Zhang and Kambhamettu [104]. We follow their idea to measure the reliability of a segment's disparity information by the density of valid points. Segments having a ratio of valid points larger than 50% are labelled as being *reliable*. We search the points with minimum disparity d_{min_i} and maximum

disparity d_{max_i} inside the ith *reliable* segment

$$d_{min_i} = \min_{(x,y)\in V_i} d_{x,y} \qquad d_{max_i} = \max_{(x,y)\in V_i} d_{x,y} \tag{5.2}$$

with V_i being the set of all valid points of the corresponding segment. We then compute the best correlation score for all unassigned pixels in a reduced search range

$$d_{x,y} = \underset{d_{min_i}-t_{tolerance}\leq d\leq d_{max_i}+t_{tolerance}}{\operatorname{argmin}} DSI(x,y,d) \qquad \forall (x,y) \in U_i \tag{5.3}$$

with U_i denoting the set of the ith *reliable* segment's unassigned points and $t_{tolerance}$ representing a small value for tolerance. In our implementation, the threshold $t_{tolerance}$ is set to the fixed value of one pixel. The reduction in search range helps to capture points with little texture information, for which the correct match was overwhelmed by noise due to the larger search scope [104]. Furthermore, it allows to propagate reliable disparity information inside the segment. This procedure directly reflects the assumption of smoothly varying disparity inside segments. We further apply a left-right consistency check using the reduced search range and remove points with insufficient support.

More matches are then gathered by increasing the window size leaving the already found valid points unchanged. First, we use the full search range for segments with density of valid points $<$ 50%. We then determine again the reliability of each segment. For all *reliable* segments, we reduce the search scope. Finally, the window size is further increased and the process is repeated. Since we are starting with a small 3×3 window, our approach is able to capture thin structures and generates a detailed disparity map. Disparity information for less-textured regions is then obtained by the use of larger windows. We therefore combine advantages of both strategies. Figure 5.4 shows a block diagram of the described algorithm. The initial disparity map calculated for the Tsukuba image pair using a 3×3, 5×5 and 7×7 window is presented in Figure 5.5. Higher disparity values are encoded by bright values. Black points represent invalid pixels for which no disparity information is estimated. The calculated initial disparity map serves to obtain the planar model of a segment and does not represent the final result of our algorithm.

5.4 Planar model fitting

Once we have calculated the initial disparity map, we use it to derive the planar model of each segment. We represent a segment's disparity by a

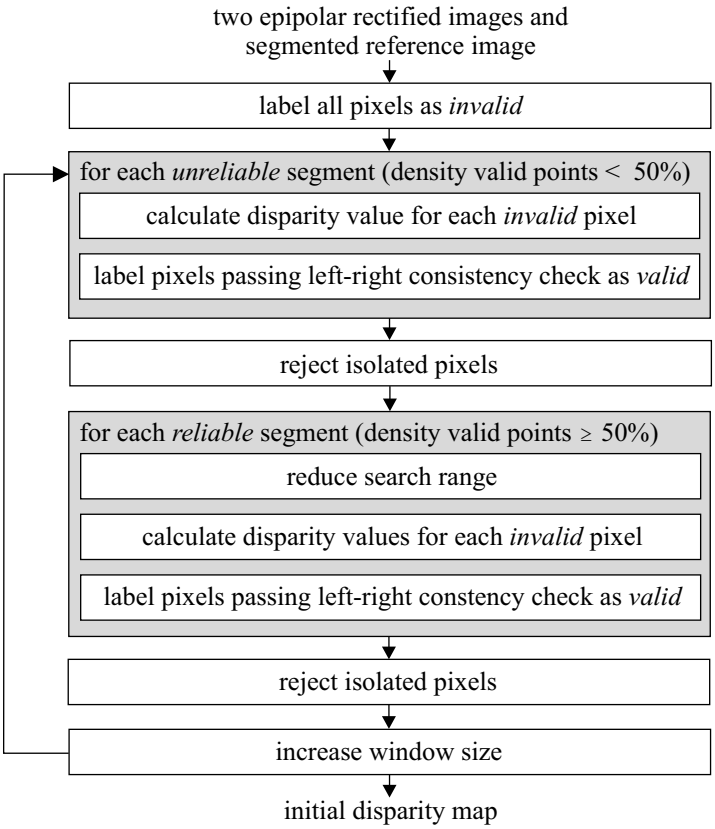
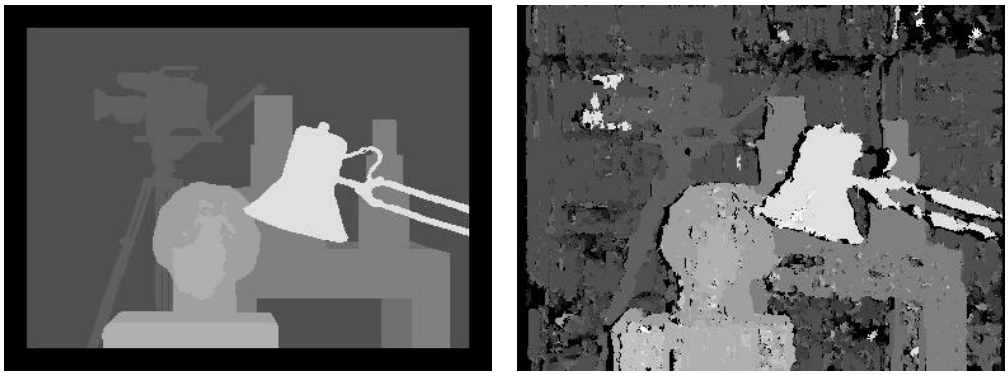


Figure 5.4: Block diagram of the algorithm creating the initial disparity map.



(a) (b)

Figure 5.5: Initial disparity map. (a) Ground truth provided with image pair. (b) Computed initial disparity map. Invalid points are coloured black.

function

$$d(x, y) = ax + by + c \tag{5.4}$$

with x and y being image coordinates and a, b and c being the plane parameters. To derive a segment's plane parameters, we apply least squares error fitting to all valid points inside the segment. The least squared error solution is then given by solving

$$\begin{bmatrix} \sum_{i=1}^{m} x_i^2 & \sum_{i=1}^{m} x_i y_i & \sum_{i=1}^{m} x_i \\ \sum_{i=1}^{m} x_i y_i & \sum_{i=1}^{m} y_i^2 & \sum_{i=1}^{m} y_i \\ \sum_{i=1}^{m} x_i & \sum_{i=1}^{m} y_i & \sum_{i=1}^{m} 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m} x_i d_i \\ \sum_{i=1}^{m} y_i d_i \\ \sum_{i=1}^{m} d_i \end{bmatrix} \tag{5.5}$$

with m being the number of valid points inside the segment, x_i and y_i being the coordinates of the ith valid point and d_i its corresponding disparity value. Unfortunately, the method of least squared errors is sensitive to outliers. Although we already try to remove outliers in the computation of the initial disparity map, there may still be erroneous points due to edge fattening, repetitive patterns or noisy image data. Figure 5.6 illustrates the implemented plane fitting algorithm that is robust to outliers. To derive a segment's planar description, we fit a plane to all valid points of the initial disparity map inside the segment. This is shown in Figure 5.6a. Not every image coordinate is represented by a point in disparity space because of invalid points in the initial disparity map. There are three valid points of high disparity representing outliers that attract the computed plane. To eliminate outliers, we search all valid points of the segment that have a distance to the computed plane that is larger than the predefined threshold $t_{outlier}$ and reject them as shown in Figure 5.6b. Formally expressed, the new set of valid points V' is derived by

$$V' = \{(x_i, y_i) \in V | d_i - (ax_i + by_i + c) \leq t_{outlier}\} \tag{5.6}$$

with V being the set of all valid points inside the segment and $t_{outlier}$ being a threshold that is set to the constant value of one pixel for all our computations. A new plane is then fitted to the points in V' using equation (5.5). This process is then iterated until

$$(a' - a)^2 + (b' - b)^2 + (c' - c)^2 \leq t_{convergence} \tag{5.7}$$

with a', b' and c' being the parameters of the new plane, a, b and c being the parameters of the plane that was derived in the previous iteration and $t_{convergence}$ being a very small value (typically 10^{-6}). Figure 5.6c illustrates the plane derived after removal of three outliers.

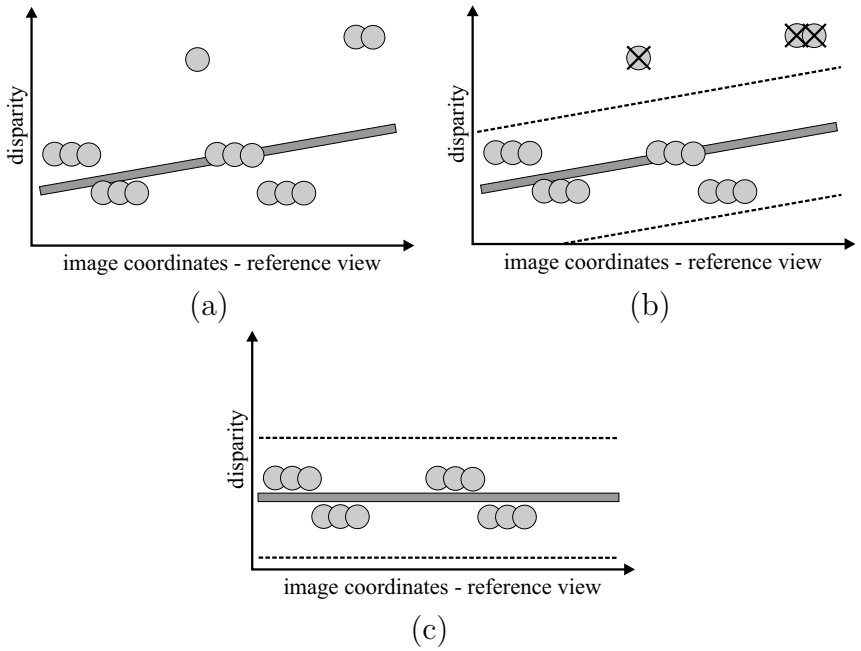


Figure 5.6: Robust plane fitting. (a) Initial computed plane. (b) Removal of outliers. (c) Final plane.

5.5 Layer extraction

One single surface that contains texture is usually divided into several segments by applying colour segmentation. However, for segments of the same surface the planar models should be very similar, as long as the surface can be well approximated as a plane. Following this idea, we define a measurement for the dissimilarity of two disparity planes and use this measurement in a clustering method to identify segments belonging to the same surface.

We exploit a distance measurement originally introduced by Tao et al. [88]. The similarity of two disparity segments is measured by calculating the intersection point of the normal vector on the first segment's plane, originating from that segment's center of gravity, with the disparity plane of the second segment. We then compute the length of the vector from the first segment's center of gravity to the point of intersection, which is denoted by dis_1. For symmetry, we also compute dis_2, which is the distance between the second segment's center and the first segment's disparity plane. The term $dis_1 + dis_2$ then describes the amount of dissimilarity between two planes. We illustrate this process in Figure 5.7 for the two-dimensional case. We believe that this measurement is specifically well suited to the task of clus-

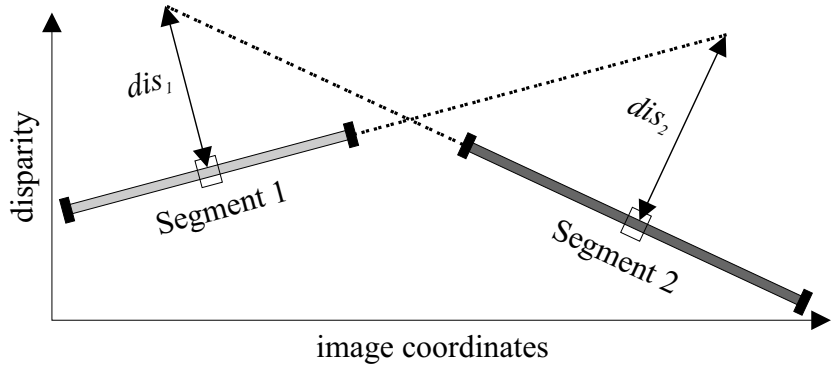


Figure 5.7: Measuring the dissimilarity of two disparity planes.

tering disparity planes, since it incorporates spatial information as well as the plane parameters.

We project each segment into a five-dimensional feature space, consisting of the three plane parameters and two spatial parameters represented by the x and y components of the center of gravity. We do not project the z component, since it can be deduced from the other five parameters. We employ the mean-shift algorithm [24], which we modify to embed the described plane dissimilarity measurement, to extract clusters in this feature space. A specific advantage of the mean-shift algorithm is that the number of clusters does not need to be known beforehand. To apply the mean-shift to a data point y_k at iteration k, we determine its neighbourhood $N(y_k)$ by

$$N(y_k) = \{x \in DP \mid dis(x, y_k) \leq r\} \tag{5.8}$$

with DP being the set of all data points, dis denoting the plane dissimilarity function and r being the radius of the mean-shift. We then compute the mean value of all data points inside the neighbourhood. Since data points represent segments covering areas of different sizes, the reference image is not uniformly sampled. A layer containing a rich amount of texture and therefore a large number of segments will be represented by more region samples than a layer representing large homogeneously coloured regions. The layer of homogeneous colour may not have enough samples to form a dense cluster in feature space. We overcome this problem by weighting each data point according to the area of the segment it describes. We then derive the location of the shifted data point y_{k+1} by computing the weighted mean value

$$y_{k+1} = \sum_{x \in N(y_k)} \frac{a_x}{A} x \tag{5.9}$$

with a_x being the number of pixels inside the segment described by the data point x and A being the summed up areas of all segments inside the neighbourhood $N(y_k)$. The mean-shift is then iteratively applied until the magnitude of the shift becomes smaller than the threshold $t_{convergence}$ set to a very small number (typically 10^{-6}). The data point is thereby shifted to a local density maximum. This procedure is applied for each data point.

In the fusion step, we then investigate the points of convergence to derive the points building a cluster. Points having a distance smaller than a threshold, set to $\frac{r}{2}$ in our implementation, are merged to form a single cluster. The distance is again computed by using the plane dissimilarity measurement defined above.

Members of the same cluster build a layer. For deriving a layer's plane equation, we use the initial disparity map. Robust plane fitting is applied to the valid points of all segments belonging to the layer.

Note that for the methods presented in this thesis, it is important that the layer extraction step identifies the complete set of disparity layers present in the scene. If the layer extraction procedure fails to capture a layer, those segments belonging to this particular disparity layer will be assigned to a wrong disparity model. It is therefore safer to generate a larger set of layers, since wrong disparity layers will most likely be eliminated in the layer assignment step of the algorithm anyway.

5.6 Layer assignment

We try to improve the current solution by optimizing the assignment of segments to layers. A cost function that uses image warping is designed to measure the quality of the current assignment. We describe an efficient hypothesis testing framework in order to optimize the specified cost function.

5.6.1 Cost function

We measure the quality of a disparity map by warping the reference view according to the current disparity map. The basic idea behind this procedure is that if the disparity map was correct, the warped image should be very similar to the real image from this viewpoint. We implemented a warping procedure based on a Z-buffer to explicitly model visibility. To obtain the second view, we warp each segment according to its current disparity plane. A naive approach would reconstruct the second view by projecting each individual pixel of the reference image into the second view using its disparity value. Consequently, pinholes would occur in the warped image for areas

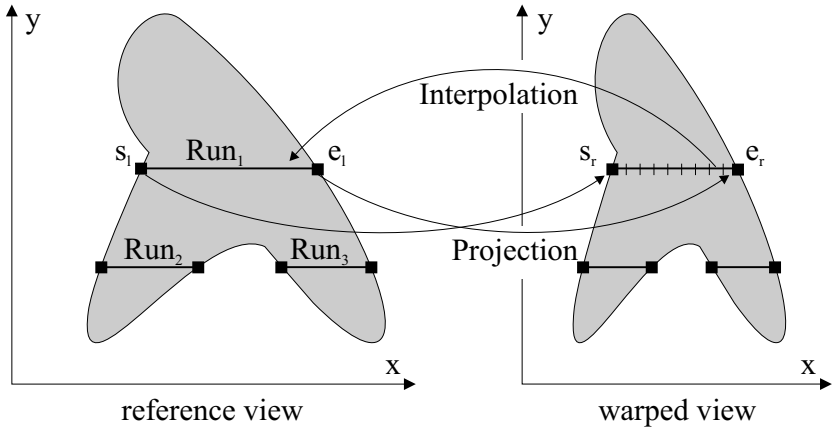


Figure 5.8: Warping a segment to the second view according to its disparity plane.

that are undersampled in the reference image. Therefore, a more elaborate strategy is used. A segment is represented by the set of all its horizontal scanline runs. A segment's scanline runs are derived by tracing each horizontal scanline from left to right. Whenever the left border of the segment is encountered, the corresponding coordinates are stored as the starting point of a run. Whenever the segment's right border is hit, the corresponding coordinates are stored as the ending point of the run. The warped view of a segment is generated by transforming all its scanline runs. Therefore, the coordinates of the starting and ending point in the warped view are computed using the segment's planar model. For all points between the warped starting and ending point, we compute the exact coordinates in the reference image according to the segment's disparity plane. The colour values for those pixels are then derived by linear interpolation of the colour values of the pixels left and right to the exact position in the reference view. This process is illustrated in Figure 5.8.

We reconstruct the second view by warping all segments of the reference view. In this procedure, pixels from the second view may receive a contribution from more than a single segment. For those pixels, we have to make a decision concerning visibility. We use a Z-buffer representing the second view, which naturally enforces visibility. Each Z-buffer cell corresponds to a single pixel of the right view. If a Z-buffer cell contains more than one pixel, only the pixel with the highest disparity is visible, since it is the one closest to the camera. The others are therefore occluded in the second view. Furthermore, we are also able to detect pixels occluded in the reference image,

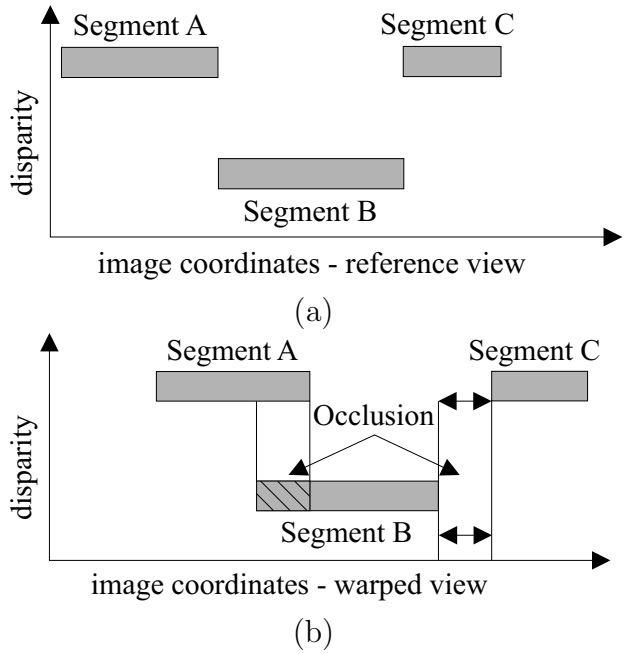


Figure 5.9: The warping operation. (a) Segments and corresponding disparity in the reference view. (b) Segments warped to the second view according to their disparity planes.

since they correspond to empty Z-buffer cells. We illustrate this in Figure 5.9.

We will now use the observations made above to formulate a cost function which is designed to measure the quality of a derived disparity map. The first term of the cost function is based on the idea that a good disparity map should produce a warped view with a high similarity to the real second image. Translated to our cost function, we calculate the colour dissimilarity between the warped and real views for all pixels visible in the warped image. According to the literature, we refer to this term as data term that is defined by

$$T_{data} = \sum_{p \in Vis} dis(W(p), R(p)) \tag{5.10}$$

with $W(p)$ denoting the pixel p in the warped image and $R(p)$ being the pixel p in the real second view. The set of visible pixels Vis is defined by the union of all pixels that have the highest disparity in their individual Z-buffer cells. Formally, the set Vis is computed by

$$Vis = \{\cup_{x,y}\, p \in Z_{x,y} \mid \forall q \in Z_{x,y} : d(p) > d(q) \vee p = q\} \tag{5.11}$$

with $Z_{x,y}$ denoting the set of all pixels inside the Z-buffer cell at image coordinates x and y and $d(p)$ being the disparity of pixel p. The colour dissimilarity function $dis(p_i, p_j)$ is defined as the summed up absolute differences of RGB values of pixels p_i and p_j. We write

$$dis(p_i, p_j) = |r(p_i) - r(p_j)| + |g(p_i) - g(p_j)| + |b(p_i) - b(p_j)| \tag{5.12}$$

with $r(p)$ being the red, $g(p)$ being the green and $b(p)$ being the blue colour components of pixel p.

The second term of the cost function accounts for occlusions. It is necessary for our cost function to penalize occlusions, since otherwise declaring all pixels as occluded would form a trivial optimum. We therefore introduce an occlusion term that penalizes occlusions in the left and right images. This term is defined by

$$T_{occlusion} = (|Occ_R| + |Occ_L|) \cdot \lambda_{occ} \tag{5.13}$$

with Occ_R being pixels that are occluded in the right view, Occ_L denoting occlusions in the left image and λ_{occ} being a constant penalty for occlusion. The set Occ_R is defined by the union of all pixels that are occluded by a pixel of higher disparity in their individual Z-buffer cells. This set is computed by

$$Occ_R = \{\cup_{x,y}\ p \in Z_{x,y} \mid \exists q \in Z_{x,y} : d(p) < d(q)\}\,. \tag{5.14}$$

The set of occlusions in the left image Occ_L is then defined by the union of all empty Z-buffer cells given by

$$Occ_L = \{\cup_{x,y}\ Z_{x,y} \mid Z_{x,y} = \emptyset\}\,. \tag{5.15}$$

The last term of the cost function motivates smoothness across segments. We introduce a discontinuity penalty that is applied when two neighbouring pixels (in 4-connectivity) are assigned to different layers in the reference image. We define this term by

$$T_{smoothness} = \sum_{(p_i,p_j)\in N} \begin{cases} \lambda_{disc} & : \ layerid(p_i) \neq layerid(p_j) \\ 0 & : \ \text{otherwise} \end{cases} \tag{5.16}$$

with $layerid(p)$ being a function that returns the id of the disparity layer to which the segment containing pixel p is assigned and λ_{disc} being a constant penalty for discontinuity. The set N defined for the left image I_L denotes pairs of pixels (p_i, p_j) with $p_i, p_j \in I_L$ and $i < j$ that are neighbours in 4-connectivity.

Putting this together we finally obtain the cost function

$$C = T_{data} + T_{occlusion} + T_{smoothness} \tag{5.17}$$

measuring the quality of a disparity map. We are therefore searching an assignment of layers to segments that minimizes C.

5.6.2 Optimization

Unfortunately, finding the assignment that minimizes C is non-trivial. Given S segments and L distinct layers there are S^L different possible assignments. The large solution space indicates the complexity of the problem. Moreover, finding the layer assignment with minimum value for C is shown to be $\mathcal{NP}$-complete and therefore not solvable by a complete algorithm in finite time. In our approach, we employ an efficient greedy search strategy to find a local optimal solution. This search strategy is similar to that used by Tao and Sawhney [87], although they do not optimize an explicit cost function, but always take the local optimal decision that minimizes colour dissimilarity between the real and warped views until convergence.

The basic idea behind the algorithm is to propagate correct disparity information from neighbouring segments. Segments can be assigned to planes giving poor disparity estimates, since a segment can be affected by occlusion or may not have enough texture information to allow correct disparity estimation. Nevertheless, the chances for a neighbouring segment to be assigned to the correct disparity model are high, since usually disparity varies smoothly, except at depth boundaries. We exploit this idea in a hypothesis testing framework. For a segment, we hypothesize that its current layer assignment is wrong and a layer of a neighbouring segment better describes the segment. To test this hypothesis, we replace the plane model of the current segment by the neighbouring layer's plane equation. We then warp the reference image to the second view according to the current layer assignment and evaluate the cost function. If the costs are improved, the hypothesis is accepted and rejected otherwise. For a segment, we test the hypotheses of all neighbouring layers as shown in Figure 5.10. In this figure the segment S_1 has five neighbouring segments assigned to layers 1, 2 and 3, which we refer to as the neighbouring layers of S_1. We avoid testing layer 1 on S_1, since this is the current assignment. The layer hypotheses of layers 2 and 3 need to be checked. Although there may be a large number of neighbouring segments, the layer neighbourhood is usually very small. Consequently, the number of layer hypotheses that need to be checked is small. If a segment is surrounded only by segments assigned to the same layer as the segment, which is usually the case for the majority of segments, no tests need to be applied at all. These observations represent major arguments for the algorithm's efficiency.

We embed the ideas of hypothesizing neighbouring layers into a greedy algorithm as follows. In the initial solution, we use the layer assignment derived from the layer extraction step. For each segment, we test the neighbouring layer hypotheses as described above. In the testing phase, the assignment of all other segments remains fixed. If there are layers generating smaller costs

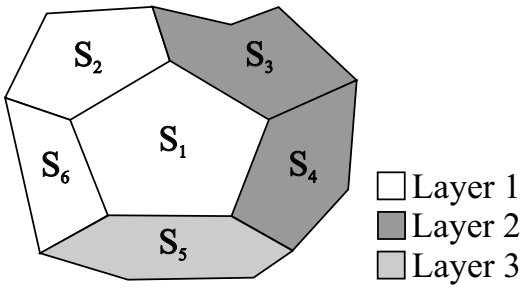


Figure 5.10: Hypothesis testing.

than the current solution, we record the one giving the largest improvement. Otherwise, the current assignment was found to be the best, and we record this assignment. After all segments have been tested, every segment gets assigned to its recorded layer. This process is then iterated and terminates if there has not been an improvement of costs for a fixed number of iterations. The generated solution with lowest costs is returned. Keeping the segments fixed during the hypothesis testing stage and updating them after all segments have been checked makes the algorithm independent of the order of applied operations. The greedy nature of the algorithm is reflected by always picking the layer hypothesis that locally gives the highest improvement of costs. An aspect concerning the computational efficiency of the proposed algorithm is that we only need to test segments if their neighbourhood has changed in the previous iteration. Otherwise, we would unnecessarily repeat the tests from the previous iteration without getting new results. Furthermore, since only small parts of the warped view are changed in the hypothesis testing, it would not be efficient to always warp the whole image. We therefore employ an incremental image warping procedure described in appendix A.1. The block diagram of the greedy algorithm is shown in Figure 5.11.

5.7 Experimental results

To test the performance of our method, we use the test bed proposed by Scharstein and Szeliski [78], which has already been discussed in Section 3.2 of this thesis. We therefore applied the algorithm on the four image pairs from the Middlebury Stereo Vision website (see Figure 3.7) and submitted the results to the online version of the test bed. All disparity maps were thereby created using constant parameter settings. The derived disparities are then compared against the ground truth data by computing the percentage of

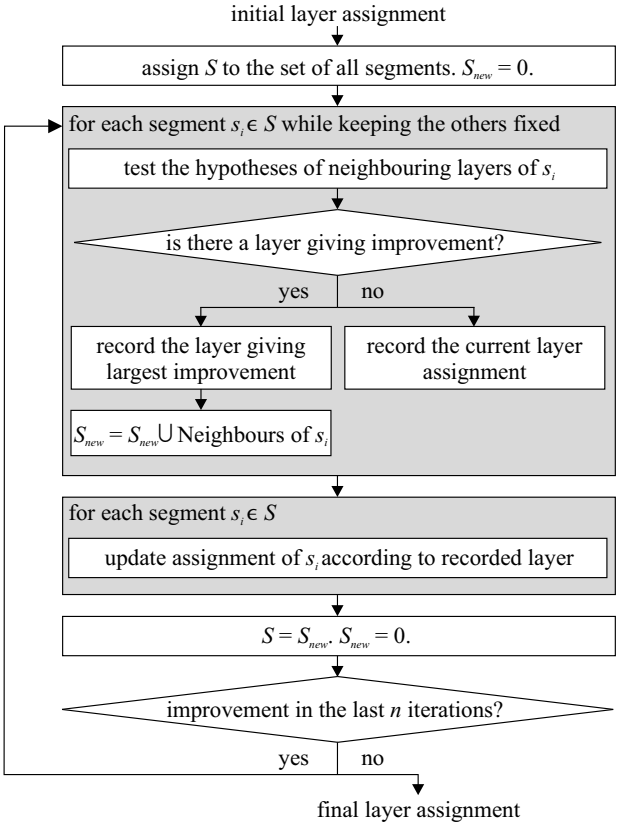


Figure 5.11: Block diagram of the greedy algorithm.

unoccluded pixels whose absolute disparity error is greater than one. At the time of publication of the corresponding journal paper, our method was ranked as having the second best overall performance among 30 different stereo algorithms tabulated. For a more recent ranking, the reader is referred to Table 3.1, where the proposed method is denoted as "*Segm.+glob.vis.*". In the following, we show and discuss in more detail the results obtained for two of the four test sets. For the examples shown in this section, disparity maps are created using individual parameter values. We discuss the sensitivity of results to different settings of the parameters in appendix A.2. For additional results as well as for results achieved with constant parameter values, the reader is referred to the Middlebury Stereo Vision website[1]. Furthermore, we present disparity maps for a more complex scene that was taken from [79] and for a self-recorded stereo pair.

The first image pair we ran our algorithm on is the "head and lamp"

[1]http://www.middlebury.edu/stereo/

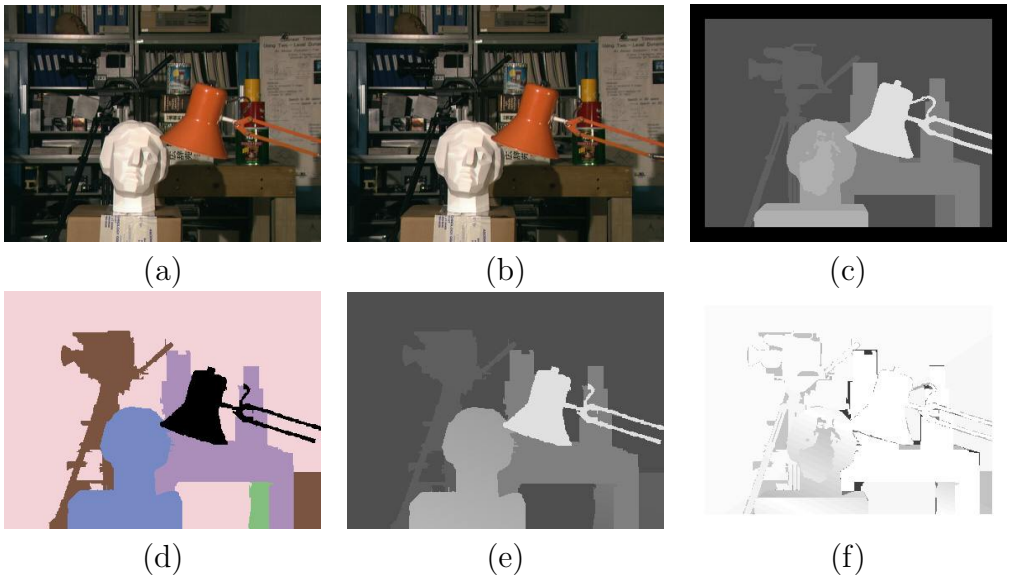
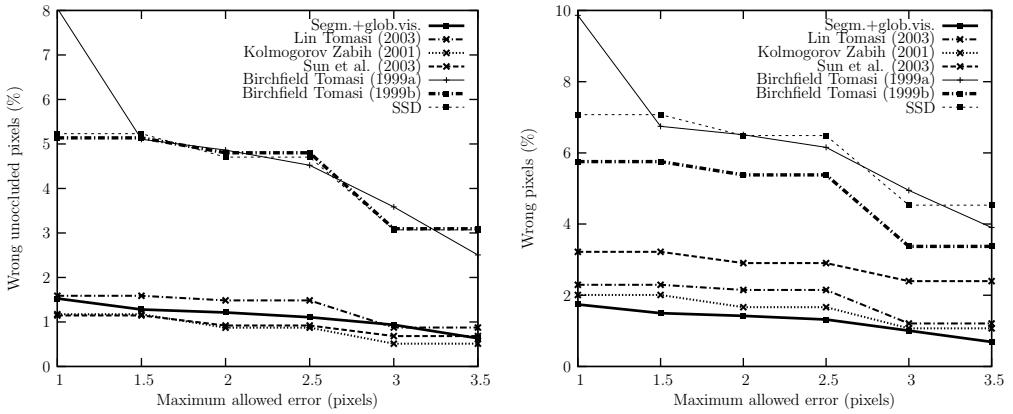


Figure 5.12: Results for the Tsukuba test set. (a) Left image. (b) Right image. (c) Ground truth provided with image pair in the geometry of the left image. The presented disparity maps are scaled by a factor of 16 for visualization. (d) Computed layers. (e) Computed disparity map. (f) Absolute errors scaled by a factor of 64.

data set of the University of Tsukuba, which became a standard test set for the stereo community. The image pair is presented in Figures 5.12a and 5.12b. It shows a rather complex scene containing untextured regions (e.g., table) and thin objects (e.g., lamp arm), which make it hard for a stereo algorithm to capture the correct disparity information. The hand-labelled ground truth for the left image is shown in Figure 5.12c. The presented disparity maps are scaled by a factor of 16 for visualization, i.e. a disparity value of one pixel is mapped to the gray value 16. We present the layers that were computed by our algorithm in Figure 5.12d. Pixels belonging to the same layer are assigned to the same colour in the figure. Our algorithm divides the reference image into six layers. Although we do not aim for a semantic segmentation, the derived layers correspond well to objects of the real world (head, lamp, camera, table, leg of the table, and background). The computed disparity map is then presented in Figure 5.12e. We used the following parameter settings: $r = 0.8$, $\lambda_{occ} = 20.0$ and $\lambda_{disc} = 20.0$. To visualize the quality of the derived disparity map we compare it against the ground truth in Figure 5.12f. We thereby show the absolute error with darker pixels representing higher deviations from the ground truth. White



(a) (b)

Figure 5.13: Quantitative results for the Tsukuba test set. The percentage of wrong pixels for different disparity error thresholds is presented for two error metrics. (a) Only *unoccluded* pixels are considered. (b) *All* pixels are considered.

pixels indicate a perfect correspondence between the computed disparity map and the ground truth. We applied a scaling factor of 64 to the computed errors and inverted the image for better visibility. Apart from some errors occurring at depth borders, which are mainly caused by colour segments that overlap depth boundaries, small errors appear on the head where the planar representation oversimplifies the real surface. To get a more accurate result for this region of the image, it would be advantageous to set the mean-shift radius r to a lower value. The head would then be reconstructed by a larger number of layers. However, this would also lead to a less robust reconstruction of the background, which would then be represented by more than one layer.

In Figure 5.13, we compare the computed disparity map against the results generated by some of the best-performing stereo algorithms tabulated on the Middlebury Stereo Vision website. For comparison, we use two different error metrics. The first one computes the percentage of wrong *unoccluded* pixels exceeding a specified disparity error threshold. This corresponds to the metric used in [78] when the threshold is set to one. For the second error metric, we do not exclude occluded pixels from the evaluation and compute the percentage of *all* erroneous pixels. For both error measurements, we only consider pixels for which ground truth is available and plot the resulting error percentages for different settings of the maximum allowed disparity error. For comparison, we choose six different stereo algorithms as represen-

tatives of several different matching strategies. The disparity maps generated by these algorithms are obtained from the Middlebury Stereo Vision website. Apart from the proposed algorithm, which is referred to as "*Segm.+glob.vis.*" in the figure, we present results from two layered methods (Birchfield and Tomasi (1999a) [11], Lin and Tomasi (2003) [57]), the graph-based method described by Kolmogorov and Zabih (2002) [50], the belief propagation algorithm of Sun et al. (2003) [85], the dynamic programming-based algorithm of Birchfield and Tomasi (1999b) [10] and an implementation of sum-of-squared-differences (SSD) by Scharstein and Szeliski [78] that uses shiftable windows. Concerning the first error metric, which is used in Figure 5.13a, it is evident that our method is able to compete with the best performing algorithms with only the graph-based approach of Kolmogorov and Zabih [50] and the belief propagation algorithm of Sun et al. [85] giving better results. In Figure 5.13b, we present the results using the second error metric that includes occluded pixels. For this error measurement, our method outperforms the others, which proves its capability to generate meaningful results in occluded regions and precisely locate depth boundaries.

As a second test set we present the Venus image pair shown in Figure 5.14a and Figure 5.14b. The corresponding ground truth in the geometry of the left image is presented in Figure 5.14c. The Venus data set consists only of planar surfaces. Although the scene structure is quite simple, there are large untextured regions that make the reconstruction difficult. Our algorithm extracts four layers as shown in Figure 5.14d. The corresponding disparity map is then presented in Figure 5.14e. The parameters were set to the following values: $r = 0.6$, $\lambda_{occ} = 15.0$ and $\lambda_{disc} = 30.0$. Since the newspaper at the right of the image consists of two planes that are joined by a crease edge, the algorithm oversimplifies this surface. Nevertheless, the resulting disparity error shown in Figure 5.14f is negligibly small. Our algorithm almost perfectly reconstructs the scene with the disparity planes correctly outlined. The quantitative results in Figure 5.15 show that for this pair our method clearly outperforms the other algorithms for both error metrics.

We further evaluated the proposed algorithm on a more complex scene using the Teddy test set taken from Scharstein and Szeliski [79]. A large disparity range, more complex scene geometry and textureless areas make the image pair challenging for stereo algorithms. Scharstein and Szeliski are planning to add this test set to their benchmark, since they argue that current test images are getting too simple to discriminate among the best-performing stereo algorithms. The Teddy test set is shown in Figure 5.16a and Figure 5.16b. The scene consists of a large number of surfaces for which some can be well approximated as a plane (background, floor, roof and walls of the house),

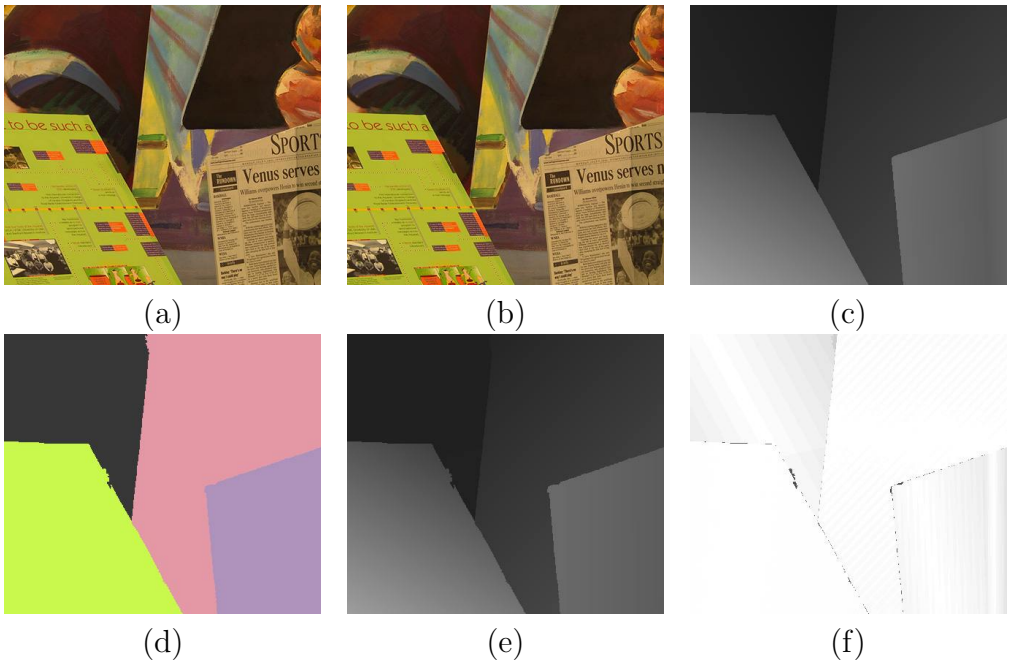


Figure 5.14: Results for the Venus test set. (a) Left image. (b) Right image. (c) Ground truth provided with image pair in the geometry of the left image. The disparity maps are scaled by a factor of 8. (d) Computed layers. (e) Computed disparity map. (f) Absolute errors scaled by a factor of 32.

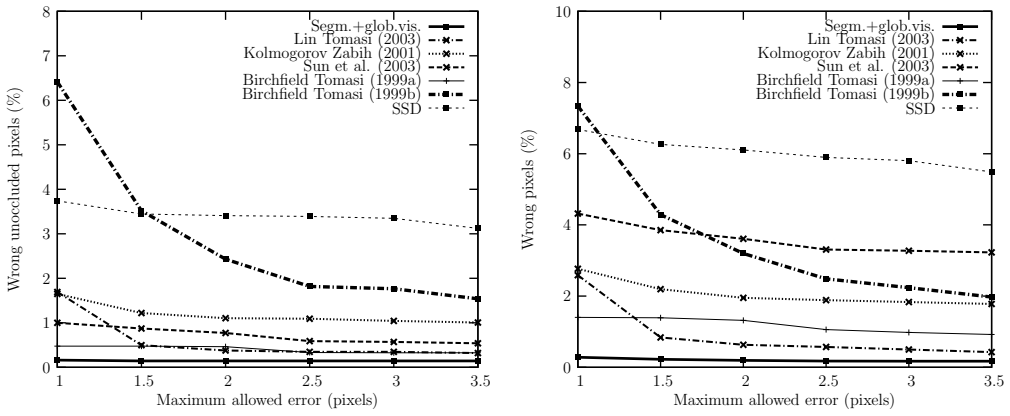


Figure 5.15: Quantitative results for the Venus test set. The percentage of wrong pixels for different disparity error thresholds is presented for two error metrics. (a) Only *unoccluded* pixels are considered. (b) *All* pixels are considered.

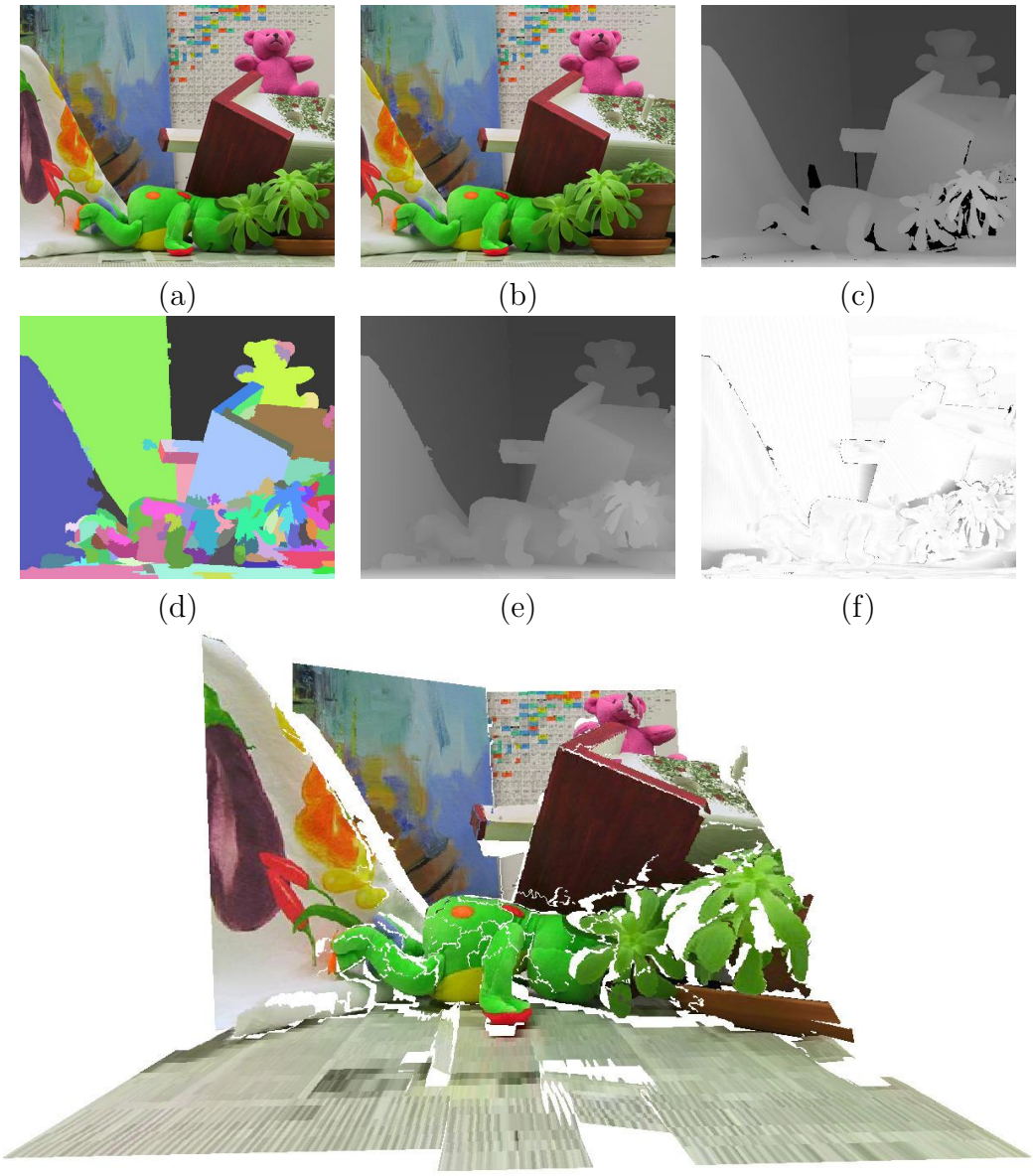


Figure 5.16: Results for the Teddy test set. (a) Left image. (b) Right image. (c) Ground truth provided with image pair. The disparity maps are scaled by a factor of 4. (d) Computed layers. (e) Computed disparity map. (f) Absolute errors scaled by a factor of 16. (g) Reconstructed view of the Teddy test set.

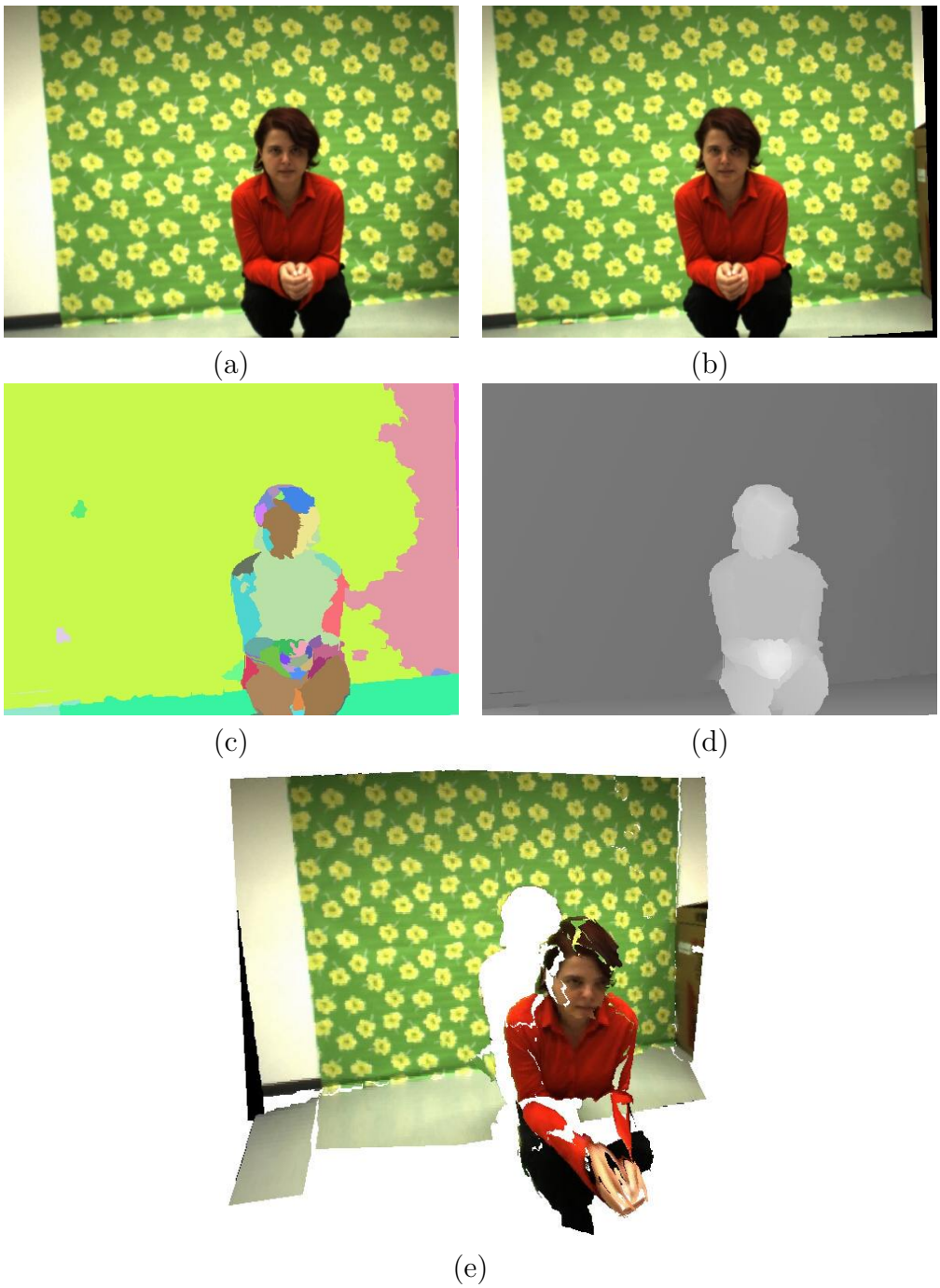
Test set	Tsukuba	Venus	Teddy
Mean signed error (pixels)	-0.04	0.05	0.04
Root mean-square error (pixels)	0.73	0.31	1.07
Maximum error (pixels)	9.13	6.75	19.00

Table 5.1: Error statistics computed from comparison against the ground truth.

whereas others have a more complex surface structure (teddies, plants). The ground truth for the left image of the Teddy scene is presented in Figure 5.16c. Pixels for which the method of Scharstein and Szeliski [79] fails to produce the ground truth are coloured black. In the final configuration of the algorithm, the scene is represented by a set of 77 planes which we show in Figure 5.16d. Surfaces that can be well approximated as a plane are thereby represented by a single or a small number of layers resulting in a robust reconstruction. However, for more complex shapes like the green teddy the surface is reconstructed by a larger number of layers providing a detailed description of the corresponding surface structure. The computed disparity map is then presented in Figure 5.16e. We used the following parameter settings: $r = 0.6$, $\lambda_{occ} = 20.0$ and $\lambda_{disc} = 2.5$. To illustrate the quality of the derived matching results, we compare it against the ground truth in Figure 5.16f. The percentage of pixels exceeding a disparity error of one including occluded regions is 6.55. If occluded regions are not considered, we get an error of 5.00%. 19.54% of pixels in occluded regions exceed the error threshold of one. Since we do not have the results for the other methods that we used for comparison for the previous two test sets, we present a reconstructed view of the scene in Figure 5.16g to give a further impression of the accuracy and detail of the computed disparity information.

In addition, we computed an error statistic for the three discussed image pairs for which ground truth is available. As opposed to the errors shown in Figure 5.13 and Figure 5.15, the error values listed in Table 5.1 also include errors smaller than one pixel.

Finally, we applied our method to a stereo pair that we recorded using two Dragonfly IEEE-1394 colour cameras as provided by Point Grey Research. We calibrated the cameras using the method described in [105] and transformed the images into epipolar geometry. The recorded stereo pair presented in Figure 5.17a and 5.17b shows a person crouching in front of a wall. The scene contains untextured regions like sections of the white wall and the floor, and complex surface structure in the form of the person. We present the disparity map that was computed using the parameter settings



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

Figure 5.17: Results for a self-recorded stereo pair. (a) Left image. (b) Right image. (c) Computed layers. (d) Computed disparity map in the geometry of the left image scaled by a factor of 4. (e) Reconstructed view.

$r = 0.75$, $\lambda_{occ} = 30.0$ and $\lambda_{disc} = 10.0$ in Figure 5.17d. Visual inspection of the disparity map indicates that the complex shape of the person's outline is correctly recovered. Furthermore, the algorithm seems to capture relatively well the person's disparity, which is better visible in the reconstructed view we present in 5.17e. The background is represented to a large extent by a single layer containing the left part of the white wall and most of the area covered by the wallpaper, which results in a robust reconstruction despite the poor texture of the white wall. In addition, regions belonging to the large occlusion left to the person's outline are correctly assigned to this background layer. A less accurate reconstruction is obtained for the floor, which we found was not only caused by its poor texture, but also by reflections of the wallpaper pattern on this surface.

We implemented the proposed method in C++ and ran our algorithm on an Intel Pentium 4 2.0 GHz computer. For the 384×288 pixel Tsukuba and the 434×383 pixel Venus test set, the algorithm needed approximately 20 seconds until termination. For the 450×375 pixel Teddy and the 640×480 pixel self-recorded image pairs, the computational effort increased to 100 and 180 seconds, respectively. The longer running times are not only caused by the larger image sizes, but also by the more complex scene structures that require more layers to represent the scene. Therefore, the number of hypothesis tests that need to be performed is increased.

5.8 Summary

In this chapter, we have proposed a new stereo matching method that takes advantage of colour segmentation and uses planar layers to describe the scene. The algorithm is able to generate correct disparity information in untextured areas and regions close to depth boundaries, which is a challenging task in stereo matching. Our method alternates between a layer extraction and an assignment step. Layers are extracted by a robust mean-shift-based clustering algorithm. The clustering method takes advantage of a plane dissimilarity measurement that incorporates spatial information as well as plane parameters. The planar model of each layer is then computed based on the layer's spatial extent. The assignment of segments to layers is made in a hypothesis testing framework. Disparity information is thereby propagated across segments. Hypotheses are accepted if they improve a global cost function. For evaluating the costs of an assignment, the reference image is warped to the second view according to the disparity map. The cost function evaluates the pixel dissimilarity between the real and warped images and penalizes occlusions in both views and discontinuities between segments. Layer extraction

and assignment are then iterated to find the generated disparity map with lowest costs.

We have demonstrated the performance of the proposed algorithm using the test bed of Scharstein and Szeliski [78]. Qualitative and quantitative evaluation proved the good quality of the achieved matching results. At the time of publication of the corresponding journal paper, the proposed method achieved second place out of 30 different stereo algorithms in the online evaluation on the Middlebury Stereo Vision website and is currently ranked on the fifth position of approximately 40 contributions (see Table 3.1). We found that the proposed technique can provide occluded regions with more accurate disparity estimates than a set of Middlebury reference algorithms that we used for comparison. Furthermore, we applied our method to a more complex image pair taken from [79] and to self-recorded data. In the absence of reference data, we presented 3D visualizations of the reconstructed scene to demonstrate the good quality of the computed disparity layers.

Chapter 6

Extending the greedy method to motion

6.1 Introduction

The good performance of the stereo algorithm described in the previous chapter encouraged us to extend our method to motion computation. In contrast to the stereo problem, an optical flow algorithm must not rely on the validity of the epipolar constraint, but also account for motion in y-direction. Since this eliminates one of the most powerful constraints, the optical flow problem can be considered as more difficult than stereo. However, the complicating factors remain the same and are roughly summarized as: (1) the treatment of untextured regions and (2) the accurate reconstruction of flow values close to motion boundaries (as a consequence of the occlusion problem). Those common problems of stereo and motion give good indication that our stereo algorithm could also be well-suited for the optical flow problem.

The presented motion algorithm is an extension of the stereo method proposed in Chapter 5. We minimize unnecessary overlaps with the previous chapter by focusing on a description of the modifications that are applied in order to eliminate the epipolar constraint. To identify those steps that need to be adapted, the algorithmic framework is illustrated in Figure 6.1. Input to our algorithm are two consecutive views of a video sequence. This means that, in contrast to the stereo algorithm, the epipolar assumption is, in general, not valid for those image pairs. In the first step, colour segmentation is applied to the reference image. To describe the motion inside the segment, the affine motion model is chosen. The model of each segment is then initialized from a set of sparse initial correspondences. Affine motion, the segmentation assumption in the context of two-dimensional motion as

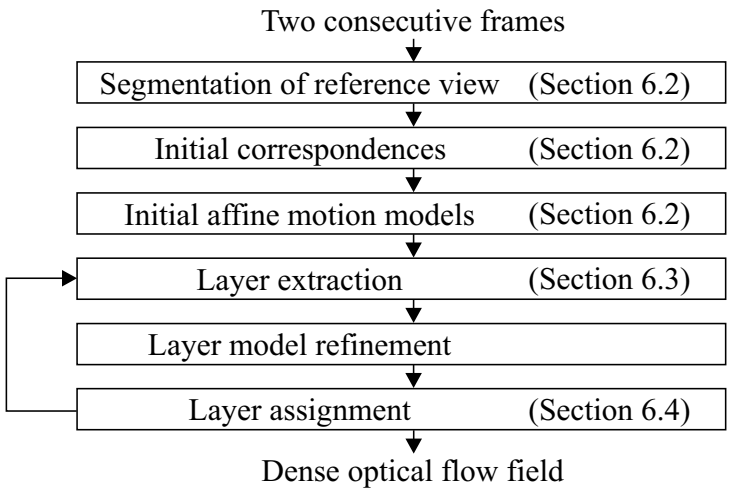


Figure 6.1: Algorithmic outline.

well as the computation of initial correspondences and motion models are described in Section 6.2. Those models that represent the dominant image motion are then extracted in the layer assignment step by clustering of the initial motion models. The clustering step is almost identical to that presented in Section 5.5 of the previous chapter, but requires a measurement for the dissimilarity of two affine motion segments, which is given in Section 6.3. In the next step, the model of each layer is refined based on its spatial extent. Finally, each segment is assigned to a single dominant motion model in the layer assignment step of the algorithm. Modifications that are carried out in this step include a new strategy for warping an individual segment, as is explained in Section 6.4.1. Moreover, since Z-buffering cannot be applied for the presented motion method, Section 6.4.2 describes a modified cost function. However, the strategy used for optimization of this novel cost function is still that of Section 5.6.2 from the previous chapter. Analogously to the stereo method, our optical flow technique iterates the layer extraction and assignment steps until the costs could not be improved for a fixed number of iterations and returns the solution of lowest costs.

6.2 Colour segmentation and affine model

The method proposed in this chapter applies colour segmentation to the reference image. Translated to motion, the two assumptions that go along with the segmentation are: Firstly, it is assumed that all pixels inside a region of

(a) (b)

Figure 6.2: Colour segmentation. (a) Reference image. (b) Segmented image. Pixels of the same colour belong to the same segment.

homogeneous colour follow the same motion model. Secondly, motion discontinuities are expected to coincide with the boundaries of segments. In Figure 6.2, we present an example of such a segmentation for the reference frame of a motion pair. This motion pair will later be used in the experimental results. The segmentation results are thereby derived by applying the algorithm of Christoudias et al. [23]. The optical flow inside each segment is then modelled by affine motion, which is

$$\begin{aligned} V_x(x,y) &= a_{x0} + a_{xx}x + a_{xy}y \\ V_y(x,y) &= a_{y0} + a_{yx}x + a_{yy}y \end{aligned} \tag{6.1}$$

with V_x and V_y being the x- and y-components of the flow vector at image coordinates x and y and the a's denoting the six parameters of the model. The affine model represents a natural extension to the planar disparity model of Chapter 5 (equation (5.4)) in the presence of two-dimensional motion. It models transformations such as translation, scaling, rotation and shear.

To compute a sparse set of correspondences, we apply the KLT feature tracker [81]. Each segment's affine parameters are then derived by least squared error fitting to all correspondences found inside this segment. We employ the algorithm of Section 5.4 in order to reduce the sensitivity to outliers.

6.3 Layer extraction

Once the initial affine models are known, we aim at identifying those models that represent the dominant motions occurring in the sequence. This is accomplished by clustering of motion segments using the mean-shift-based

algorithm of Section 5.5. However, since we no longer cluster disparity segments, but segments that undergo two-dimensional motion, there is the need to develop a new measurement for computing the dissimilarity of two motion segments that are both described by the affine model. Taking a closer look at the definition of the affine model in equation (6.1), it is clear that the model is the composite of two planar equations, i.e. one for the transformation of the x- and the other for the transformation of the y-coordinate. To determine the dissimilarity of two motion segments, we can therefore use our plane dissimilarity measurement from Section 5.5. More precisely, we compute the dissimilarity of the two planes used for the transformation of the x-coordinate as well as the dissimilarity of the planes that implement the transformation of the y-coordinate. Our motion segment dissimilarity measurement is then defined as the sum of both values.

6.4 Layer assignment

The task of this step is to assign each segment of the reference view to one of the extracted layers. The goodness of such an assignment is computed by evaluation of a global cost function. To calculate the costs, we warp the reference view to the second view using the segments' current motion assignments. Major differences to the layer assignment step of Chapter 5 are twofold. Firstly, we develop a warping technique that does not generate pinholes when using the affine motion model. Secondly, we modify our cost function in order to model the optical flow problem. For computing a local optimum of costs, we apply the algorithm of Section 5.6.2.

6.4.1 Segment warping

In order to generate the warped view, we have to transform each individual segment to the second image according to its current motion model. However, the scanline-based warping approach of Section 5.6 cannot be applied to the case of affine motion. The reason for this is that pixels of the same horizontal scanline in the reference image will have multiple different y-coordinates in the warped second view. The scanline-based technique would therefore generate pinholes in the warped image. To overcome this problem, we developed the simple strategy illustrated in Figure 6.3. In the first step, we compute the bounding rectangle, which encloses the segment that we wish to warp. The bounding rectangle is then transformed to the second view by warping its corner points R_1–R_4 using the segment's affine motion model. The warped bounding rectangle is then defined by the pixels R'_1–R'_4. The

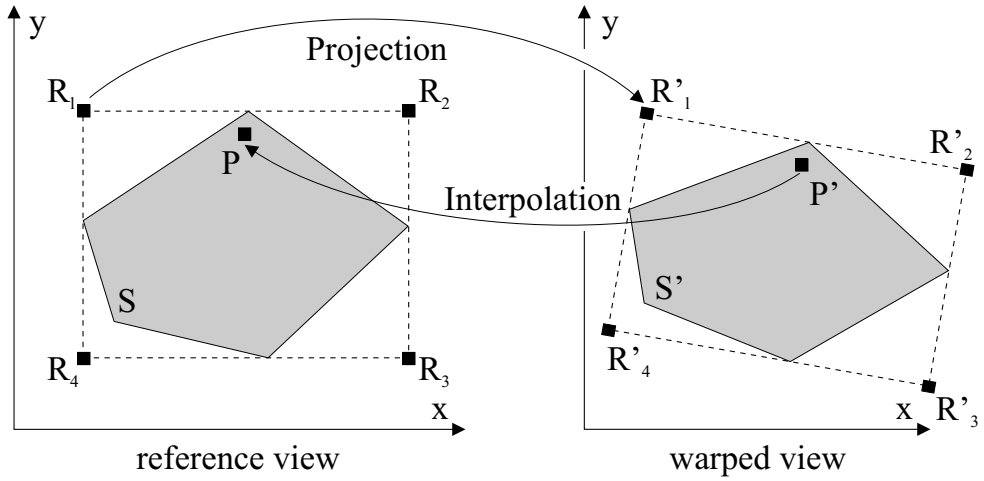
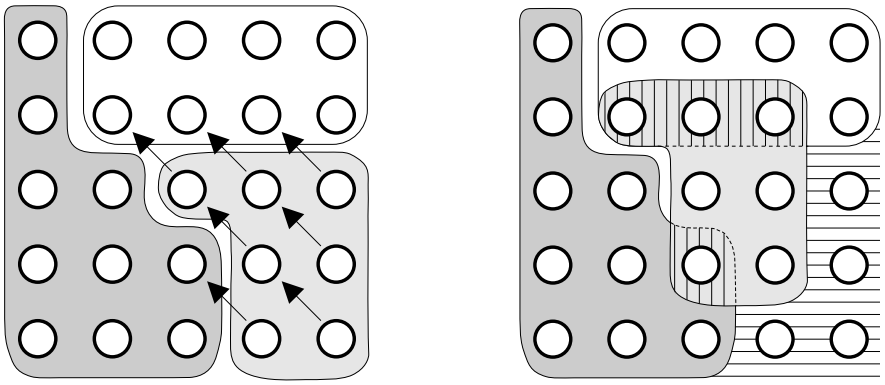


Figure 6.3: Warping a segment to the second view according to its affine motion model.

idea behind this is that the warped rectangle must as well enclose the warped segment S' when using the affine motion model. For each point P' inside this rectangle, we then compute its matching point P in the reference view using the segment's inverse motion model. In the next step, we check if P is inside the segment S of the reference view. If this is not the case, then P' does not belong to the warped segment S' either. Otherwise, we estimate a colour value from the horizontal and vertical neighbours of P by linear interpolation and store this value in a buffer B for the pixel P' of the second view.

6.4.2 Cost function

Analogously to the previous chapter, detection of occlusions and reasoning about visibility has to be performed in the warping process. We illustrate this in Figure 6.4. We show the reference view that is divided into three segments in Figure 6.4a. The estimated motion for two segments is zero, while the third segment undergoes a translational motion as indicated by the arrows. In Figure 6.4b the reference image is then warped according to the estimated motion field. Hatched areas in the image represent regions that are affected by occlusion. Equivalently to the stereo warping process, there are two cases where occlusions are detected. The first and simpler case arises for pixels in the warped image that do not receive contribution from any pixel, which occurs at the horizontally hatched area of Figure 6.4b. This corresponds to an occlusion in the reference view. In the second case, a single



(a) (b)

Figure 6.4: Detection of occlusions by image warping. (a) Reference view. (b) Warped view. Hatched areas represent occluded regions.

pixel of the warped view gets contribution from more than one pixel of the reference view, which is illustrated by the vertically hatched areas of Figure 6.4b. Since we assume surfaces to be opaque, only one of those pixels can be visible. Consequently, the other pixels are occluded in the second view. Unfortunately, and this is the main difference to the method of Chapter 5, for a motion algorithm the reasoning about the pixels' visibility is not obvious. The reason for this is that there is, in general, no relationship between a pixel's motion vector and its depth in the scene. Therefore, Z-buffering does not work. However, it is quite reasonable to assume that the visible point has a very similar colour value to the pixel at the same coordinates in the real second view. Consequently, we decided to declare the point of lowest pixel dissimilarity as being visible, while the other pixels are marked as being occluded.

To account for this different visibility reasoning, a reformulation of the cost function given in equation (5.17) is required. We avoid unnecessary redundancy by stating only those terms that are modified. This involves the data term defined in equation (5.10), which is computed over the set of visible pixels Vis. In the case of motion, we determine Vis by

$$Vis = \{\cup_{x,y}\, p \in B_{x,y} \mid \forall q \in B_{x,y} : dis(W(p), R(p)) < dis(W(q), R(q)) \vee p = q\} \tag{6.2}$$

with dis being the function from equation (5.12) that calculates the pixel dissimilarity and W and R being the warped and real second views, respectively. We write $B_{x,y}$ to denote the buffer cell at coordinates (x, y). The second term

that we need to consider is the occlusion term stated in equation (5.13). The occlusion term is defined over the set of occluded pixels in the second view Occ_R and in the reference image Occ_L. In the case of motion, we define Occ_R by

$$Occ_R = \{\cup_{x,y}\ p \in B_{x,y} \mid \exists q \in B_{x,y} : dis(W(p), R(p)) > dis(W(q), R(q))\}\,. \tag{6.3}$$

For completeness, we as well present the definition of the set Occ_L:

$$Occ_L = \{\cup_{x,y}\ B_{x,y} \mid B_{x,y} = \emptyset\}\,. \tag{6.4}$$

6.5 Experimental results

We demonstrate the performance of the proposed algorithm using the frames 50 and 54 of the Mobile & Calendar sequence[1] that are shown in Figure 6.5a and 6.5b, respectively. In this sequence, the camera pans to the left, while there are moving objects (calendar, train and ball) in the scene. Since no ground truth is available, we have to focus on a qualitative discussion of the results. Figure 6.5c presents the final layer assignment. Although motion segmentation is not the primary goal of this work, the computed layers seem to correspond well to scene objects. To visualize the flow field, we plot the absolute x- and y-components of the flow vectors scaled by a factor of 32 in Figures 6.5d and 6.5e. Motion boundaries appear to be correctly captured, while also the image motion in untextured regions seems to be accurately identified (e.g., lower part of calendar). Finally, we show the two-dimensional flow vectors for some pixels of the reference frame in Figure 6.5f. For the 352×240 pixel input images, our current C++ implementation needed 47 seconds on an Intel Pentium 4 2.0 GHz computer to generate the results.

As a second test pair, we used the frames 11 and 14 of the Tennis sequence[2] that are shown in Figures 6.6a and 6.6b. There are two moving objects in the scene, which are the arm and the ball. While the arm undergoes a relatively small motion, there is large motion on the ball. Since the x-components of the flow vectors are almost zero, we decided to show the warped view in Figure 6.6c instead. This image is generated by warping the reference view according to the computed flow vectors and should be compared against the real second view presented in Figure 6.6b. Regions that

[1] There is very little motion between the frames 50 and 51. For the task of motion segmentation, this motion is not enough to capture the different layers present in the scene. This is why we select frames 50 and 54.

[2] We select frames 11 and 14, since we want to test our algorithm on an image pair that contains large motion.

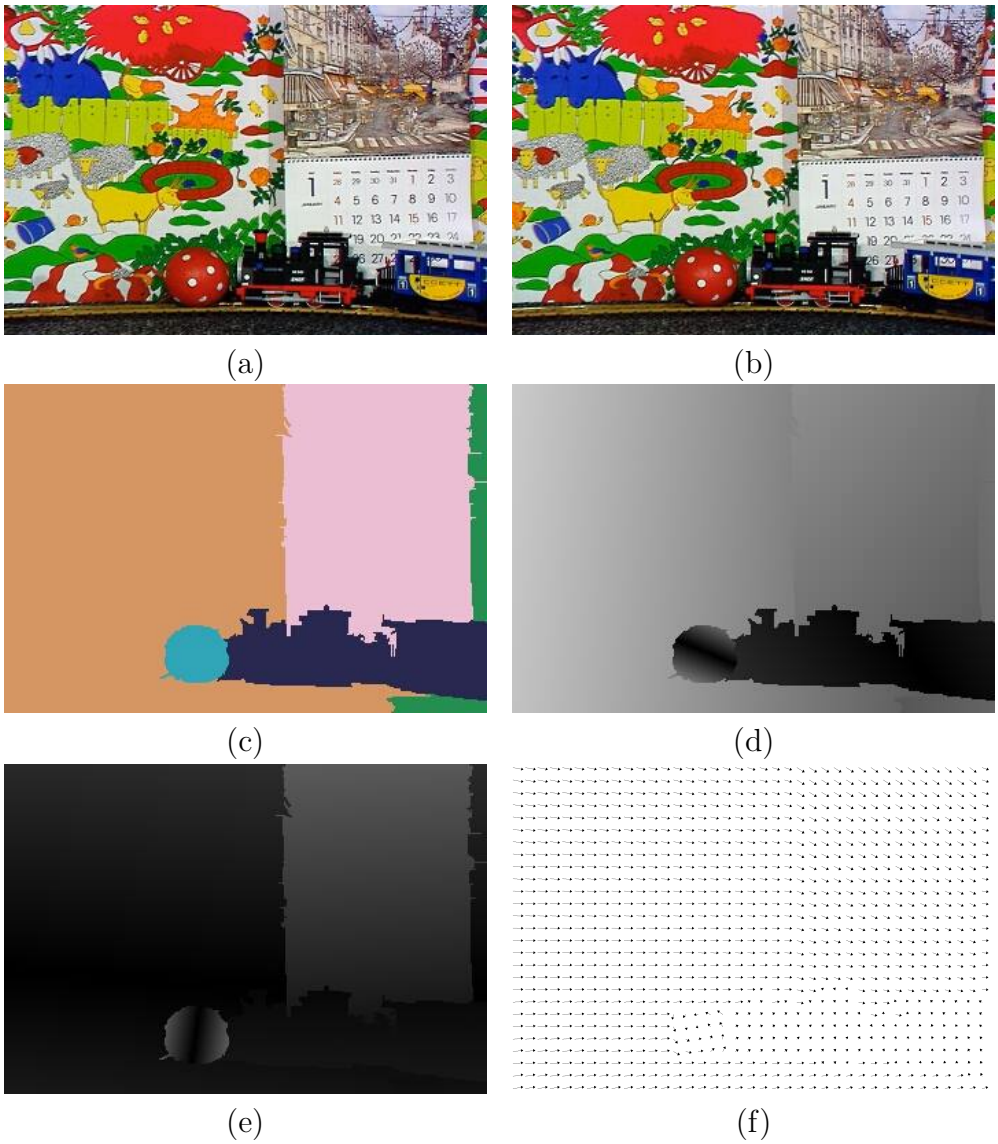


Figure 6.5: Results for the *Mobile & Calendar* sequence. (a) Frame 50. (b) Frame 54. (c) Final layer assignments. (d) Absolute x-components. (e) Absolute y-components. (f) Flow vectors.

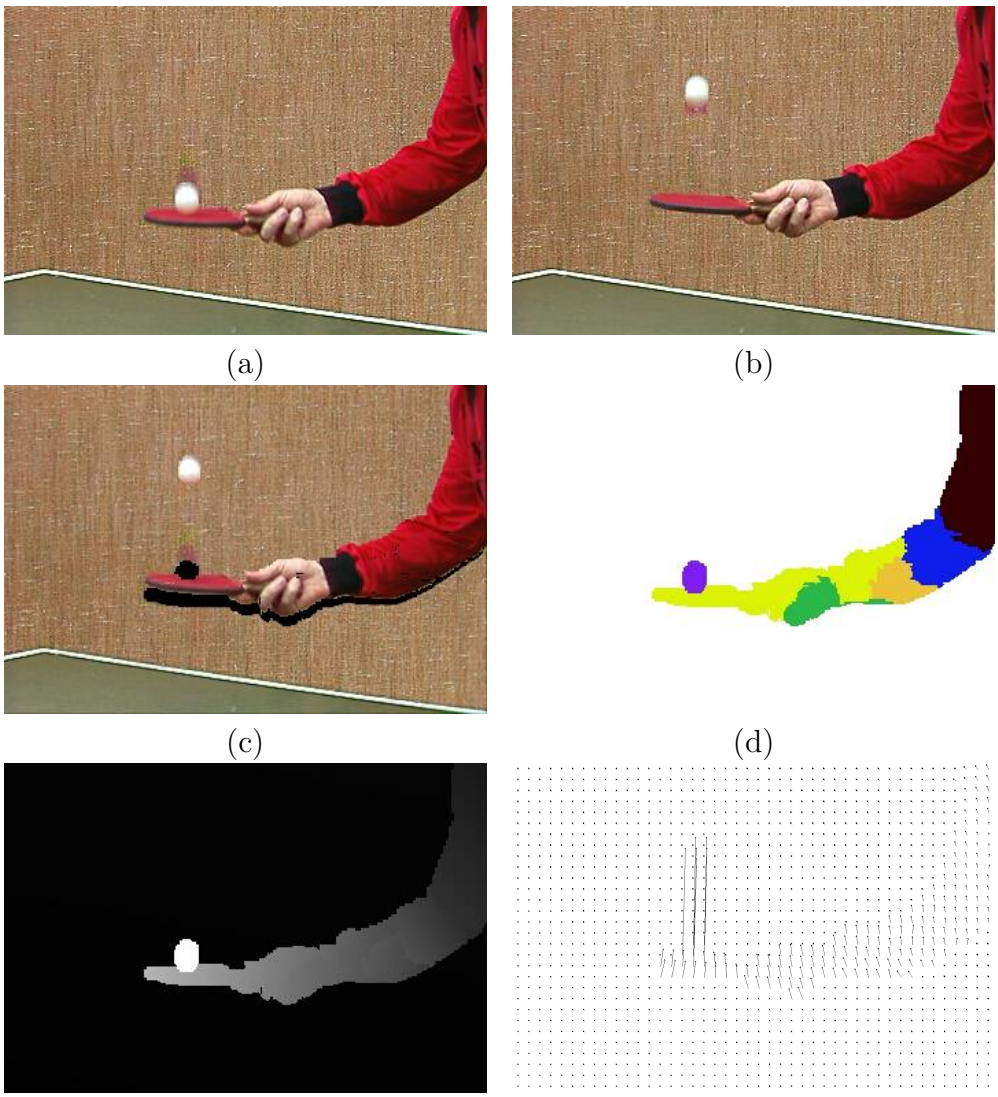


Figure 6.6: Results for the *Tennis* sequence. (a) Frame 11. (b) Frame 14. (c) Warped image. (d) Final layer assignments. (e) Absolute y-components. (f) Flow vectors.

were identified as being occluded in the reference view are coloured black. We then present the final layer assignment in Figure 6.6d. The arm is thereby represented by five different layers, which is most likely for the reason that only a single affine motion model can hardly capture the real motion of the arm. The y-components of the flow vectors scaled by a factor of 16 are then shown in Figure 6.6e. The motion boundaries seem to be correctly identified and also the large motion of the ball seems to be captured. Finally, we present the corresponding flow vectors in Figure 6.6f. 99 seconds were needed to generate the results for the 352×240 pixel images.

6.6 Summary

In this chapter, we have extended the stereo method of Chapter 5 to the optical flow problem. Optical flow computation suffers from the same problems as stereo matching, which has been the major motivation to test our segmentation-based stereo algorithm on this problem. We have described the modifications necessary to adapt our method to the motion correspondence problem. This includes a model that can describe two-dimensional motion, for which we have chosen the affine model. Furthermore, we have presented a way how to modify the layer extraction and assignment steps in order to model the optical flow task. A major problem is that the Z-buffer mechanism of the stereo technique cannot be applied in the case of motion. We have therefore chosen to base our reasoning about a pixel's visibility on its colour similarity to the real second view. Experimental results have demonstrated the good performance of the algorithm, especially in usually difficult regions, such as areas of poor texture or close to motion boundaries.

Chapter 7

A graph-cut formulation

7.1 Introduction

In Chapter 5, we have formulated stereo as a two step problem. First, a set of disparity layers, which correspond to dominant depth planes occurring in the scene, is extracted. Second, each region of an image is then assigned to exactly one of those disparity layers. The second of these steps can thereby be regarded as the more challenging one, since it requires to tackle all those points that make stereo matching difficult (textureless areas, occlusion problem, etc.). In the previous chapters, we have modelled this layer assignment problem as a cost minimization task. One problem related with this methodology is that having a cost function that well models a problem is not sufficient, if one is not able to effectively optimize it. Unfortunately, this holds partially true for the previously proposed approach. Upon bad initialization, the greedy algorithm that is employed for optimization can get stuck in a "weak" optimum due to its local nature, and there is no guarantee on how far such a local optimum is away from the global one. In this chapter, we aim to overcome this problem by employing a robust global optimization scheme to the layer assignment task, namely graph-cuts. A drawback of that approach, however, is the computational lower efficiency. As will be explained later in this chapter, optimization via graph-cuts works only for a restricted type of cost functions, and when developing a cost function one has to ensure that it belongs to this class. This is why we cannot directly use graph-cuts to optimize the cost function of Chapter 5, but have to design a new one. Throughout this chapter, we assume that the layer extraction task is already solved so that the disparity layers are known. A disparity layer is still described by a planar equation. Nevertheless, our approach can, in theory, be used in conjunction with any smooth surface model.

The goal of this chapter is to develop a cost function for the layer assignment problem that takes benefit of the segmentation information and can then be optimized via graph-cuts. The major motivation for using the segmentation information is that graph-based approaches often optimize energy functions whose smoothness terms bias towards the reconstruction of simple object shapes, i.e. they aim at minimizing border lengths. We overcome this undesired property by enforcing disparity discontinuities to coincide with segment borders. Special care is taken on the accurate treatment of occlusions. This, together with the segmentation information, leads to an improved performance in regions close to depth boundaries. Using a region-based approach, we take benefit of increased robustness in regions of poor texture as well as of the capability to hypothesize flow values for occluded areas, which in our problem formulation is even possible if the whole segment is occluded. The major novelty of our approach lies in the way how occlusions are dealt with. In order to correctly handle occlusions, we introduce a cost function that is defined on two levels, one corresponding to pixels and the other to segments. We describe the basic idea behind this in the following.

7.2 Problem formulation

7.2.1 Basic idea

The cost function that we will develop in the following builds upon the observations of Section 4.3 where we have discussed the role of occlusions in region-based matching. To explain our approach, let us again consider the example that we have used in this previous discussion. Figure 7.1a illustrates two segments with the foreground segment S_2 being slightly displaced in the right view. Moreover, we show the resulting occluded areas of both frames marked by red colour in Figure 7.1b. Let us then consider Figure 7.1c to explain how we can model this example in the proposed cost function.

We thereby start by taking a closer look at the segment level shown at the top of the illustration. The segment level corresponds to all those regions extracted by colour segmentation in the reference (left) view, which are the segments S_1 and S_2 in our example. Note that the matching primitives on this level are complete segments. Each of them gets assigned to a specific disparity model (or in our terminology, disparity layer). So a typical statement on the segment level would, for example, be: There is zero disparity on segment S_1. However, as we know from our previous discussion, in the domain of segments it is not possible to express the fact that a segment is partially

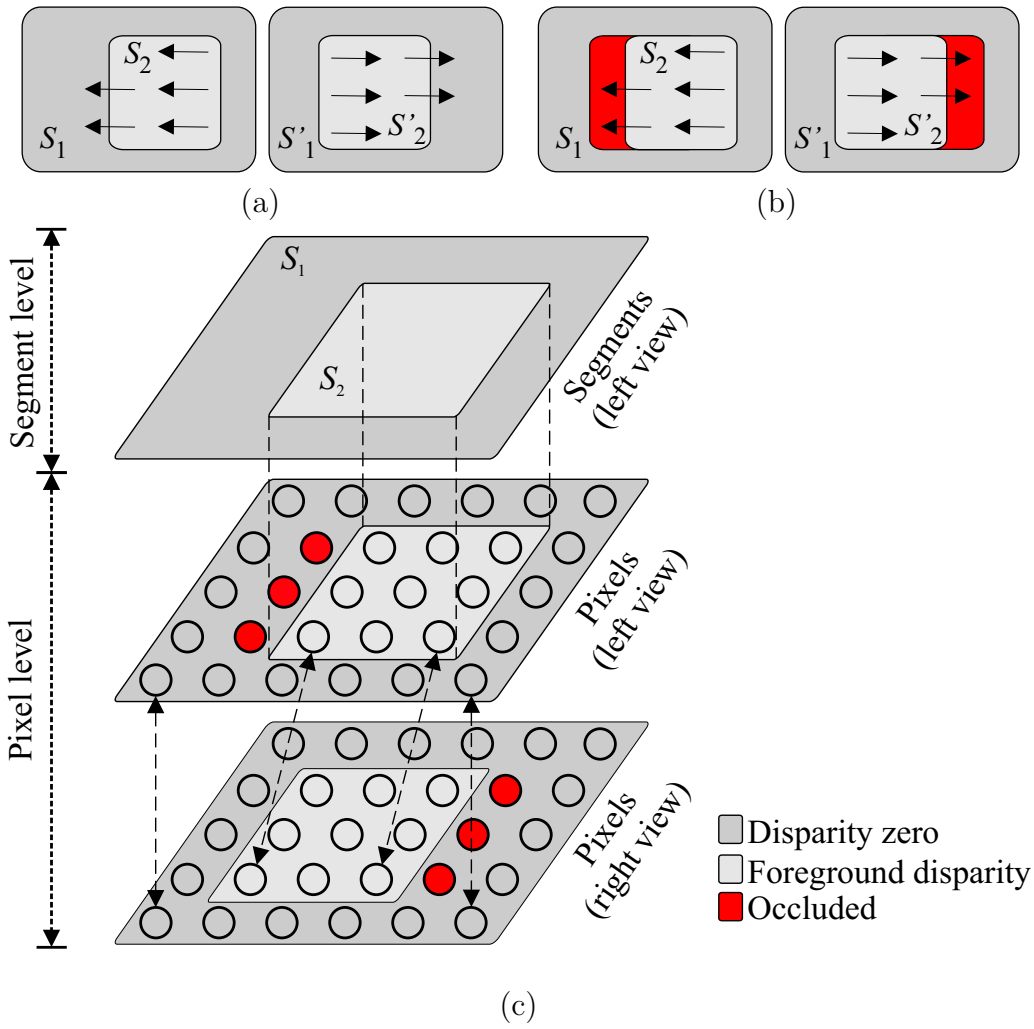


Figure 7.1: Basic idea behind the proposed cost function. Explanation is given in the text.

occluded[1]. Nevertheless, this is required in our example in order to correctly model the occluded parts of segment S_1.

In the proposed approach, we overcome the problem of partially occluded segments by including the set of all pixels of the reference view into our cost function. This is illustrated by the middle layer of Figure 7.1c. Therefore, in addition to all segments, we as well assign every pixel of the reference image to a disparity layer. In our formulation, there is a dependency between the disparity layer assignments of segments and pixels of the reference view. This link is built by the segmentation assumption. Recall that the assumption states that all pixels inside a segment follow the same disparity model. Our formulation implements this segmentation constraint by enforcing that every (visible) pixel is assigned to the same disparity layer as the segment to which it belongs. However, and this is the important point, a pixel is also allowed to be occluded. Figure 7.1c shows that by inclusion of the middle layer representing the pixels of the left image we are now able to correctly model the partial occlusion to the left of segment S_2 in the reference view. The dashed lines between segment and pixel levels shall thereby indicate that the segmentation assumption is enforced on the pixel level.

Nevertheless, at this point, we still have not modelled the complete information that is present in the input image pair, since occlusions in the second view have not been considered so far. To account for those occluded regions, we as well include every pixel of the right image into our problem formulation. This is represented by the bottom layer of Figure 7.1c. The disparity layer assignments of pixels in the left image thereby depend on the assignments of points in the right view and vice versa. Our basic consistency constraint is that a (visible) pixel and its matching point in the other image must both have identical disparity layer assignments. As seen from Figure 7.1c, we are now able to model the occlusions to the right of segment S_2 in the second view. The arrows between middle and bottom layers illustrate the consistency constraint that operates between the left and right views. Modelling the pixels of both images in our cost function allows us to treat occlusions symmetrically. Moreover, it serves to model the uniqueness constraint, as we will describe later in this chapter.

7.2.2 Notations

In the following, we define some notations that are required in order to set up our cost function. We thereby regard the task of assigning pixels and

[1]Indeed, one could only state that a segment is occluded as a whole. However, occluded regions will almost never coincide with regions extracted by colour segmentation.

segments to disparity layers as a labelling problem. The labels $1, 2, \ldots, N$ correspond to the N disparity layers that have been computed in the layer extraction step. Moreover, a dedicated label 0 denotes pixels and segments that are occluded and therefore not assigned to any of those disparity layers. A labelling function $f(\cdot)$ is then defined for both, pixels and segments.

Let $p = (x, y, v)$ be a pixel defined by its image coordinates x and y as well as its view $v \in \{LEFT,\ RIGHT\}$. The set $I = I_{LEFT} \cup I_{RIGHT}$ denotes the union of all pixels from both views, with I_{LEFT} being the left image and I_{RIGHT} being the right image. The labelling function $f(p)$ on the pixel level then projects each pixel $p \in I$ to exactly one label k:

$$\forall p \in I: \qquad f(p) = f(x, y, v) = k, \qquad k \in \{0, 1, 2, \ldots, N\}. \tag{7.1}$$

Moreover, let S be the set of segments extracted in the left view. Analogously, the labelling function $f(s)$ on the segment level projects each segment $s \in S$ to exactly one label k:

$$\forall s \in S: \qquad f(s) = k, \qquad k \in \{0, 1, 2, \ldots, N\}. \tag{7.2}$$

Labelling a pixel by a label $k \neq 0$ defines the corresponding point in the other view. The matching point $m[k](p)$ of pixel $p = (x, y, v)$ assigned to label k is obtained by computing the disparity according to equation (5.4) at the point (x, y, v) using the plane model of the kth disparity layer and adding it to x. Formally expressed,

$$m[k](p) = m[k](x, y, v) = (x + d[k](x, y, v), y, \neg v) \tag{7.3}$$

with $d[k](x, y, v)$ being the disparity at point (x, y, v) according to the plane model of the kth disparity layer and $\neg LEFT = RIGHT$ and vice versa. The plane parameters used for computation of $d[k](x, y, v)$ depend on the view v. A transformation from $LEFT$ to $RIGHT$ is done using the original plane parameters, which results in negative disparity values, whereas a transformation in the opposite direction is accomplished using the parameters of the "inverse" plane, which gives positive disparity values. To derive whole-numbered image coordinates, we round the computed disparity to the closest neighbour.

7.2.3 Cost function

Using the notation introduced above we design a cost function $C(f)$, which measures the optimality of a label configuration f. We therefore define a set of terms that incorporate our basic ideas. Some of these terms operate

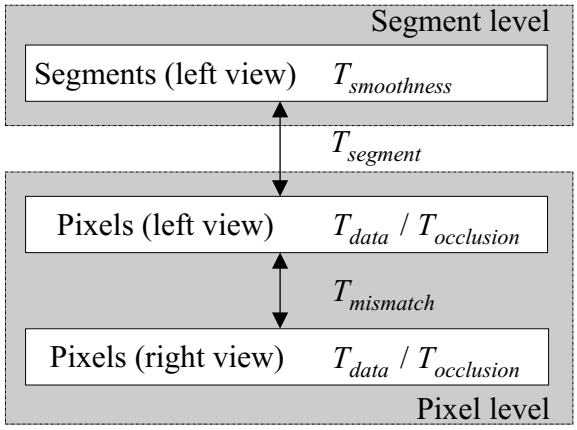


Figure 7.2: Terms of the cost function $C(f)$ and their scope.

directly on the pixel level or on the segment level, while others propagate disparity layer assignments between the different layers of Figure 7.1c. We give an overview of those terms and their scope in Figure 7.2. The overall cost function $C(f)$, which is subject to minimization, is then built by summation of these terms:

$$C(f) = T_{data}(f)+T_{occlusion}(f)+T_{segmentation}(f)+T_{mismatch}(f)+T_{smoothness}(f). \tag{7.4}$$

The individual terms of $C(f)$ are described one after the other in the following.

Data term

The first term T_{data} measures the agreement of f with the input data by exploiting the photo consistency constraint. Thus, we assume that pixels of the left and right images that are the projections of the same scene point show similar colour values. We incorporate this assumption on the pixel level by measuring the pixel dissimilarity at each visible point of both images. More precisely, we compute the dissimilarity between a pixel p and its matching point $m[f(p)](p)$ in the other view according to p's current disparity layer assignment $f(p)$. Formally, the data term T_{data} is defined by

$$T_{data}(f) = \sum_{p \in I} \begin{cases} dis(p, m[f(p)](p)) & : \quad f(p) \neq 0 \\ 0 & : \quad \text{otherwise} \end{cases} \tag{7.5}$$

with $dis(p_i, p_j)$ being a function that computes the colour dissimilarity of two pixels p_i and p_j. In our current implementation, we use the pixel dis-

similarity measurement of Birchfield and Tomasi [9]. This measurement has the advantage of being less sensitive to image sampling. Since it has been originally proposed to compute the dissimilarity of grey value pixels only, we have applied a simple modification in order to make the measurement work on RGB values as well.

Occlusion term

The occlusion term of our cost function serves to penalize occluded pixels in both input views. This penalty is necessary, since otherwise declaring all pixels as occluded would result in a trivial minimum of $C(f)$. We therefore define the occlusion term $T_{occlusion}$ by

$$T_{occlusion}(f) = \sum_{p \in I} \begin{cases} \lambda_{occ} & : \quad f(p) = 0 \\ 0 & : \quad \text{otherwise} \end{cases} \tag{7.6}$$

with λ_{occ} denoting a constant user-set parameter.

Segmentation term

The segmentation term propagates disparity layer assignments between the segments and the pixels of the reference view. It enforces the assumption of smoothly varying disparity inside a segment on the pixel level. We embed this assumption by imposing a penalty set to infinity for every visible pixel of the left image that carries a different disparity layer assignment than its corresponding segment. Formally, we define the segmentation term $T_{segment}$ by

$$T_{segment}(f) = \sum_{p \in I_{LEFT}} \begin{cases} \infty & : \quad f(p) \neq 0 \wedge f(p) \neq f(seg(p)) \\ 0 & : \quad \text{otherwise} \end{cases} \tag{7.7}$$

with $seg(p)$ being a function that returns the segment to which the pixel p belongs. The consequence of the segmentation term is the following. Let us consider two pixels of the same segment. Both pixels are visible, i.e. they do not carry the occlusion label. If one pixel is now assigned to the same disparity layer as its corresponding segment, then also the other pixel must be assigned to exactly this particular disparity layer. Otherwise, the segmentation term generates infinite costs and such a configuration will therefore not be produced in the optimization part of the algorithm. Consequently, it is not possible that two pixels of the same segment are assigned to two different disparity layers. This is obviously equivalent to the statement that all non-occluded pixels inside a segment are modelled by the same disparity

layer, which is exactly what our segmentation assumption requires. This is why the term introduced above enforces the segmentation constraint on the pixel level. However, note that occluded pixels are not affected by the segmentation term. Therefore, a pixel of the reference view can always carry the occlusion label independently of its segment's disparity layer assignment.

View consistency term

The view consistency term propagates disparity layer assignments from the reference image to the second view and vice versa. It motivates consistent disparity layer assignments across views, meaning that if a pixel in one image is assigned to a particular disparity layer, also its matching point in the other image should be assigned to exactly this disparity layer. We call assignments that violate this constraint view inconsistent. Such view inconsistent assignments are penalized by adding a constant value to the solution's costs. We define the view consistency term $T_{mismatch}$ by

$$T_{mismatch}(f) = \sum_{p \in I} \begin{cases} \lambda_{mismatch} & : \quad f(p) \neq 0 \ \wedge \ f(p) \neq f(m[f(p)](p)) \\ 0 & : \quad \text{otherwise} \end{cases} \tag{7.8}$$

with $\lambda_{mismatch}$ being a user-defined penalty. This term is as well used in the work of Lin and Tomasi [57]. Ideally, view consistency should be enforced by penalizing inconsistent solutions with infinite costs. This is, however, not possible in our formulation for reasons related to the optimization part of the algorithm. More precisely, view inconsistent solutions are generated in intermediate steps of the optimization method.

Smoothness term

The last term of our cost function is the smoothness term. Note that smoothness is, to some extent, already enforced due to the region-based nature of the proposed algorithm. However, we apply a strong oversegmentation and therefore image areas that can be well modelled by the same disparity layer will, in general, be represented by more than one segment. Consequently, it makes sense to incorporate an explicit smoothness term into our cost function in order to propagate disparity layer assignments across neighbouring segments. Our cost function implements the smoothness assumption on the segment level by penalizing neighbouring segments that are assigned to different disparity layers. Formally, the smoothness term $T_{smoothness}$ is computed

by

$$T_{smoothness}(f) = \sum_{(s_i,s_j)\in NB} \begin{cases} \lambda_{disc} \cdot bl(s_i, s_j) \cdot cs(s_i, s_j) & : \quad f(s_i) \neq f(s_j) \\ 0 & : \quad \text{otherwise} \end{cases} \tag{7.9}$$

with λ_{disc} being a user-set constant penalty for discontinuity and NB being the set of all neighbouring segments. The function $bl(s_i, s_j)$ computes the border length by counting the number of neighbouring pixels (p_i, p_j) in 4-connectivity with p_i belonging to segment s_i and p_j to segment s_j. The second function $cs(s_i, s_j)$ measures the colour similarity of segments s_i and s_j. The basic idea behind weighting the smoothness penalty by the function $cs(\cdot, \cdot)$ is that we consider two segments showing similar colour as more likely to originate from the same real-world surface than two segments of completely different colour. As an example, consider an image background of relatively homogeneous colour that is divided into several segments by colour segmentation. In our implementation, we define the function $cs(s_i, s_j)$ by

$$cs(s_i, s_j) = (1 - \frac{\min(|meancolour(s_i) - meancolour(s_j)|, 255)}{255}) \cdot 0.5 + 0.5 \tag{7.10}$$

with $meancolour(s)$ being the componentwise summed up RGB values of pixels inside segment s divided by the segment's number of pixels. The absolute difference of the two RGB values is computed by summing up the absolute differences of each component, which gives a maximum value of $3 \cdot 255$ using an 8-bit coding for each colour channel. For identical mean colour values, the colour similarity function returns a value of 1, whereas for colour differences larger or equal to 255, it gives a value of 0.5. The costs of assigning two neighbouring segments of similar colour to different disparity layers are therefore higher than separating two segments of low colour similarity.

The reasons why the smoothness term operates on the segment level instead of being defined on both views in the domain of pixels are twofold. Firstly, defining the term on the pixel level would mean that smoothness is as well enforced for pixels which carry the occlusion label. However, while the smoothness assumption, in general, holds true for visible compact shapes, it is not valid for occluded areas that are usually long and thin. Secondly, defining the smoothness term on the segment level allows to propagate "meaningful" disparity layers to segments that are completely occluded, i.e. segments that do not contain a single visible pixel on the pixel level.

7.2.4 Modelling the uniqueness assumption

The uniqueness assumption is known to be powerful in the identification of occlusions. Recall from Section 2.2.2 that this constraint states that a pixel of one view matches at most a single pixel in the other image. Incorporation of the uniqueness assumption into our approach is relatively simple, since it just requires careful setting of two parameters of the cost function, as we describe in the following.

Let us first construct an example where the uniqueness constraint can help us to detect an occluded pixel. We illustrate such a case in Figure 7.3a. The illustration shows two pixels p_1 and p_2 assigned to different disparity layers l_1 and l_2, respectively. According to their current disparity layer assignments, both pixels project to the same matching point p'_2. Therefore, if we assume that the uniqueness constraint is valid, p_1 and p_2 cannot be visible at the same time. Obviously, our cost function penalizes configurations such as the one of the illustration. More precisely, since p_1 and p_2 originate from different surfaces (i.e. they are modelled by different disparity layers), only one of them can have a view consistent disparity layer assignment with its matching point p'_2. This is the pixel p_2 in our example, and the costs for assigning p_2 are solely that produced by the data term of equation (7.5) (see also Figure 7.3a). In contrast to this, the costs for the assignment of p_1 are not only those given from the data term, but as well those of the view consistency term of equation (7.8), which adds $\lambda_{mismatch}$ to the overall costs (see again Figure 7.3a). The basic idea now is to not only penalize configurations that violate the uniqueness constraint, but rather to avoid them completely. Our simple solution to this is to set the occlusion penalty λ_{occ} given by the occlusion term of equation (7.6) to a lower value than that of the mismatch penalty $\lambda_{mismatch}$.[2] By doing so, we guarantee that the costs for declaring a pixel as occluded are always lower than the costs for assigning it to a view inconsistent disparity layer. Referring again to Figure 7.3a, in the best case, the view inconsistent assignment of p_1 generates costs of $\lambda_{mismatch}$ (if there is a perfect agreement in colour values between p_1 and p'_2 so that the pixel dissimilarity is zero). However, the costs for assigning p_1 to the occlusion label are λ_{occ} with $\lambda_{occ} < \lambda_{mismatch}$ and therefore this is the configuration that will be produced by the optimization algorithm.

In our discussion of the uniqueness assumption in Section 2.2.2, we have pointed out that this constraint does not hold true for slanted surfaces. Due to different sampling in the two input images, there are pixels of the same surface that correctly match more than one point of the other view. Application of the uniqueness constraint for the reconstruction of slanted surfaces

[2] In our experiments, we use $\lambda_{occ} := \lambda_{mismatch} - 1$.

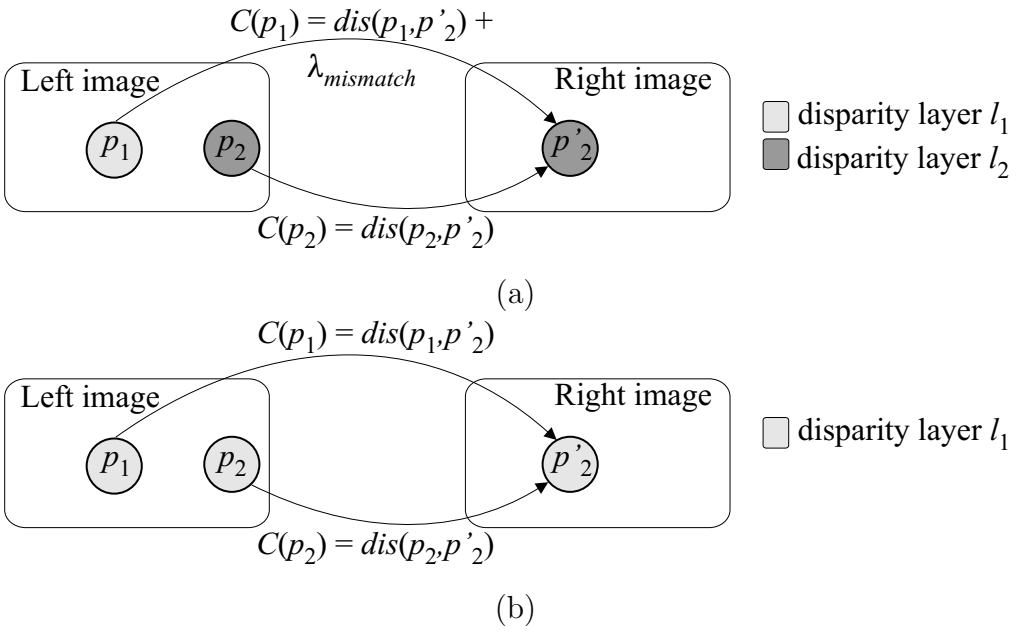


Figure 7.3: Modelling the uniqueness assumption. (a) Two pixels assigned to different disparity layers match the same pixel in the other view. (b) Two pixels assigned to the same disparity layer match the same pixel in the other view. More details are found in the text.

can therefore lead to suboptimal results [69]. In Figure 7.3b, we illustrate two pixels p_1 and p_2 that are both assigned to disparity layer l_1, which means that they lie on the same surface. Due to some slant on their surface, both pixels have the same matching point p'_2 in the other view. For this reason, this configuration clearly violates the uniqueness constraint. However, both disparity layer assignments (i.e. that of p_1 as well as that of p_2) are view consistent with the assignment of the matching point p'_2, since all of the three pixels are assigned to disparity layer l_1. Therefore, our cost function does not penalize this configuration by the view consistency term (see Figure 7.3b). So what we actually implement is not strictly the uniqueness constraint, but rather some improved form of it that can also handle slanted surfaces.

7.3 Optimization

7.3.1 α-expansion algorithm

To approximate a labelling f of minimum costs $C(f)$, we use the α-expansion framework of Boykov et al. [17]. This framework represents an efficient optimization strategy for various labelling problems in computer vision and has been briefly discussed in Section 3.1.5 in the context of stereo. The main idea behind the optimization scheme is the following. Exact minimization of cost functions such as the one that we aim to optimize is known to be $\mathcal{NP}$-complete. It is therefore almost surely not possible to compute the global optimum in polynomial time. However, for a specific type of cost functions that is characterized by Kolmogorov and Zabih [52], it is possible to determine the global optimal solution for a subproblem via a single cut in a graph. Roughly described, this subproblem is: Given some label configuration f, what is the label configuration f' of lowest costs that differs from f in the sense that a subset of labels are changed to the label α? In other words, the subproblem is to find the optimal expansion of the label α. Iteratively solving this subproblem by expanding each label efficiently approximates the global optimum of costs for the overall problem. We go into more detail on this in the following.

Translated to our problem formulation, an α-expansion move changes the assignment of a subset of pixels and segments to the label α and leaves the other pixels and segments assigned to their old labels. Formally expressed, let f be the current label configuration of pixels and segments. The configuration f' is within one α-expansion move from f, if for each pixel p, $f'(p) = f(p)$ or $f'(p) = \alpha$ and for each segment s, $f'(s) = f(s)$ or $f'(s) = \alpha$. We give an example of an α-expansion move on the segment level in Figure 7.4. The problem of finding the move of lowest costs within one α-expansion from f for our cost function is then solved to optimality by computing the cut in a special purpose graph.

We embed the α-expansion move into a greedy algorithm, which is the one proposed by Boykov et al. [17]. An initial label configuration is generated by assigning all pixels and segments to the occlusion label. Note that the α-expansion algorithm is robust enough to produce strong results even if the initial configuration is far away from the global optimum. Starting from this configuration, the algorithm computes the cheapest α-expansion move for each disparity layer in fixed or random order. In addition to the extracted layers, we also test a special disparity layer that carries the occlusion label. If a move decreases the costs, then this is the new label configuration. This procedure is then iterated until there is no disparity layer that further de-

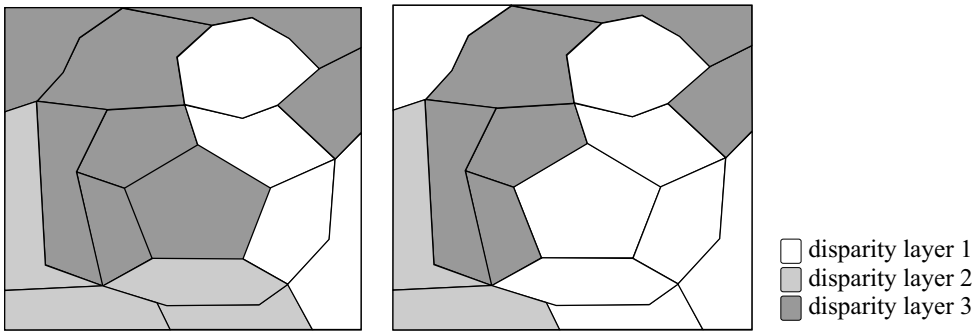


Figure 7.4: α-expansion on the segment level. (a) The image is divided into a set of segments. Each segment is assigned to one of three disparity layers. (b) α-expansion of disparity layer 1. Some segments change their assignments to disparity layer 1, while the others keep their original assignments.

creases the costs by application of the α-expansion, which is usually the case after very few iterations. We present a block diagram of the greedy algorithm in Figure 7.5.

7.3.2 Optimal α-expansion move via graph-cuts

The problem of finding the optimal α-expansion move for our cost function among all possible α-expansion moves is solved by computing the minimum cut in a special purpose graph. Let $\mathcal{G}$ be a weighted directed graph with two special vertices, which are the source src and the sink snk. From Section 3.1.5 we recall that a cut divides the vertices of $\mathcal{G}$ into two disjoint sets SRC and SNK so that $src \in SRC$ and $snk \in SNK$. The sum of all edges that go from SRC to SNK defines the costs of the cut and the minimum cut is the one that exhibits lowest costs.

To find the optimal α-expansion move for our cost function, we construct the following graph. Each segment as well as each pixel are represented by exactly one vertex v_i. Additionally, the graph contains the two dedicated vertices src and snk. Edges between vertices represent terms of the cost function. The resulting graph is illustrated in Figure 7.6.

The basic idea behind the construction of this graph is that there is a one-to-one correspondence between cuts in our graph and label configurations f' within one α-expansion from the current assignment f. Since an α-expansion move can be regarded as binary labelling, we can represent each pixel and segment by a binary variable x_i with $x_i = 0$ if the old label is kept, $f'(\cdot) =$

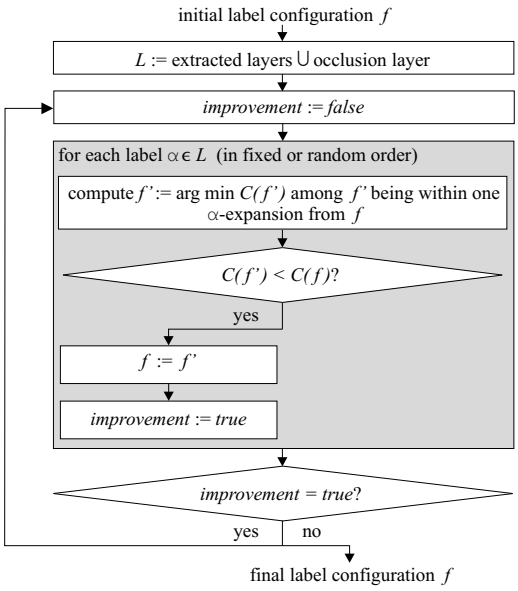


Figure 7.5: Block diagram of the greedy algorithm that employs the α-expansion move.

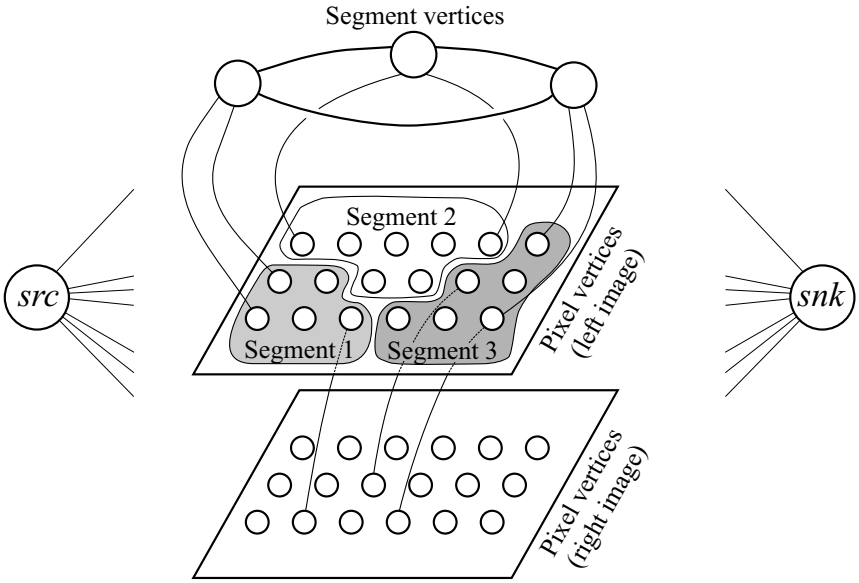


Figure 7.6: Layout of the graph. Not all edges are shown for legibility. Each of the illustrated vertices is connected to the source src and sink snk vertices.

$f(\cdot)$, and $x_i = 1$ if the new label is taken, $f'(\cdot) = \alpha$. Analogously, a cut in the graph represents a binary partition of vertices into the source set SRC and the sink set SNK. We define the correspondence between a vertex v_i after computation of the cut and the binary labelling x_i by

$$x_i = \begin{cases} 0 & : \quad v_i \in SRC \\ 1 & : \quad v_i \in SNK. \end{cases} \tag{7.11}$$

Therefore, each cut in the graph uniquely defines the new label configuration f' and vice versa. Edges in the graph are inserted in a way that the costs of each cut in the graph are equal to the costs of the resulting label configuration f'. Since the computed cut is the one of minimum costs, also the resulting configuration f' has minimum costs within one α-expansion move from f.

Kolmogorov and Zabih [52] characterize the class of cost functions that can be minimized by graph-cuts. According to their results, cost functions of n binary variables in the form of

$$C(x_1, \cdots, x_n) = \sum_i C^i(x_i) + \sum_{i<j} C^{i,j}(x_i, x_j)$$

can be optimized by graph-cuts if and only if

$$C^{i,j}(0,0) + C^{i,j}(1,1) \leq C^{i,j}(0,1) + C^{i,j}(1,0). \tag{7.12}$$

This defines the condition that we have to check for each individual term of our cost function in order to prove that it belongs to this class. We then adjust the weights of graph edges in a way that the resulting graph represents our cost function. This is done using the construction rules given by Kolmogorov and Zabih [52]. Since this construction is fairly complex, we have decided to postpone its discussion to the appendix B.1. The minimum cut in this graph is computed by applying the maximum flow algorithm of Boykov and Kolmogorov [16]. This maximum flow method is specifically optimized for high computational performance on graphs arising in computer vision.

7.4 Experimental results

To test our new layer assignment step, we first run the layer extraction step of Chapter 5 on the input image pair. Knowing the disparity layers, we then invoke the proposed layer assignment procedure. Upon convergence of the α-expansion algorithm, we refit each disparity layer that is present

in the generated solution over its new spatial extent. We again run the α-expansion algorithm in order to check whether any of those new disparity layer models can produce a solution of lower cost. If this is the case, the procedure is iterated. Otherwise, the algorithm returns the current disparity layer assignment as final output.

To evaluate the proposed algorithm, we again use the Middlebury test bed provided by Scharstein and Szeliski [78] that has been described in Section 3.2 of this thesis. We therefore generated results for all of the four test image pairs used in the benchmark. The algorithm's parameters were thereby kept constant. At the time of publication of the corresponding conference paper, the proposed method was ranked on second place[3] on the Middlebury Stereo Vision website.[4] The graph-cut method shows slightly better overall performance than the stereo algorithm of Chapter 5. This can also be seen from Table 3.1 that shows a more recent ranking in which our method takes the fourth place among current submissions. In the following, we show results for the Tsukuba and Venus test sets that are used in the Middlebury benchmark. Furthermore, we present results for the more complex Teddy and Cones stereo images that were taken from Scharstein and Szeliski [79]. Finally, we show results for a self-recorded test set.

As a first image pair we present the Teddy test set shown in Figures 7.7a and 7.7b. The corresponding ground truth is presented in Figure 7.7c. The Teddy image pair is challenging for stereo algorithms, since it has a complex scene structure, a large disparity range ($0 \cdots 64$ pixels) and untextured, as well as large occluded regions. We show the disparity estimates on the pixel level for the left and right images in Figures 7.7d and 7.7e. It can be seen that most occluded pixels (coloured blue in the colour version and black in the grey-level version of this thesis) are correctly identified in both images, although some visible pixels erroneously carry the occlusion label. This happens for pixels whose pixel dissimilarity is larger than λ_{occ}. Figures 7.7g and 7.7h show the corresponding layer assignments on the pixel level. As a consequence of the view consistency term, the assignments are consistent across views. The disparity map on the segment level, which also represents the final output of our algorithm, is presented in Figure 7.7f.

On the segment level, surfaces are represented by their planar model and therefore by a continuous-valued function, yielding subpixel-precision. Furthermore, occluded regions are filled in by meaningful disparity values as a consequence of the segmentation information and the smoothness term of the

[3]This conference paper has appeared after publication of the algorithm described in Chapter 5.

[4]http://www.middlebury.edu/stereo/

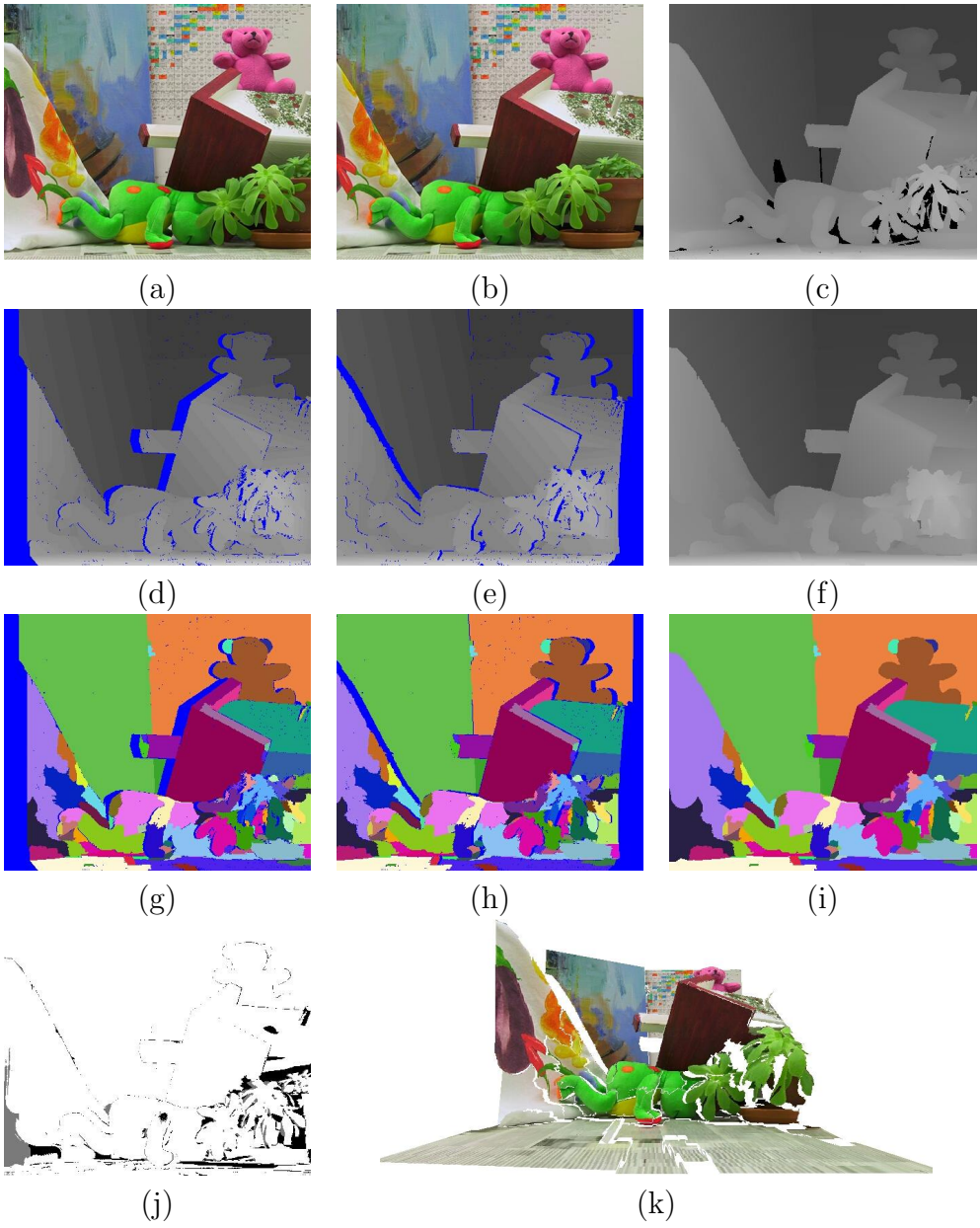


Figure 7.7: Results for the Teddy test set. (a,b) Left and right images. (c) Ground truth provided with image pair. (d,e) Disparity assignments for pixels of the left and right views. Pixels assigned to the occlusion label are coloured blue. (f) Disparity assignments for segments of the left view. (g,h) Disparity layer assignments for pixels of the left and right views. (i) Disparity layer assignments for segments of the left view. (j) Comparison of the disparity map (f) against the ground truth (c). (k) Reconstructed view.

cost function. The disparity layer assignments of segments are then presented in Figure 7.7i. These assignments are consistent with the assignments on the pixel level as a consequence of the segmentation term. We compare the computed disparity map against the ground truth in Figure 7.7j. We therefore plot pixels that have a disparity error larger than one pixel. Erroneous pixels in visible regions are coloured black and wrong pixels in occluded regions are assigned to grey. From this comparison against the ground truth it can be seen that our algorithm performs specifically well in the reconstruction of disparity discontinuities. This can be attributed to the incorporation of colour segmentation into our approach as well as to the accurate treatment of occlusions in both views. For quantitative evaluation, we compute two error percentages. First, we calculate the percentage of pixels exceeding an error threshold of one when considering *unoccluded* pixels only, which is also the error metric used in the Middlebury benchmark. Using this measurement, the error percentage is 4.77%. Second, we compute the error percentage for *all* pixels including occluded ones. The percentage of wrong pixels according to this metric is 6.77%. To give a further impression of the accuracy and detail of the computed disparities, we show a 3d-reconstruction in Figure 7.7k.

As a second test image pair, we use the well-known Tsukuba set. The results for this image pair are shown in Figure 7.8. Wrong disparity assignments for this image pair are mainly caused by segments that overlap a depth discontinuity (e.g. the tripod). Moreover, representing the head by two planar disparity layers oversimplifies the real surface. However, this could easily be improved by the use of a more sophisticated disparity model. The error percentage computed over all *unoccluded* pixels is 1.63%, while the percentage of *all* wrong pixels including occluded ones is 1.99%.

In Figure 7.9 we present additional results for standard test sets as well as for a self-recorded one. The corresponding right images and ground truth data for the standard images can be found on the Middlebury Stereo Vision website. The Venus test set along with computed results is presented in Figures 7.9(a1-a3). The algorithm correctly finds all five planes of which the scene consists. We point out that the newspaper at the right of Figure 7.9(a1) consists of two planes that are joined by a crease edge, which is also accurately reconstructed by the algorithm. The more complex Cones image pair and corresponding results are then shown in Figures 7.9(b1-b3). Wrong disparity values are mostly obtained in occluded regions. However, the scene is reconstructed quite accurately by a large number of disparity layers. Finally, we show a self-recorded stereo pair and the computed disparity map in Figures 7.9(c1-c3). The background of the scene is represented to a large extent by a single layer, whereas the disparity of the teddy, which has a more

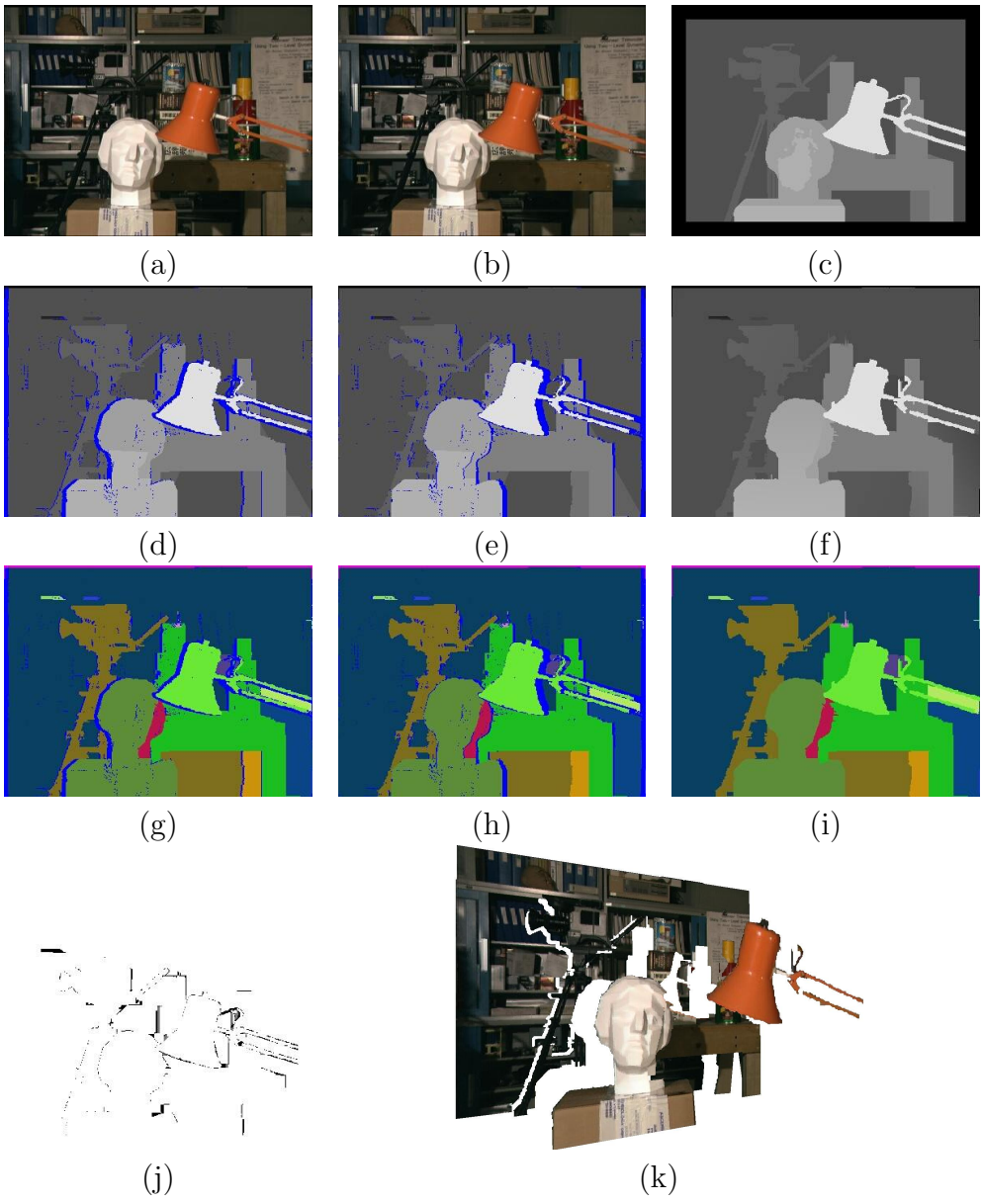


Figure 7.8: Results for the Tsukuba test set. (a,b) Left and right images. (c) Ground truth provided with image pair. (d,e) Disparity assignments for pixels of the left and right views. Pixels assigned to the occlusion label are coloured blue. (f) Disparity assignments for segments of the left view. (g,h) Disparity layer assignments for pixels of the left and right views. (i) Disparity layer assignments for segments of the left view. (j) Comparison of the disparity map (f) against the ground truth (c). (k) Reconstructed view.

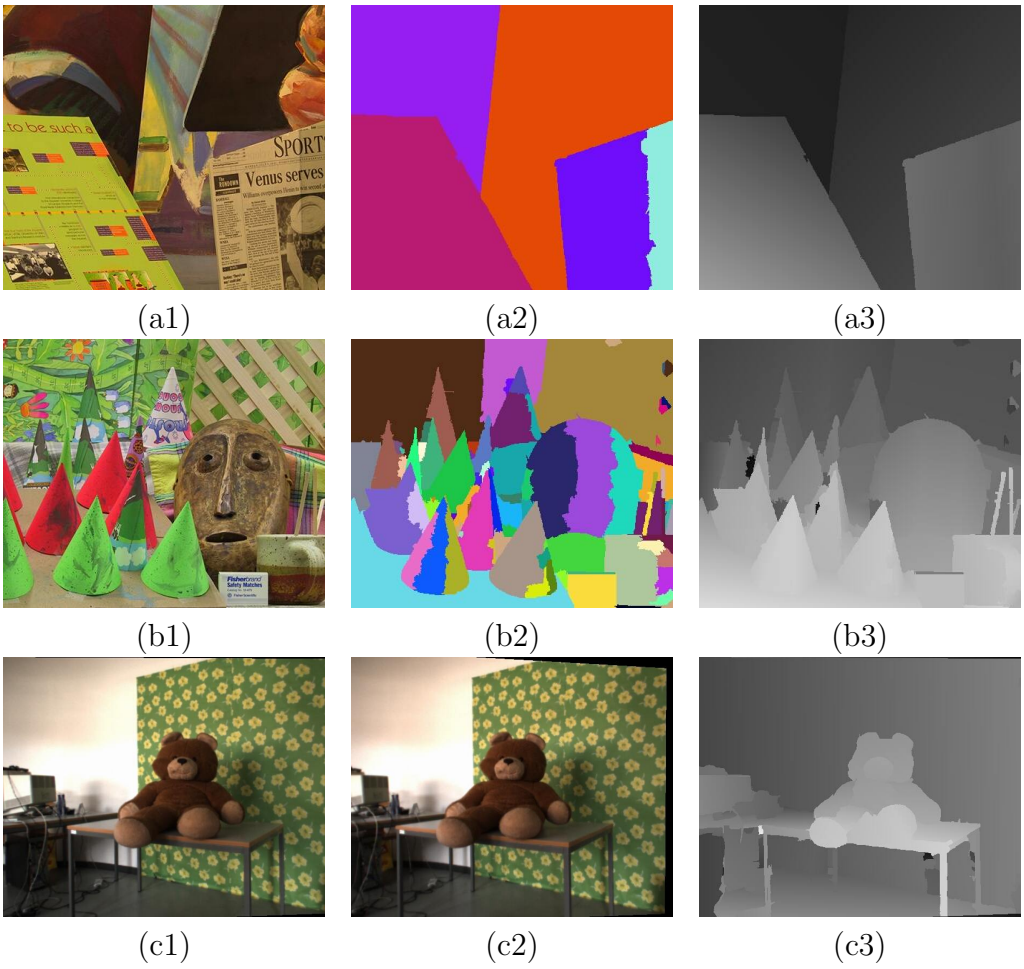


Figure 7.9: Results on standard and self-recorded image pairs. (a1) Left image of the Venus test set. (a2) Disparity layers. (a3) Disparity map. (b1) Left image of the Cones test set. (b2) Disparity layers. (b3) Disparity map. (c1) Left image of a self-recorded image set. (c2) Right image. (c3) Disparity map.

complex structure, is well reconstructed using more disparity layers. Furthermore, the algorithm was able to capture the thin structures represented by the legs of the table.

7.5 Summary

In this chapter, we have described a new method to solve the layer assignment problem. We have formulated this problem as a cost minimization task. Therefore, a global cost function has been designed that can be optimized via graph-cuts. Since partial occlusions of segments cannot be dealt with in the domain of segments, we have also included the pixel level into our cost function. We have modelled the segmentation assumption on the pixel level by enforcing that all pixels inside the same segment follow the same disparity model. However, to express the fact that there are occlusions, a pixel can as well be assigned to a special occlusion label. Moreover, we have included pixels of the second view into our cost function to allow for a symmetrical treatment of occlusions. We have shown how our cost function models the uniqueness constraint or, as we have seen later, some improved form of it. To approximate the global optimum of costs, we have used robust graph-cut optimization. Results obtained for the Middlebury test set and a self-recorded image pair have shown the good performance of the proposed method. Our algorithm performs specifically well in the accurate reconstruction of disparity boundaries.

Chapter 8

A graph-cut formulation for motion

8.1 Introduction

This chapter describes how the cost function of the previous chapter is embedded into an algorithm that computes the optical flow between two or more images. In contrast to the preceding chapter, we do not only focus on the layer assignment step, but also reinvestigate the layer extraction problem.

We translate the layer extraction task to the following question: Given a set of initial motion models, what are those models that represent the dominant image motion? Our idea is to as well formulate this task in terms of a global cost function, whose optimization can be effectively accomplished via graph-based optimization. This cost function can be regarded as a simplified version of that used in the previous chapter. Minimization of costs is computationally more efficient for this function, which is, however, at the price of ignoring occlusions. Occlusions are then dealt with in the layer assignment phase of the algorithm.

In the layer assignment step, we focus on extending the cost function of the preceding chapter to take more than two images as input. This makes sense from a practical point of view. When processing an image sequence, it is more than likely that one does not only have two images at hand, but rather a larger number of frames. Since the occluded regions are, in general, different in each of these frames, this allows us to gather additional information from non-occluded image parts.

In the end of this chapter, we show a concrete application example of our optical flow algorithm, namely motion segmentation. We thereby divide a complete image sequence into a set of homogeneously moving objects. Ex-

traction of moving objects from an image sequence is of specific importance in the context of new coding schemes such as MPEG-4 that encode a scene as the composition of multiple video objects, each, for example, compressed with different parameters.

8.2 A novel layer extraction step

In a first step, we proceed as in Chapter 6. That is, we apply colour segmentation to the reference image by using the algorithm of Christoudias et al. [23] (Figure 8.1b). We then compute a set of sparse correspondences with the KLT-tracker [81] and fit each segment to these correspondences using the method of Section 5.4. Our motion model is again the affine one, which is defined by

$$\begin{aligned} V_x(x,y) &= a_{x0} + a_{xx}x + a_{xy}y \\ V_y(x,y) &= a_{y0} + a_{yx}x + a_{yy}y \end{aligned} \tag{8.1}$$

with V_x and V_y being the x- and y-components of the flow vector.

In order to extract a small set of layers out of these initial motion segments, we have to answer the following question: How many layers are present in the sequence and what are their motion parameters? Since an object that undergoes homogeneous motion will in general not be identified as a single colour segment, it is clear that layers do not coincide with segments. This is especially true in strongly textured regions and when applying oversegmentation. Considering our initial motion models as potential layer candidates, a simple strategy is to locally select the motion model that gives the highest matching score for each segment. A motion model that was selected for at least one segment can then be declared as being a layer. Obviously, the resulting number of extracted layers would clearly exceed the correct number. However, we propose to use a very similar strategy that exploits the smoothness constraint in addition.

Initially, the set of layers $\mathcal{L}$ is built by all motion models found in the previous step.[1] To extract a small set of layers out of $\mathcal{L}$, we design a cost function that is defined in the domain of segments only. This function is denoted by $C'(\cdot)$ in order not to be confused with the cost function $C(\cdot)$ of the previous chapter. The cost function $C'(f)$ measures the quality of an assignment f of segments to motion layers and is in the form of

$$C'(f) = T'_{data}(f) + T'_{smooth}(f). \tag{8.2}$$

[1]For efficiency, when building $\mathcal{L}$, a motion model is only included, if the set does not already contain a very similar one.

The data term T'_{data} calculates how well f agrees with the input images and is defined by

$$T'_{data}(f) = \sum_{s \in \mathcal{S}} \sum_{p \in s} dis(p, m[f(s)](p)) \tag{8.3}$$

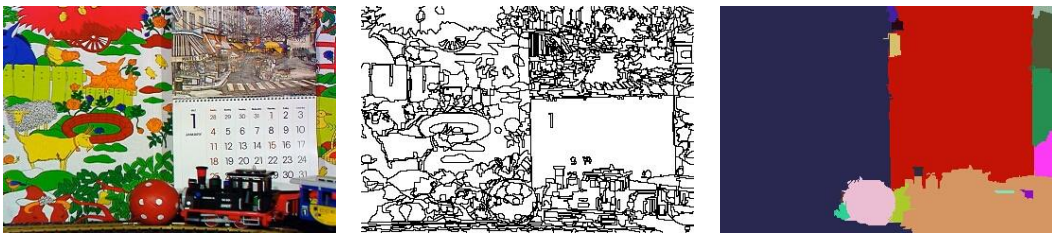
with S being the set of all segments of the reference view and $f(s)$ being the index of the layer to which segment s is assigned. We write $m[k](p)$ to denote the matching point of a pixel p in the other view according to the kth motion layer. More precisely, $m[k](p)$ is derived by computing the displacement vector at p using the affine parameters of the layer at index k (equation (8.1)) and adding it to the coordinates of p. The function $dis(\cdot, \cdot)$ computes the dissimilarity of two pixels, which is the sum-of-absolute-differences of RGB values in our implementation. The second term T'_{smooth} of the energy function measures to which extent the current assignment f is spatially smooth. T'_{smooth} is defined by

$$T'_{smooth}(f) = \sum_{(s_i, s_j) \in NB} \begin{cases} \lambda_{disc} \cdot bl(s_i, s_j) & : \quad f(s_i) \neq f(s_j) \\ 0 & : \quad \text{otherwise} \end{cases} \tag{8.4}$$

with NB being all pairs of neighbouring segments, $bl(\cdot, \cdot)$ computing the border length between such and λ_{disc} being a constant user-defined penalty.

We approximate the minimum of the energy function in equation (8.2) using the α-expansion algorithm of Boykov et al. [17]. Starting from an arbitrary configuration f, we iteratively change this configuration by computing the optimal α-expansion move for each layer until convergence. The graph built for calculating the optimal α-expansion consists of nodes that correspond to segments. Since the number of segments is significantly lower than the number of pixels, minimization of equation (8.2) via graph-cuts is very efficient.

Those layers that are not present in the newly computed configuration f^* are removed from the set of layers $\mathcal{L}$, which drastically decreases the number of layers. However, it is quite likely that the correct layer was not contained in our initial set, due to the small spatial extent over which the motion parameters were initially computed. We therefore refit the layers over their new spatial extents according to the assignment of segments to layers in f^* to derive a set of refined layers $\mathcal{L}'$. We then update $\mathcal{L}$ by $\mathcal{L} := \mathcal{L} \cup \mathcal{L}'$. Starting from the configuration f^*, we apply the α-expansion algorithm using our refined layer set $\mathcal{L}$ to obtain the new configuration f^{**}. We again remove those layers from $\mathcal{L}$ that do not occur in f^{**}. If the costs of f^{**} are not lower than those of f^*, $\mathcal{L}$ represents our final set of layers. Otherwise, this procedure is iterated.



(a) (b) (c)

Figure 8.1: Colour segmentation and layer extraction. (a) Original image. (b) Result of the colour segmentation step. Segment borders are shown. (c) Result of the layer extraction step. Pixels of the same colour belong to the same layer.

We show results of the layer extraction step in Figure 8.1c. Since the proposed algorithm operates on the segment level only, it is not capable of handling occlusions. It therefore produces artefacts in regions close to motion boundaries. Although there are only small occluded areas in the sequence shown in Figure 8.1 such artefacts are visible in the proximity of the rotating ball.[2] However, this strategy works well enough to deliver the dominant image motion and it is computationally efficient. Moreover, those wrong assignments can partially be identified, since they usually show large pixel dissimilarity and are relatively small. We therefore remove such layers to speed up the computationally more expensive assignment step.

8.3 Layer assignment with multiple input images

Knowing the set of layers, the task of the assignment step is to estimate which parts of the images are covered by which layers as well as to identify occlusions. For this step, we make use of our cost function $C(f)$ from equation (7.4) of the previous chapter. Integration of $C(f)$ into our motion algorithm is straightforward with the only difference being that a pixel's matching point is differently computed by using the affine motion instead of the planar model.

The cost function $C(f)$ has been originally designed to be used with only two input images. However, often frames in between these two images are available as well and can be used to improve the matching results. Let I_1

[2]We will present an example where this effect is more severe in the experimental results.

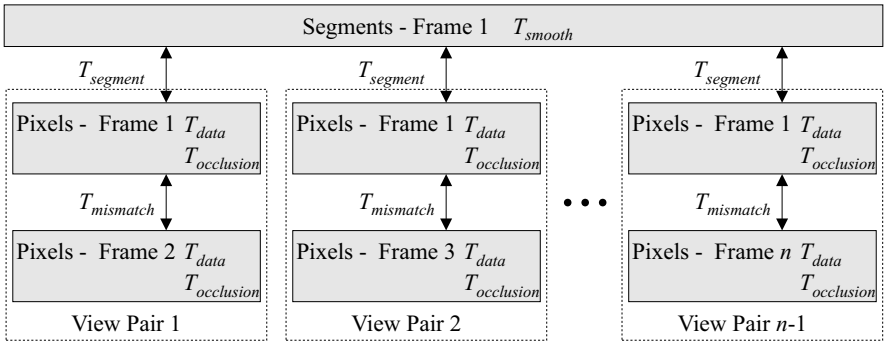


Figure 8.2: Conceptual view of the cost function $C(f)$ when using more than two input images.

and I_n be the first and last views of a short video clip of n frames. For computing the optical flow between I_1 and I_n, we do not only match I_1 against I_n, but also match I_1 against any intermediate view I_k with $1 < k < n$. The basic idea behind this is that a pixel of the reference frame I_1, which is occluded when matching I_1 and I_n, might be visible (and therefore matchable) when computing the correspondences between I_1 and I_k. This concept was originally used by Xiao and Shah [101, 100].

To implement this idea, we split up a video sequence of n images into $n-1$ view pairs. Each view pair thereby consists of the reference frame I_1, on which we apply colour segmentation, and a second image $I_k \neq I_1$, i.e. we derive the view pairs $I_1 - I_2$, $I_1 - I_3, \cdots, I_1 - I_n$. From the layer extraction step, we have the dominant motion models of the view pair $I_1 - I_n$. For simplicity, we assume that within a very short image sequence the motion is linear, so that the motion models for the other view pairs can be linearly interpolated from those. To propagate the layer assignments of the individual view pairs between each other, we connect the reference frame I_1 of each view pair to the segment level using the term $T_{segment}$ (Figure 8.2). From its definition in equation (7.7), $T_{segment}$ enforces a pixel of the reference view to have the same layer assignment as its corresponding segment, unless the pixel is occluded. Since the reference frames of all view pairs are now connected to the segment level, a pixel p of I_1 in view pair VP that is assigned to layer l has to be assigned to l in any other view pair VP' or carry the occlusion label. Otherwise, the segmentation term generates infinite costs. This constraint is what Xiao and Shah refer to as the *General Occlusion Constraint* [100], which is integrated into our energy function without any additional effort.

Approximation of the minimum of $C(f)$ is accomplished using the α-expansion algorithm as explained in Section 7.3. After the layer assignment

step the algorithm terminates and delivers the computed assignments of segments to motion layers as an output.

8.4 Experimental results

We have tested our algorithm on two standard data sets (Figure 8.3 and Figure 8.4) as well as on a self-recorded one (Figure 8.5). Since ground truth data is not available for any of those sequences, our analysis is limited to a qualitative evaluation of the results. The algorithm uses three user-defined parameters, which are λ_{occ}, $\lambda_{mismatch}$ and λ_{disc}. Throughout our test runs, we set $\lambda_{occ} := \lambda_{mismatch} - 1$. The effect of this is that every view inconsistent pixel is labelled as being occluded on the pixel level so that the uniqueness constraint is enforced (see Section 7.2.4). The remaining two parameters were optimized for good performance. Nevertheless, as a consequence of the robust optimization scheme, the sensitivity of the algorithm against variation of parameter settings is not very high.

As a first test sequence, we have picked five frames from the *Mobile & Calendar* sequence that is often used in the evaluation of optical flow and motion segmentation algorithms. Three of those images are shown in Figure 8.3a. Within this short sequence, there is translational motion on the train and the poster, while rotational motion originates from the ball that is pushed by the train. Moreover, the camera zooms out of the scene and pans to the left. To present the computed flow values on the segment level, we plot their absolute x- and y-components scaled by a factor of 32 in Figure 8.3b. The final assignment of segments to layers, which can be interpreted as a motion segmentation of the sequence, is then shown in Figure 8.3c. We draw the flow vectors for some pixels in Figure 8.3d. To allow for an easier interpretation of this result, we also outline the layer boundaries. We superimpose the layer borders on the reference image in Figure 8.3e to show their agreement with actual object boundaries. The object outlines seem to be well preserved. Finally, we present the results on the pixel level in Figure 8.3f with occlusions marked in red colour. Although occluded pixels seem to be correctly identified, some non-occluded pixels are as well assigned to the occlusion label. This happens for pixels whose dissimilarity is larger than λ_{occ}. The overall time needed to compute the results using five input frames of 352×240 pixels was approximately 10 minutes on an Intel Pentium 4 2.0 GHz computer with 30 seconds spent on the layer extraction and most of the remaining time used for the assignment step.

The next test set that we applied our algorithm on is the *Tennis* sequence shown in Figure 8.4a. There are two moving objects in the scene, which are

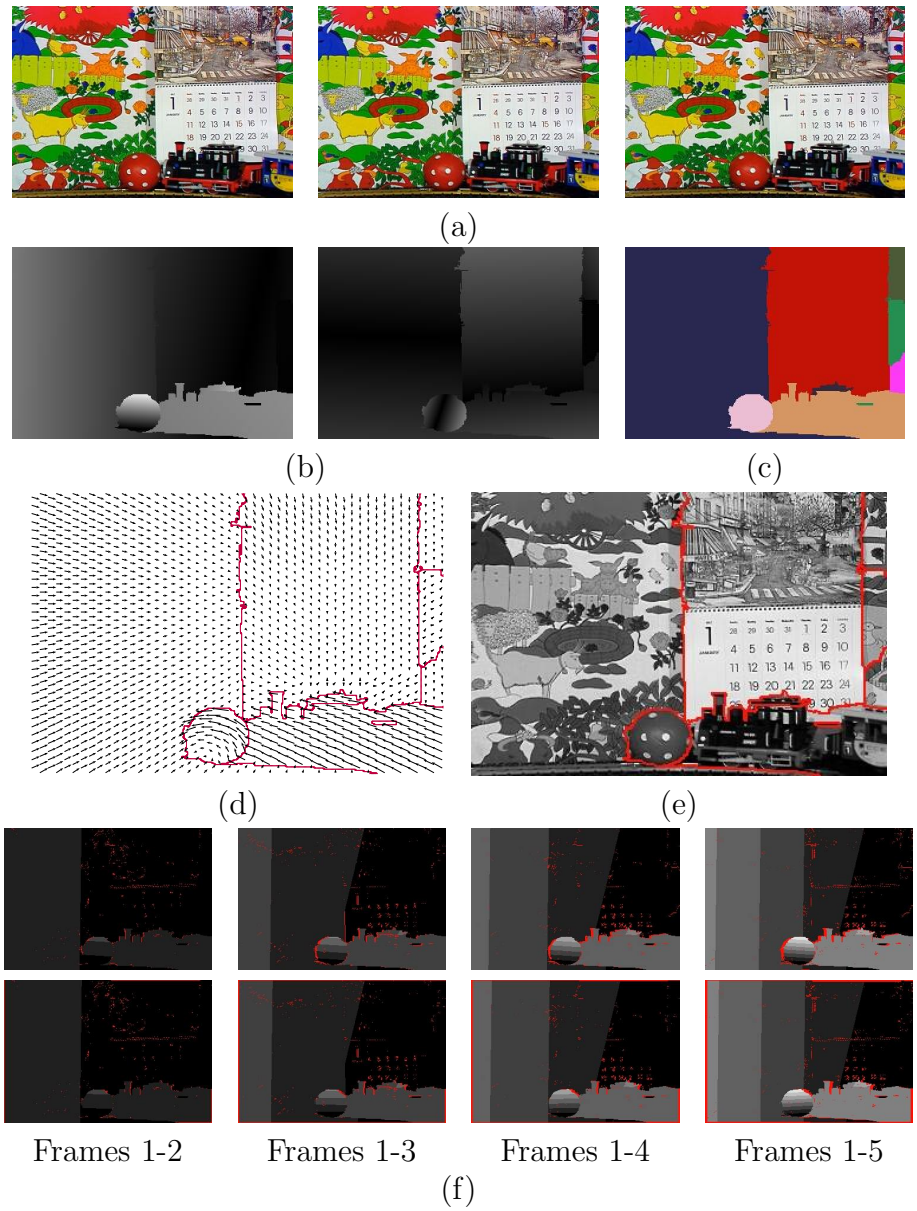


Figure 8.3: Results for the *Mobile & Calendar* sequence. (a) Frames 1, 3 and 5 of five input frames. (b) Absolute x- and y-components of the computed flow vectors. (c) Assignment of segments to layers. (d) Flow vectors with layer boundaries outlined. (e) Layer boundaries coloured in red (see electronic version) superimposed on input frame 1. (f) Absolute x-components of the flow vectors on the pixel level. The top row shows the reference view (frame 1), while the match images (frames 2 – 5) are presented at the bottom. Pixels carrying the occlusion label are coloured in red.

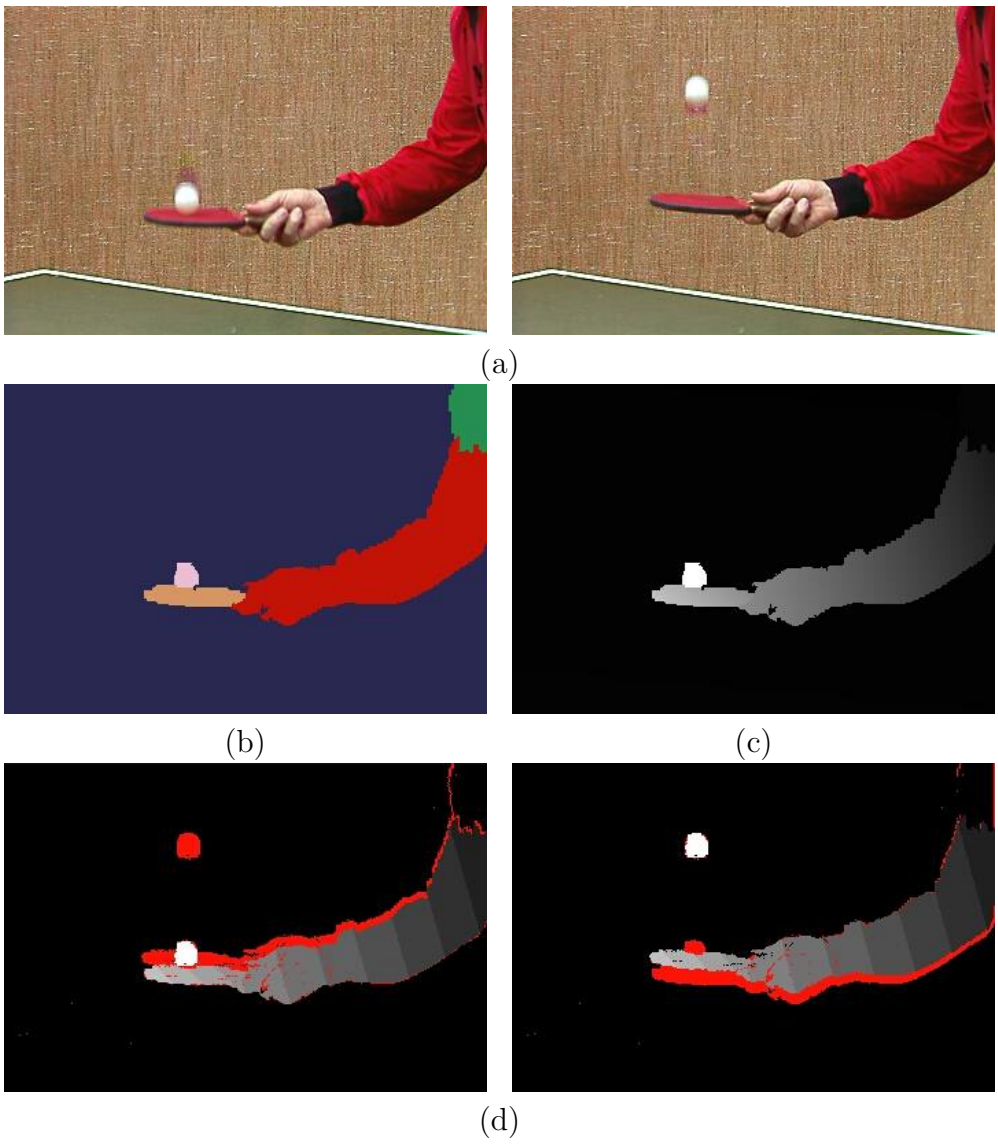


Figure 8.4: Results for the *Tennis* sequence. (a) The two input frames. (b) Assignments of segments to layers. (c) Absolute y-components of the computed flow vectors on the segment level. (d) Absolute y-components on the pixel level. Red pixels denote detected occlusions.

the arm and the ball. While the arm undergoes a relatively small motion, there is large motion on the ball. Moreover, the ball is affected by motion blur. We use only two input images, since the motion of this ball is not linear. The computed layer assignments are shown in Figure 8.4b. Figure 8.4c then shows the y-components on the segment level on which occlusions are automatically filled in. The values are scaled by a factor of 16. Finally, we present the y-components of the flow vectors on the pixel level in Figure 8.4d to show that the large occluded areas are correctly detected. The algorithm needed 4 minutes to generate these results with the image sizes being 352×240 pixels.

In addition to the standard test sets, we tested the proposed method on a self-recorded sequence. As input for our algorithm, we used three consecutive frames of which two are shown in Figure 8.5a. In this sequence, a train is moving from right to left in front of a static background. Although the motion is relatively simple, the scene contains complex motion boundaries (e.g. the link connecting the wagons) and relatively large occluded areas. These occlusions are the reason why the layer extraction step delivers poor results in the proximity of the motion discontinuities as shown in Figure 8.5b. In contrast to this, the assignment step that explicitly models occlusions seems to be able to outline the motion boundaries correctly, which we demonstrate in Figure 8.5c. Due to the lens distortion of the camera used in recording this sequence, the motion of the train cannot be described by a single affine motion model. The train is therefore modelled by two different layers. We then present the x-components of the flow vectors scaled by a factor of 16 in Figure 8.5d and show the layer borders superimposed on the reference frame in 8.5e. Given the three images of 546×318 pixels as input, the algorithm terminated after 6 minutes.

Finally, we used our algorithm to segment a complete video sequence into homogeneously moving objects. In the first step, we pick the frames 1–5 of the Mobile & Calendar sequence to compute the results that were shown in Figure 8.3. The final assignment of segments to layers thereby represents our motion segmentation result for frame 1. The layers that are present in the final solution then build the input for computing the segmentation of frame 2. More precisely, when segmenting frame 2 using the images 2–6, we do not invoke the layer extraction step, but directly assign the layers derived from the segmentation of frame 1. Since the layers' motion models will change over time, we refit each layer to correspondences derived from the KLT-tracker after each assignment step and then pass these layers to segment the next frame of the sequence. We show the computed segmentation results for each fifth frame of the Mobile & Calendar sequence in Figure 8.6. The algorithm's parameters were kept constant throughout the computation.

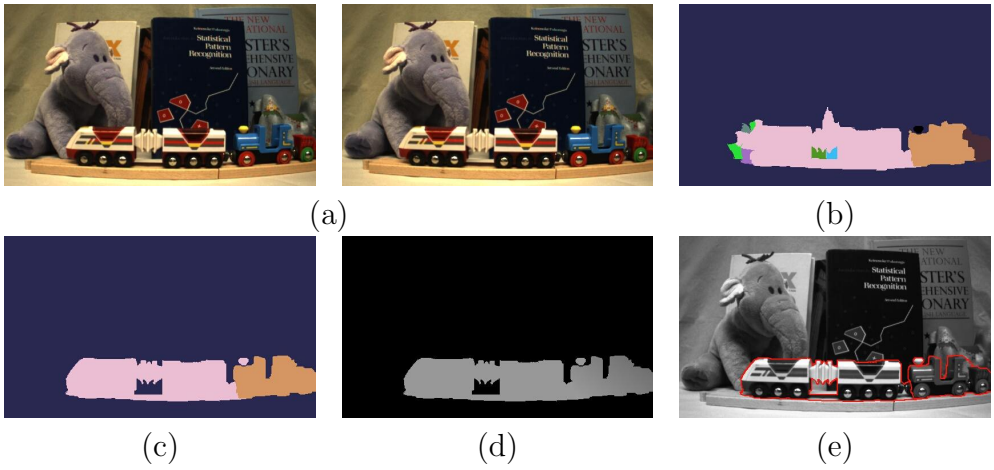


Figure 8.5: Results for a self-recorded sequence. (a) Frames 1 and 3 of three input frames. (b) Results of the layer extraction step. (c) Assignments of segments to layers. (d) Absolute x-components of flow vectors. (e) Layer boundaries superimposed on view 1.

The results of this motion segmentation are useful in the following sense. By segmentation of a single frame based on motion, we can identify those regions of the image that move homogeneously. Since a region of homogeneous motion usually originates from a single real-world object in motion, this often represents a semantic segmentation (e.g. the motion layer representing the ball in Figure 8.6). Moreover, by passing the layers as input for the segmentation of the subsequent frame, we can track this semantic object throughout the video sequence. Our algorithm can therefore be used to automatically extract a set of so-called video objects from an image sequence. We show the video objects extracted from the Mobile & Calendar sequence in Figure 8.7. As an application example, we use these video objects to manipulate the Mobile & Calendar sequence. This is shown in Figure 8.8. In this video, we reinsert the extracted video object representing the ball into the scene using different motion parameters. The ball thereby bounces against the video object that corresponds to the train.

8.5 Summary

In this chapter, we have applied our graph-cut formulation to the task of motion computation. First, we have presented a novel layer extraction step. To extract a small set of layers, we have minimized a simple cost function that

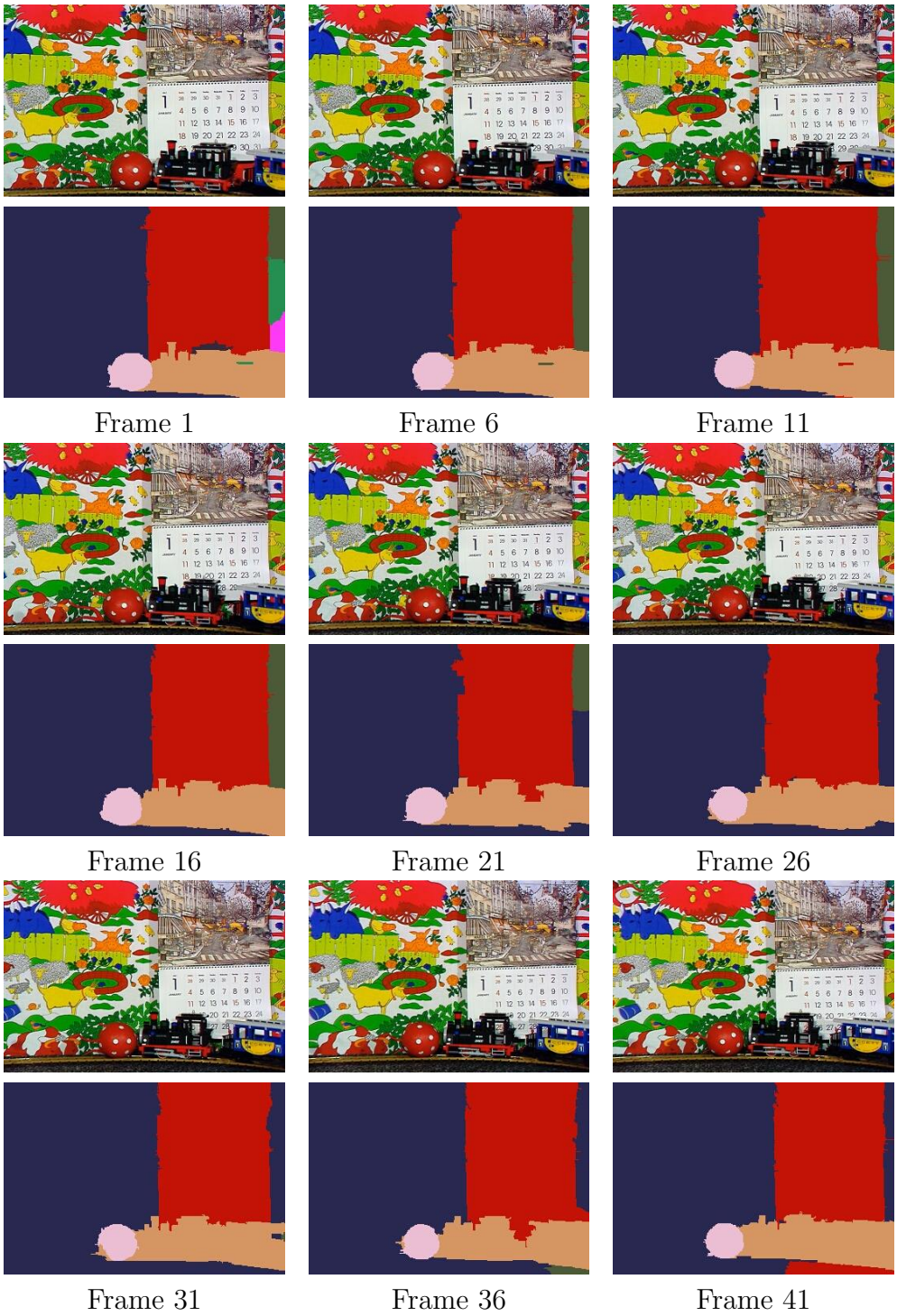


Figure 8.6: Motion segmentation.

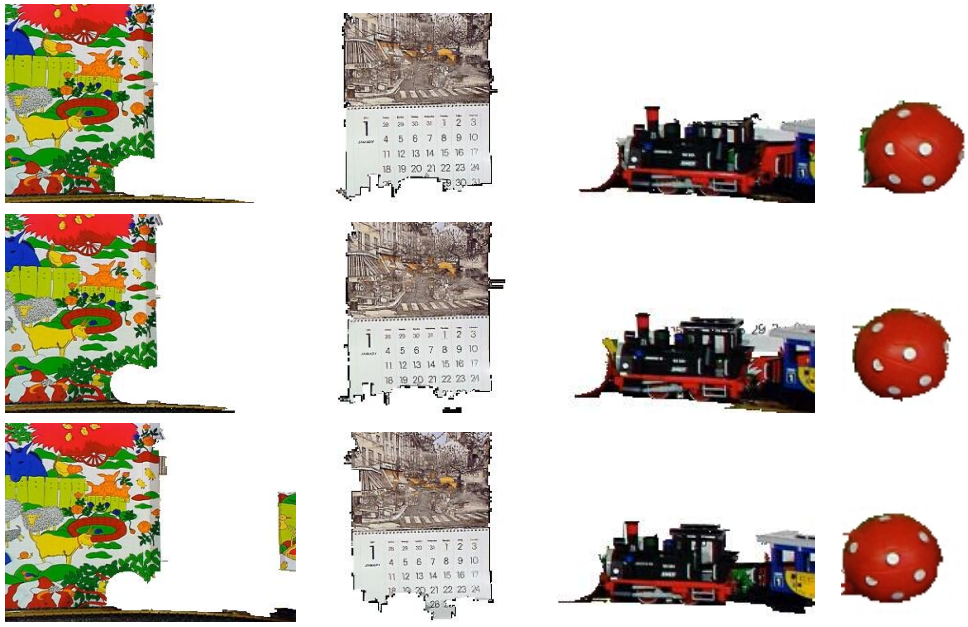


Figure 8.7: Automatically extracted video objects in frames 1, 11 and 21.

Figure 8.8: Video object insertion. The order of the presented images is from top left to bottom right.

models the optical flow problem on the segment level only. The advantage of this cost function is that its optimization via graph-cuts is computationally very efficient. We can therefore check for a large number of different motion models whether they represent a layer. However, the cost function does not account for occlusions. To overcome this problem, we have used the cost function of the previous chapter in the layer assignment step of our approach. This cost function handles occlusions by incorporation of the pixel level and its integration into our motion algorithm is straightforward. When dealing with video sequences, one usually has more than two frames available. We have therefore described how our cost function is extended to operate on multiple input images. This is advantageous, since the occlusions between different image pairs are different. Minimization of the resulting cost function has then been accomplished using the α-expansion algorithm. In the experimental results, we have not only applied our algorithm to the task of optical flow computation, but also used it for motion segmentation. We have demonstrated that our method is capable of successfully dividing a complete sequence of images into different video objects.

Chapter 9

Conclusions

9.1 Summary

This thesis has focused on the dense correspondence problem occurring in the field of low-level computer vision. We have described two novel methods that tackle the problem using a segmentation-based approach. Our basic assumptions are that disparity varies smoothly inside a region of homogeneous colour, while disparity discontinuities are located at the borders of those regions. The purpose of applying this segmentation assumption is to improve the performance of our algorithms in untextured regions and in the proximity of disparity boundaries.

We have presented two novel segmentation-based stereo methods of which both make use of a layered representation. The dominant disparity planes, which we refer to as layers, have been determined by mean-shift-based clustering. For optimal assignment of each segment of the reference view to exactly one of the extracted disparity layers and to detect occlusions, we have then set up a global cost function.

In our first method, this cost function uses image warping to compute the quality of an assignment of segments to disparity layers. The warped reference image is thereby compared against the real second view by computing the pixel dissimilarities. Moreover, occlusions are detected in the warping process, and reasoning about a pixel's visibility is accomplished using a Z-buffer. Since the resulting cost function is known to be $\mathcal{NP}$-complete, exact minimization of costs is not possible. We have therefore employed an efficient greedy search strategy in order to determine a local optimum. We have further shown that the algorithm can be extended to optical flow computation. The major difference lies in that the visibility reasoning has to be modified.

In our second approach, we have constructed a cost function in a way

that it can be optimized via graph-cuts. This cost function operates on two levels. While the first level corresponds to the segments of the reference view, the second level represents the pixels of this image. This two-level representation is motivated by the fact that partial occlusions of segments cannot be expressed when modelling the problem on the segment level only. For a symmetrical treatment of occlusions, we have as well included all pixels of the second view into our problem formulation. Occlusions in both images are then detected by enforcing the uniqueness constraint. In order to approximate the global optimum of costs, we have employed the α-expansion framework. In a further step, we have applied this cost function to the optical flow problem. While this extension is straightforward, we have shown how the cost function can be used to compute the image motion by taking more than two views as an input. Moreover, we have introduced a new layer extraction procedure. The resulting algorithm has proven to be robust enough to divide a complete video sequence into homogeneously moving video objects.

In our experiments, we have demonstrated that our methods are capable of computing good results on standard images as well as on self-recorded ones. We have evaluated our stereo methods using the Middlebury data set for which ground truth is available. Our stereo algorithms are ranked among the top five of approximately 40 contributions in this benchmark. The methods show particularly good results in regions of poor texture and are capable of precisely outlining disparity and motion discontinuities.

9.2 Open topics and future research

Although the algorithms presented in this thesis have shown strong results, there are some open topics that we are planning to address in future research:

- The segmentation assumption is not guaranteed to hold true. This is probably the most severe limitation of our approaches, and our current remedy to this is to apply a strong oversegmentation. However, since this does not completely overcome this problem, our algorithms could, for example, take benefit from an operation that allows splitting segments. It would as well be interesting to develop a special purpose colour segmentation method that avoids (as far as possible) producing segments which overlap a depth discontinuity.

- The disparity model might be too simple to represent the actual surface. In our current implementation, we assume that surfaces are planar. If this is not the case, our algorithms tend to oversimplify the real shapes.

However, both algorithms can be modified to use a more elaborate surface model (e.g., a spline model). Especially for the second algorithm, this modification is straightforward.

- Image noise and other deviations from Lambertian surfaces have not been modelled in our methods. We leave this for further work.

- The algorithms' parameters are chosen empirically. In a more advanced implementation, parameter estimation could be automated (e.g., based on the expected level of image noise or disparity variation).

- Stereo and optical flow computation are considered as separate problems. In further research, we are planning to set up a stereo system that also exploits temporal relationships. This should improve the quality of stereo as well as of motion estimates.

Nevertheless, we have shown that colour segmentation combined with accurate treatment of occlusions can aid in the computation of dense stereo and motion estimates.

Appendix A

Supplements to the greedy method

A.1 Incremental image warping

From a computational point of view, warping the complete reference image according to the current layer assignment is a costly operation. Fortunately, this operation, which is called the base warp, only needs to be performed once for the initial solution. In the hypothesis testing phase usually only small parts of the warped image are changed. We therefore employ an incremental warping procedure that builds upon the base warp and only warps those segments to the second view that have a new assignment. In this process, we also incrementally calculate the costs of the formed solutions. For the implementation of the described hypothesis testing algorithm, we require two efficient basic operations. One operation serves to add a segment to the Z-buffer and the other is used to delete a segment from the Z-buffer. For each applied operation, we determine the resulting change of costs allowing an incremental computation of the current solution's costs.

To insert a segment into the Z-buffer, we apply the segment warping procedure described in Section 5.6.1. The coordinates in the second view and the colour values of the image points are retrieved and added to their corresponding Z-buffer cells. To calculate the change of costs in each individual Z-buffer cell occupied by the segment, we distinguish between three cases, as illustrated in Figure A.1. In the first case, a new entry is added to an empty cell. In the second case, the new entry is occluded by a pixel of the same cell having higher disparity, and in the third case, the new pixel occludes the pixel that was visible before insertion. Separating these three cases, the change of costs $\delta_{add}(p)$ implicated by adding pixel p to a Z-buffer

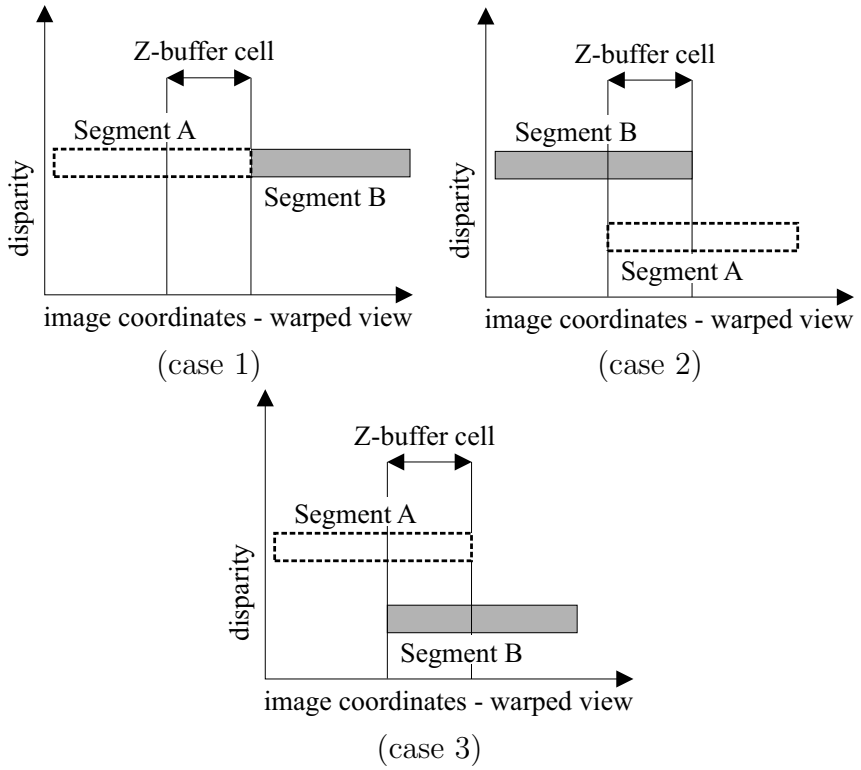


Figure A.1: Incremental computation of the costs in the Z-buffer. The insertion of Segment A implicates 3 different cases.

cell is computed by

$$\delta_{add}(p) = \begin{cases} dis(p) - \lambda_{occ} & : \text{ case 1} \\ \lambda_{occ} & : \text{ case 2} \\ dis(p) - dis(p_{vis}) + \lambda_{occ} & : \text{ case 3} \end{cases} \tag{A.1}$$

with $dis(p)$ being the colour dissimilarity of the pixel p in the real and in the warped view, λ_{occ} denoting the occlusion penalty and p_{vis} being the pixel that was visible before the insertion of p. The change of costs Δ_{add} introduced by adding the segment to the Z-buffer is then computed by summing up the individual changes of costs over all cells occupied by the segment. Additionally, the discontinuity penalty λ_{disc} is added for each pixel on the segment's border to a segment of a different layer assignment in the reference image. Since deleting a segment obviously represents the inverse operation, the change of costs Δ_{del} for deletion of a segment can be deduced analogously.

In hypothesis testing we first delete the current segment from the Z-buffer to test neighbouring layers' plane models. We record the resulting change of

costs Δ_{del}. We then replace the segment's planar model by the plane of a neighbouring layer and add it to the Z-buffer. The computed change of costs Δ_{add} is stored. We then use the delete function to remove the segment again. We test the hypotheses of all other neighbouring layers. Finally, we restore the Z-buffer to its original state by adding the segment using its old planar description. If there are neighbouring layers for which

$$\Delta_{del} + \Delta_{add} < 0 \tag{A.2}$$

we found assignments for this segment that give locally lower costs than the current one. In this case, we record the assignment that gave the minimum value for this term. After all segments have been tested, we replace the old assignment by the recorded one. This update also needs to be applied to the Z-buffer using the described delete and insert functions. In each iteration of the algorithm, usually only a fraction of segments will be assigned to a new layer. Especially, when the algorithm converges to a local optimum, the number of updated segments will be very small. The use of the incremental delete and add functions for the update procedure therefore provides a significant gain of efficiency over the computational expensive operation of a base warp.

A.2 Sensitivity of results to variations in parameter values

As for every global stereo matching method, the setting of parameters plays an important role. There are three parameters the user can tune to influence the algorithm's results: the mean-shift radius r, the occlusion penalty λ_{occ} and the discontinuity penalty λ_{disc}. All other parameters and thresholds are set to constant values, which are given in the main text of the paper.

We take a closer look at the effects of varying the parameters using the Teddy test set shown in Figure 5.16a and Figure 5.16b. We have chosen the Teddy test set, since it has the most complex scene structure of the presented stereo pairs and thus presents the most challenging reconstruction task of the selected stereo pairs. The disparity map shown in Figure 5.16e was generated using the following parameter values: $r = 0.6$, $\lambda_{occ} = 20.0$ and $\lambda_{disc} = 2.5$. For studying the role of a specific parameter, we generate results by varying its setting. The two other parameters are thereby kept fixed and set to the values given above. Each result is then compared against the ground truth by computing the percentage of *all* pixels having a disparity error larger than one. The resulting plots are shown in Figure A.2 and are interpreted as follows.

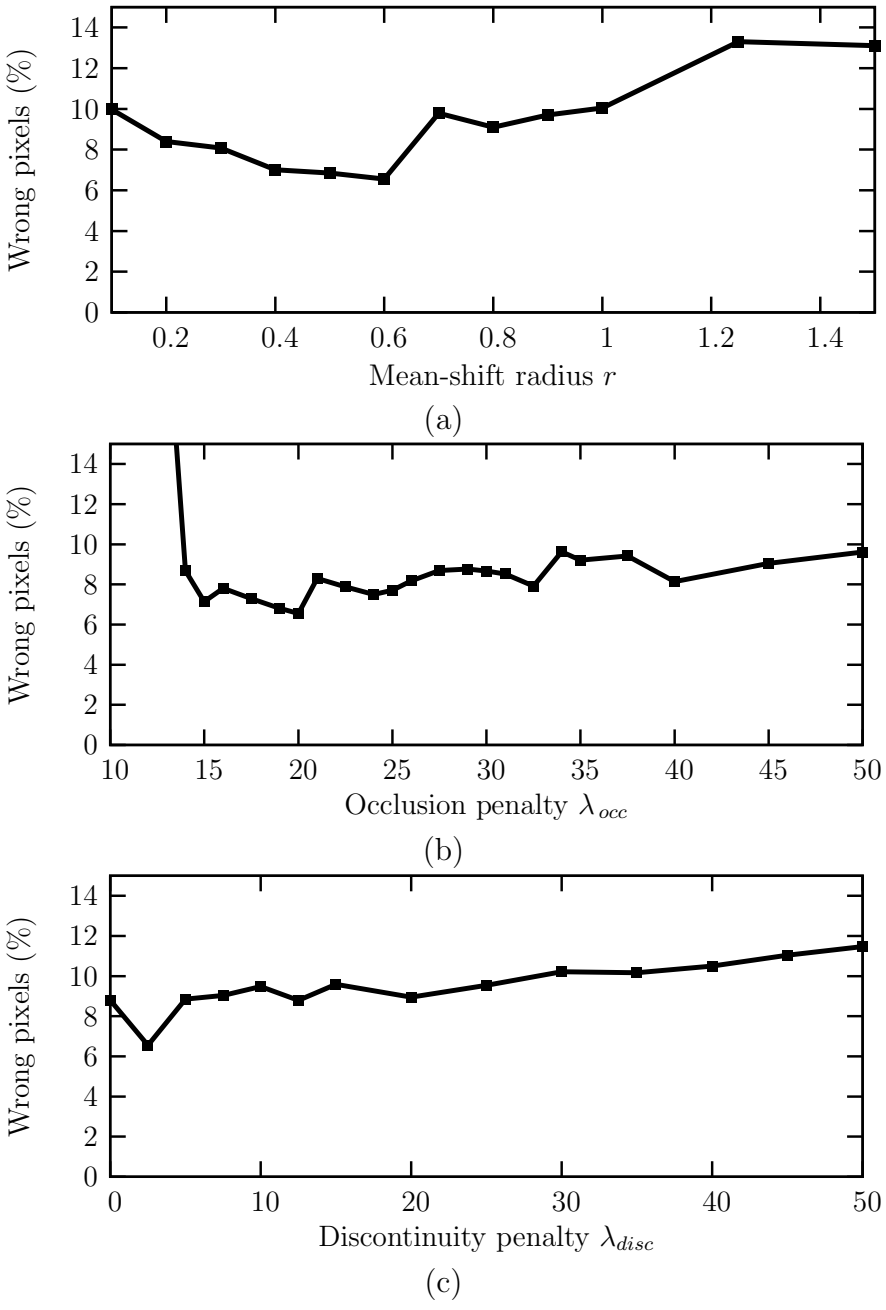


Figure A.2: Percentage of *all* wrong pixels for varying parameter values. (a) Different settings for the mean-shift radius r ($\lambda_{occ} = 20.0$, $\lambda_{disc} = 2.5$). (b) Different settings for the occlusion penalty λ_{occ} ($r = 0.6$, $\lambda_{disc} = 2.5$). (c) Different settings for the discontinuity penalty λ_{disc} ($r = 0.6$, $\lambda_{occ} = 20.0$).

The mean-shift radius r, whose plot is shown in Figure A.2a, controls the number of layers that are found in the layer extraction step of the algorithm. If r is set to low values, the number of extracted clusters and therefore layers will be high. In this case, the layers will not be very robust as a consequence of the small spatial extent over which their plane parameters were computed. On the other hand, setting the mean-shift radius to a high value causes two different surfaces to be represented by the same layer, which is not desirable either. For the Teddy test set, values in the range of $[0.5, \cdots, 0.6]$ represent a good trade-off between these two competing effects, as can be seen in the plot.

The plot for different settings of the occlusion penalty λ_{occ} is shown in Figure A.2b. If λ_{occ} is given a very low value, the algorithm tries to propagate planes that create occlusions in the warped view, which in general results in bad solutions. For $\lambda_{occ} = 10$, we receive 38.1% of wrong pixels on the Teddy test set. On the other hand, overpenalizing occlusions usually decreases the performance in segments close to depth boundaries, since the algorithm then tries to generate continuous disparity transitions instead of modelling jumps in disparity that go along with occlusions. In this example, however, the results are not very sensitive to the occlusion penalty as long as it is not too low.

Finally, we show the plot for different settings of the discontinuity penalty λ_{disc} in Figure A.2c. The plotted results show a minimum value at $\lambda_{disc} = 2.5$ and relatively small variations over the rest of the displayed parameter range. A slight increase of the error rate with larger values of λ_{disc} can be attributed to the large number of layers that is needed to accurately represent the scene. As a consequence, the boundary lengths between different layers are relatively large (e.g., the boundaries between the different plants in Figure 5.16), which is penalized by λ_{disc}. Assigning large values to the discontinuity penalty λ_{disc} therefore decreases the performance. Nevertheless, λ_{disc} significantly contributes to the reconstruction of scenes consisting of large planar surfaces as the Venus test set shown in Figure 5.14a and Figure 5.14b.

Appendix B

Supplements to the graph-cut method

B.1 Graph construction

In the following, we show that our cost function is representable by a set of functions of binary variables that fulfil condition (7.12) and build the overall graph. The functions that will be used in the construction as well as the corresponding graphs that implement those functions are presented in Figure B.1. The graph constructions are with small deviations those presented by Kolmogorov and Zabih [52].

Let us, for example, consider construction (b4) of Figure B.1. In construction (b4), the function $C^{i,j}(x_i, x_j)$ returns the non-negative constant c for all settings of x_i and x_j, except for $x_i = 1$ and $x_j = 1$ for which it returns 0. $C^{i,j}(x_i, x_j)$ obviously fulfils condition (7.12), since c is non-negative and therefore

$$C^{i,j}(0,0) + C^{i,j}(1,1) = c + 0 \leq c + c = C^{i,j}(0,1) + C^{i,j}(1,0). \tag{B.1}$$

In Figure B.2 we show that for any cut in the graph of construction (b4), the costs of this cut are equal to the costs of the corresponding binary labelling as specified by the function $C^{i,j}(x_i, x_j)$. In the first case (Figure B.2a), we examine the labelling $x_i = 0$ and $x_j = 0$ with $C^{i,j}(0,0) = c$. According to equation (7.11), this labelling corresponds to the cut $v_i \in SRC$ and $v_j \in SRC$. In this configuration, there is exactly one edge in the graph that goes from SRC to SNK, namely the edge (v_j, snk). The costs of the cut are therefore equal to the weight on the edge (v_j, snk), which is c. In the second case (Figure B.2b), the labelling $x_i = 0$ and $x_j = 1$ with $C^{i,j}(0,1) = c$ corresponds to the cut $v_i \in SRC$ and $v_j \in SNK$. In this configuration, the

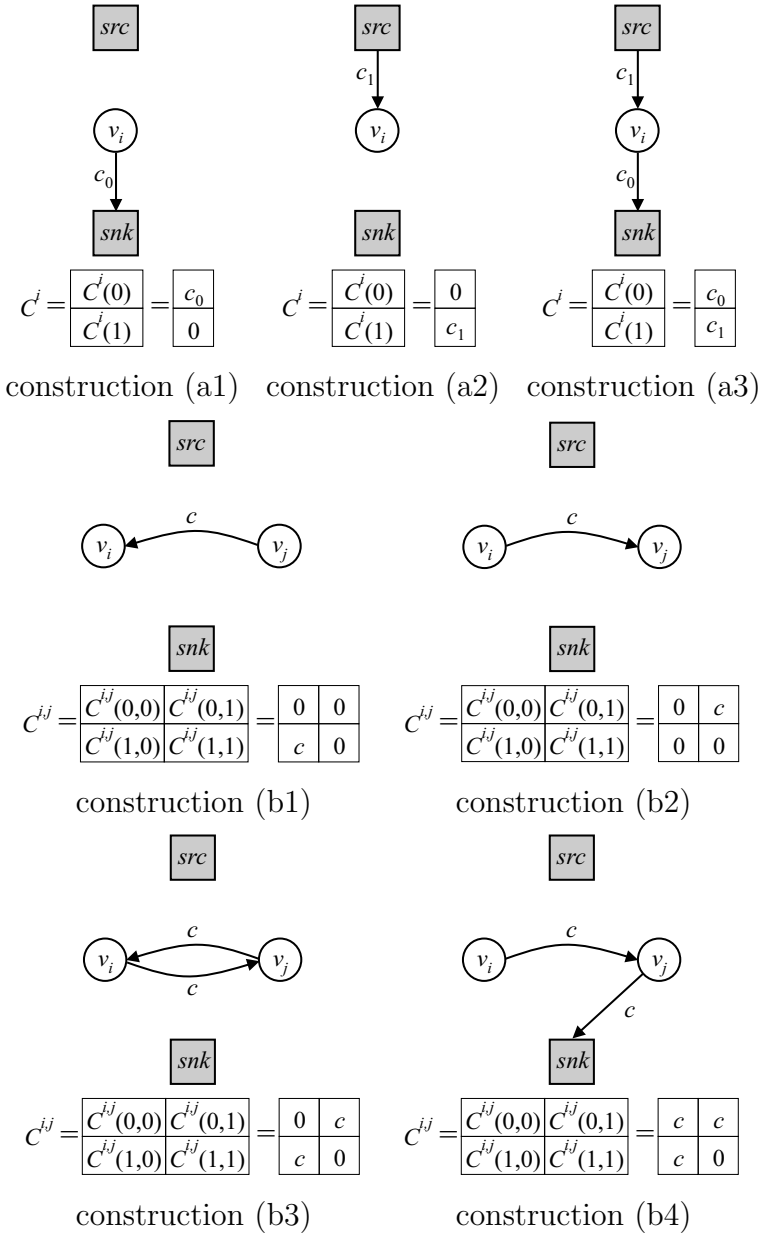


Figure B.1: Functions of binary variables and corresponding graphs used in the construction. Constructions (a1-a3) Functions in the form of $C^i(x_i)$ that only depend on one binary variable x_i. Constructions (b1-b4) Functions in the form of $C^{i,j}(x_i, x_j)$ that depend on two binary variables x_i and x_j. The constant c is non-negative.

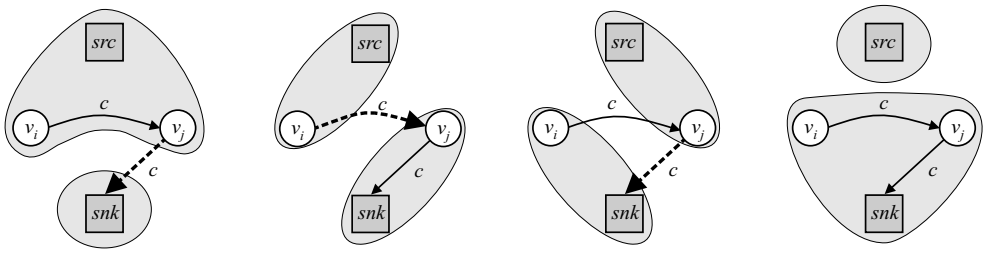


Figure B.2: Costs produced by different cuts in graph of construction (b4) of Figure B.1. Dashed edges generate costs. (a) $C^{i,j}(0,0) = c$. (b) $C^{i,j}(0,1) = c$. (c) $C^{i,j}(1,0) = c$. (d) $C^{i,j}(1,1) = 0$.

edge (v_i, v_j) that goes from SRC to SNK produces costs of c. In the third case (Figure B.2c), the binary labelling is given by $x_i = 1$ and $x_j = 0$ with $C^{i,j}(1,0) = c$ so that the graph is cut by $v_i \in SNK$ and $v_j \in SRC$. The only edge that goes from SRC to SNK is (v_j, snk) and therefore the cut generates costs of c. Finally, in the last case (Figure B.2d), we consider the labelling $x_i = 1$ and $x_j = 1$ with $C^{i,j}(1,1) = 0$, which corresponds to the cut $v_i \in SNK$ and $v_j \in SNK$. In this configuration, there is no edge that goes from SRC to SNK and therefore the costs of the cut are 0. The correctness of the other constructions of Figure B.2 can be shown analogously.

In the following, we use the constructions of Figure B.1 to represent the individual term of our cost function.

Data and occlusion terms

For each pixel p of both views, the data term defined in equation (7.5) measures the pixel dissimilarity, if p is not occluded ($f(p) \neq 0$). Otherwise, if p carries the occlusion label ($f(p) = 0$), the occlusion term of equation (7.6) generates costs of λ_{occ}. Let us suppose that a pixel p remains assigned to its old label in f' ($x_p = 0$). The costs defined by the data and occlusion terms are then computed by $C^p(0) = dis(p, m[f(p)](p))$ for $f(p) \neq 0$ and $C^p(0) = \lambda_{occ}$ for $f(p) = 0$. Analogously, the costs for assigning the pixel p to the label α in f' ($x_p = 1$) are derived by $C^p(1) = dis(p, m[\alpha](p))$ for $\alpha \neq 0$ and $C^p(1) = \lambda_{occ}$ for $\alpha = 0$. To implement the function $C^p(x_p)$ at each pixel p of both views, we apply construction (a3).

Segmentation term

The segmentation term defined in equation (7.7) penalizes each non-occluded pixel p ($f(p) \neq 0$) of the left view by an infinite penalty, if it carries a different label than the segment s to which it belongs ($f(p) \neq f(s)$). In the following, we analyse the cases that can arise in the construction of the segmentation term at each pixel p of the left view. We can thereby rely on the fact that our current label configuration fulfils the segmentation constraint, since segmentation inconsistent assignments are not contained in the initial configuration nor created in the optimization process.

Let us suppose that $f(p) \neq 0$ and $\alpha \neq 0$. Due to the validity of the segmentation constraint, we then know that the pixel p and the segment s both carry the same label in f. The case of $f(p) = \alpha$ is trivial, since for any combination of x_p and x_s no costs are generated. Let us consider the case of $f(p) \neq \alpha$. A configuration f' within one α-expansion from f that fulfils the segmentation constraint is then derived if either p as well as s keep their old labels in f' ($x_p = x_s = 0$) or both change their labels to α in f' ($x_p = x_s = 1$). Therefore, in this case, $C^{p,s}(0,0) = C^{p,s}(1,1) = 0$ and $C^{p,s}(0,1) = C^{p,s}(1,0) = \infty$, which is implemented by construction (b3).

Let us now consider the case of $f(p) \neq 0$ and $\alpha = 0$. In this configuration, the pixel p is allowed to change its label to the occlusion label in f' ($x_p = 1$), while the segment s keeps its original label in f' ($x_s = 0$). Furthermore, we allow that p and s both keep their old labels ($x_p = x_s = 0$) or both change their assignments to the occlusion label ($x_p = x_s = 1$). However, p must not remain assigned to its old label ($x_p = 0$), if the assignment of s is changed to the occlusion label ($x_s = 1$), since this would violate the segmentation constraint. We therefore derive the function $C^{p,s}(x_p, x_s)$ defined by $C^{p,s}(0,1) = \infty$ and $C^{p,s}(0,0) = C^{p,s}(1,0) = C^{p,s}(1,1) = 0$ and apply construction (b2). The case of $f(p) = 0$ and $\alpha \neq 0$ is constructed analogously by disallowing p to change its label to α ($x_p = 1$), while s keeps its original label ($x_s = 0$), which is implemented by construction (b1). Finally, the case of $f(p) = 0$ and $\alpha = 0$ is trivial, since any setting of x_p and x_s is allowed.

View consistency term

The view consistency term defined in equation (7.8) imposes the non-negative constant mismatch penalty $\lambda_{mismatch}$ for each visible pixel p ($f(p) \neq 0$) that is view inconsistent so that $f(p) \neq f(m[f(p)](p))$. In the construction of the view consistency term at each pixel p of both views, we analyse a set of different cases separately.

Let us suppose that the pixel's p old assignment is equal to α ($f(p) = \alpha$) and α is not the occlusion label ($\alpha \neq 0$). The matching point $q = m[\alpha](p)$ is therefore the same for both settings of the binary variable x_p. If also $f(q) = \alpha$, any setting of x_p and x_q results in a view consistent labelling and does not produce costs. Otherwise, if $f(q) \neq \alpha$, assigning q to its old label in f' ($x_q = 0$) needs to be penalized by $\lambda_{mismatch}$. We then derive the function $C^{p,q}(x_p, x_q)$ defined by $C^{p,q}(0,0) = C^{p,q}(1,0) = \lambda_{mismatch}$ and $C^{p,q}(0,1) = C^{p,q}(1,1) = 0$. Since the penalty is given independently of x_p, this is the same as the function $C^q(x_q)$ with $C^q(0) = \lambda_{mismatch}$ and $C^q(1) = 0$. Accordingly, we use construction (a1) to build the graph.

In the following, let us consider that the pixel's p old assignment is not equal to α ($f(p) \neq \alpha$). As a consequence, the pixel's p matching point q is in general different depending on the setting of x_p.[1] We will therefore apply two constructions on the pixel p, one for $x_p = 0$ and the other one for $x_p = 1$.

Let us start by assuming that $x_p = 1$ ($f'(p) = \alpha$), since this is the simpler one of the two constructions. If $\alpha = 0$, the pixel p is then occluded in f', which does not generate any costs. Otherwise, if $\alpha \neq 0$, the matching point $q = m[\alpha](p)$ becomes defined. Let us suppose that $f(q) = \alpha$. As a consequence, for any setting of x_q the matching point q will be assigned to the label α. The labellings of p and q are therefore view consistent and no costs are produced. Otherwise, if $f(q) \neq \alpha$, assigning q to its old label in f' ($x_q = 0$) results in a view inconsistent labelling. In this case, we derive the function $C^{p,q}(x_p, x_q)$ with $C^{p,q}(1,0) = \lambda_{mismatch}$ and $C^{p,q}(0,0) = C^{p,q}(0,1) = C^{p,q}(1,1) = 0$. Accordingly, construction (b1) is used.

Let us now suppose that $x_p = 0$ ($f'(p) = f(p)$). If $f(p) = 0$, the pixel p is occluded in f' and no costs are produced. Otherwise, if $f(p) \neq 0$, the matching point q is computed by $q = m[f(p)](p)$. If $f(p) = f(q)$, assigning q to its old label in f' ($x_q = 0$), results in a view consistent labelling. Furthermore, we know that $f(p) \neq \alpha$, since we filtered out the other case before. If therefore q changes its label to α in f' ($x_q = 1$), the configuration becomes view inconsistent. In this case, we derive $C^{p,q}(x_p, x_q)$ defined by $C^{p,q}(0,1) = \lambda_{mismatch}$ and $C^{p,q}(0,0) = C^{p,q}(1,0) = C^{p,q}(1,1) = 0$, which is implemented by construction (b2). Otherwise, if $f(p) \neq f(q)$, assigning q to its old label in f' ($x_q = 0$), results in a view inconsistent labelling. In addition, we know that $f(p) \neq \alpha$, so that changing the label of q to α in f' ($x_q = 1$) does not produce a consistent labelling either. Since in this case the costs of $\lambda_{mismatch}$ are given for any setting of x_q, we derive $C^p(x_p)$ defined by $C^p(0) = \lambda_{mismatch}$ and $C^p(1) = 0$. Accordingly, we apply construction (a1).

[1] However, the following construction also works, if the matching point q is the same for both settings of x_p ($m[f(p)](p) = m[\alpha](p)$).

Smoothness term

The smoothness term imposes a non-negative penalty for each pair of neighbouring segments (s,t) of the left view, if segment s is assigned to a different label than segment t ($f(s) \neq f(t)$). The weighted penalty is computed according to equation (7.9) and referred to by λ_{smooth} in the following.

We start our construction with the trivial case of $f(s) = f(t) = \alpha$, which does not generate costs for any setting of x_s and x_t. Let us now assume that $f(s) = \alpha$, but $f(t) \neq \alpha$. In this case, if t keeps its old label in f' ($x_t = 0$), the penalty λ_{smooth} is imposed. This defines the function $C^t(x_t)$ with $C^t(0) = \lambda_{smooth}$ and $C^t(1) = 0$, which is implemented according to construction (a1). The case of $f(s) \neq \alpha$ and $f(t) = \alpha$ is derived analogously.

In the following, let us assume that $f(s) \neq \alpha$ and $f(t) \neq \alpha$. We then have to analyse two cases separately. In the first case, we examine the configuration $f(s) = f(t)$. The segments s and t will then be assigned to the same label in f', only if either both keep their original labels ($x_s = x_t = 0$) or both change their labels to α ($x_s = x_t = 1$). We therefore derive the function $C^{s,t}(x_s, x_t)$ with $C^{s,t}(0,0) = C^{s,t}(1,1) = 0$ and $C^{s,t}(0,1) = C^{s,t}(1,0) = \lambda_{smooth}$, which is implemented by construction (b3). In the second case, if $f(s) \neq f(t)$, the segments s and t both have to change their labels to α in f' ($x_s = x_t = 1$) in order to have the same assignment. This defines the function $C^{s,t}(x_s, x_t)$ with $C^{s,t}(1,1) = 0$ and $C^{s,t}(0,0) = C^{s,t}(0,1) = C^{s,t}(1,0) = \lambda_{smooth}$. Accordingly, construction (b4) is applied.

Bibliography

[1] Intel open source computer vision library, beta 3.1. http://www.intel.com/research/mrl/research/opencv/.

[2] Point grey research inc. http://www.ptgrey.com/.

[3] M. Agrawal and L.S. Davis. Window-based, discontinuity preserving stereo. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 66–73, 2004.

[4] Y. Altunbasak, P.E. Eren, and A.M. Tekalp. Region-based parametric motion segmentation using color information. *Graphical Models and Image Processing*, 60(1):13–23, 1998.

[5] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, 1989.

[6] S. Ayer and H.S. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and mdl encoding. In *International Conference on Computer Vision*, pages 777–784, 1995.

[7] J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[8] J. Bigün, G.H. Granlund, and J. Wiklund. Multidimensional orientation estimation with applications to texture analysis and optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):775–790, 1991.

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

[10] S. Birchfield and C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. *International Journal of Computer Vision*, 35(3):269–293, 1999.

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

[12] M.J. Black. *Robust incremental optical flow*. PhD thesis, Yale University, 1992.

[13] M.J. Black and P. Anandan. Robust dynamic motion estimation over time. In *Conference on Computer Vision and Pattern Recognition*, pages 296–302, 1991.

[14] M.J. Black and A.D. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *Transactions on Pattern Analysis and Machine Intelligence*, 18(10):972–986, 1996.

[15] A. Bobick and S. Intille. Large occlusion stereo. *International Journal of Computer Vision*, 33(3):181–200, 1999.

[16] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[17] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[18] R. Brockers, M. Hund, and B. Mertsching. A fast cost relaxation stereo algorithm with occlusion detection for mobile robot applications. In *Vision, Modeling and Vizualisation Conference*, pages 47–53, 2004.

[19] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[20] C. Buehler, S. Gortler, M. Cohen, and L. McMillan. Minimal surfaces for stereo. In *European Conference on Computer Vision*, pages 885–899, 2002.

[21] P. Burt and B. Julesz. A gradient limit for binocular fusion. *Science*, 208:615–617, 1980.

[22] M.M. Chang, A.M. Tekalp, and M.I. Sezan. Simultaneous motion estimation and segmentation. *Transactions on Image Processing*, 6(9):1326–1333, 1997.

[23] C. Christoudias, B. Georgescu, and P. Meer. Synergism in low-level vision. In *International Conference on Pattern Recognition*, volume 4, pages 150–155, 2002.

[24] D. Comaniciu and P. Meer. Distribution free decomposition of multivariate data. *Pattern Analysis and Applications*, 1(2):22–30, 1999.

[25] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567, 1996.

[26] A. Criminisi, J. Shotton ans A. Blake, C. Rother, and P. H. S. Torr. Efficient dense-stereo with occlusions and new view synthesis by four state dp for gaze correction. *International Journal of Computer Vision*, 2005. submitted.

[27] Y. Deng, Q. Yang, X. Lin, and X. Tang. A symmetric patch-based correspondence model for occlusion handling. In *International Conference on Computer Vision*, pages 542–567, 2005. To appear.

[28] M. Etienne and P. Perez. Hierarchical estimation and segmentation of dense motion fields. *International Journal of Computer Vision*, 46(2):129–155, 2002.

[29] O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy. Real time correlation based stereo: algorithm implementations and applications. Technical report, RR-2013, INRIA, 1996.

[30] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[31] S. Forstmann, Y. Kanou, J. Ohya, S. Thuering, and A. Schmitt. Real-time stereo by using dynamic programming. In *Conference on Computer Vision and Pattern Recognition Workshop*, page 29, 2004.

[32] P.V. Fua. Combining stereo and monocular information to compute dense depth maps that preserve depth discontinuities. In *International Joint Conference on Artificial Intelligence*, pages 1292–1298, 1991.

[33] A. Fusiello, V. Roberto, and E. Trucco. Efficient stereo with multiple windowing. In *Conference on Computer Vision and Pattern Recognition*, pages 858–863, 1997.

[34] D. Geiger, B. Ladendorf, and A. Yuille. Occlusions and binocular stereo. *International Journal of Computer Vision*, 14:211–226, 1995.

[35] M. Gong and Y.H. Yang. Genetic-based stereo algorithm and disparity map evaluation. *International Journal of Computer Vision*, 47(1-3):63–77, 2002.

[36] M. Gong and Y.H. Yang. Near real-time reliable stereo matching using programmable graphics hardware. In *Conference on Computer Vision and Pattern Recognition*, pages 924–931, 2005.

[37] J. Y. Goulermas and P. Liatsis. A collective-based adaptive symbiotic model for surface reconstruction in area-based stereo. *IEEE Transactions on Evolutionary Computation*, 7(5):482–502, 2003.

[38] J. Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton University Bulletin*, 13:49–52, 1902.

[39] H. Hirschmüller, P. Innocent, and J. Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision*, 47:229–246, 2002.

[40] L. Hong and G. Chen. Segment-based stereo matching using graph cuts. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 74–81, 2004.

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

[42] H. Ishikawa. *Global Optimization Using Embedded Graphs.* PhD thesis, New York University, 2000.

[43] H. Ishikawa. Exact optimization for markov random fields with convex priors. *Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1333–1336, 2003.

[44] H. Ishikawa and D. Geiger. Occlusions, discontinuities, and epipolar lines in stereo. In *European Conference on Computer Vision*, pages 232–248, 1998.

[45] D.S. Kalivas and A.A. Sawchuk. A region matching motion estimation algorithm. *CVGIP: Image Understanding*, 54(2):275–288, 1991.

[46] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932, 1994.

[47] S.B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 103–110, 2001.

[48] Q. Ke and T. Kanade. A subspace approach to layer extraction. In *Conference on Computer Vision and Pattern Recognition*, pages 255–262, 2001.

[49] J.C. Kim, K.M. Lee, B.T. Choi, and S.U. Lee. A dense stereo matching using two-pass dynamic programming with generalized ground control points. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1075–1082, 2005.

[50] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *International Conference on Computer Vision*, volume 2, pages 508–515, 2002.

[51] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *European Conference on Computer Vision*, volume 3, pages 82–96, 2002.

[52] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

[53] K. Konolige. Small vision system: Hardware and implementation. 1997. http://www.ai.sri.com/ konolige/svs/svm.htm.

[54] S.H. Lee, Y. Kanatsugu, and J.I. Park. Map-based stochastic diffusion for stereo matching and line fields estimation. *International Journal of Computer Vision*, 47(1-3):195–218, 2002.

[55] M. Lhuillier and L. Quan. Match propagation for image-based modeling and rendering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1140–1146, 2002.

[56] M. Lin and C. Tomasi. Surfaces with occlusions from layered stereo. In *Conference on Computer Vision and Pattern Recognition*, pages 710–717, 2003.

[57] M. Lin and C. Tomasi. Surfaces with occlusions from layered stereo. *Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1073–1078, 2004.

[58] J.J. Little. Accurate early detection of discontinuities. In *International Conference on Vision Interface*, pages 2–7, 1992.

[59] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo. *International Joint Conference on Artificial Intelligence*, pages 121–130, 1981.

[60] R. Manduchi and C. Tomasi. Distinctiveness maps for image matching. In *International Conference on Image Analysis and Processing*, pages 26–31, 1999.

[61] D. Markovic and M. Gelautz. Experimental combination of intensity and stereo edges for improved snake segmentation. In *7th International Conference on Pattern Recognition and Image Analysis*, pages 18–23, 2004.

[62] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194:209–236, 1976.

[63] D. Marr and T. Poggio. A computational theory of human stereo vision. In *Proceedings Of the Royal Society of London*, volume 204, pages 301–328, 1979.

[64] H. Mayer. Analysis of means to improve cooperative disparity estimation. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XXXIV, pages 25–31, 2003 (Part 3/W8).

[65] B. McCane, K. Novins, D. Crannitch, and B. Galvin. On benchmarking optical flow. *Computer Vision and Image Understanding*, 84(1):216–143, 2001.

[66] A. Mitiche and P. Bouthemy. Computation and analysis of image motion: A synopsis of current problems and methods. *International Journal of Computer Vision*, 19(1):29–55, 2001.

[67] K. Mühlmann, D. Maier, J. Hesser, and R. Männer. Calculating dense disparity maps from color stereo images, an efficient implementation. *International Journal of Computer Vision*, 47(1):79–88, 2002.

[68] H.H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *Transactions on Pattern Analysis and Machine Intelligence*, 8(5):565–593, 1986.

[69] A. S. Ogale and Y. Aloimonos. Stereo correspondence with slanted surfaces: critical implications of horizontal slant. In *Conference on Computer Vision and Pattern Recognition*, pages 568–573, 2004.

[70] Y. Ohta and T. Kanade. Stereo by intra- and inter- scanline search. *Transactions on Pattern Analysis and Machine Intelligence*, 7(2):139–154, 1985.

[71] I. Patras, E. Hendirks, and R. Lagendijk. Video segmentation by map labeling of watershed segments. *Transactions on Pattern Analysis and Machine Intelligence*, 23(3):326–332, 2001.

[72] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317:314–319, 1985.

[73] R. Potts. Some generalized order-disorder transformation. In *Proceedings of the Cambridge Philosophical Society*, volume 48, pages 106–109, 1952.

[74] M. Proesmans, L. van Gool, E. Pauwels, and A. Oosterlinck. Determination of optical flow and its discontinuities using non-linear diffusion. In *European Conference on Computer Vision*, pages 295–304, 1994.

[75] S. Roy and I. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *International Conference on Computer Vision*, pages 492–499, 1998.

[76] R. Sara. Finding the largest unambiguous component of stereo matching. In *European Conference on Computer Vision*, volume 2, pages 900–914, 2002.

[77] D. Scharstein. View synthesis using stereo vision. *Lecture Notes in Computer Science (LNCS)*, 1583, 1999.

[78] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7–42, 2002. http://www.middlebury.edu/stereo/.

[79] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Conference on Computer Vision and Pattern Recognition*, pages 195–202, 2003.

[80] J. Shao. Generation of temporally consistent multiple virtual camera views from stereoscopic image sequences. *International Journal of Computer Vision*, 47(1-3):171–180, 2002.

[81] J. Shi and C. Tomasi. Good features to track. In *Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[82] C. Stiller and J. Konrad. Estimating motion in image sequences. *IEEE Signal Processing Magazine*, 16:70–91, 1999.

[83] C. Sun. Fast stereo matching using rectangular subregioning and 3d maximum-surface techniques. *International Journal of Computer Vision*, 47(1-3):99–117, 2002.

[84] J. Sun, Y. Li, S.B. Kang, and H.Y. Shum. Symmetric stereo matching for occlusion handling. In *Conference on Computer Vision and Pattern Recognition*, volume 25, pages 399–406, 2005.

[85] J. Sun, N.N. Zheng, and H.Y. Shum. Stereo matching using belief propagation. *Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

[86] R. Szeliski and D. Scharstein. Symmetric subpixel stereo matching. In *European Conference on Computer Vision*, volume 2, pages 525–540, 2002.

[87] H. Tao and H. Sawhney. Global matching criterion and color segmentation based stereo. In *Workshop on the Application of Computer Vision*, pages 246–253, 2000.

[88] H. Tao, H. Sawhney, and R. Kumar. A global matching framework for stereo computation. In *IEEE International Conference on Computer Vision*, pages 532–539, 2001.

[89] H. Tao, H. Sawhney, and R. Kumar. Object tracking with bayesian estimation of dynamic layer representations. *Transactions on Pattern Analysis and Machine Intelligence*, 24(1):75–89, 2002.

[90] M.F. Tappen and W.T. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In *International Conference on Computer Vision*, volume 2, pages 900–906, 2003.

[91] R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, 1987.

[92] O. Veksler. Dense features for semi-dense stereo correspondence. *International Journal of Computer Vision*, 47(1–3):247–260, 2002.

[93] O. Veksler. Stereo correspondence with compact windows via minimum ratio cycle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1654–1660, 2002.

[94] O. Veksler. Fast variable window for stereo correspondence using integral images. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 556–564, 2003.

[95] O. Veksler. Stereo correspondence by dynamic programming on a tree. In *Conference on Computer Vision and Pattern Recognition*, pages 384–390, 2005.

[96] J. Wang and E. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994.

[97] Y. Wei and L. Quan. Region-based progressive stereo matching. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 106–113, 2004.

[98] J. Weickert and C. Schnorr. Variational optic flow computation with a spatio-temporal smoothness constraint. *Journal of Mathematical Imaging and Vision*, 14(3):245–255, 2001.

[99] J. Willis, S. Agarwal, and S. Belongie. What went where. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 37–44, 2003.

[100] J. Xiao and M. Shah. Accurate motion layer segmentation and matting. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pages 698–703, 2005.

[101] J. Xiao and M. Shah. Motion layer extraction in the presence of occlusion using graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1644–1659, 2005.

[102] K.J. Yoon and I.S. Kweon. Locally adaptive support-weight approach for visual correspondence search. In *Conference on Computer Vision and Pattern Recognition*, pages 924–931, 2005.

[103] A.L. Yuille and T. Poggio. A generalized ordering constraint for stereo correspondence. *A.I. Laboratory Memo 777, MIT, Cambridge*, 1984.

[104] Y. Zhang and C. Kambhamettu. Stereo matching with segmentation-based cooperation. In *European Conference on Computer Vision*, pages 556–571, 2002.

[105] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[106] Z. Zhang and Y. Shan. A progressive scheme for stereo matching. *Lecture Notes in Computer Science (LNCS): 3D Structure from Images - SMILE 2000*, 2018:68–85, 2001.

[107] Y. Zhou and H. Tao. A background layer model for object tracking through occlusion. In *International Conference on Computer Vision*, pages 1079–1085, 2003.

[108] C. Zitnick and T. Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):675–684, 2000.

[109] L. Zitnick, S. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Transaction on Graphics*, 23(3):600–608, 2004, http://research.microsoft.com/users/mattu/.

Curriculum vitae

Michael Bleyer was born in Wiener Neustadt, Austria, in 1977. He studied computer science at the Vienna University of Technology and received his Dipl.-Ing. degree with distinction in 2002. He is currently working as a PhD student at the Institute for Software Technology and Interactive Systems of the Vienna University of Technology. He is a member of the Interactive Media Systems group where he is involved in the project "3D Video Analysis for Interactive Multimedia Applications (Video3D)", which is funded by the Austrian Science Fund (FWF) under project number P15663. Michael Bleyer's research interest lies in the area of Computer Vision. More specifically, his work is focused on the 3-dimensional scene reconstruction from two or more images, the analysis of motion vectors (optical flow), as well as video object segmentation.