Institutionen för systemteknik

Department of Electrical Engineering

Examensarbete

Modern Stereo Correspondence Algorithms: Investigation and evaluation

Examensarbete utfört i informationskodning
av

Anders Olofsson

LiTH-ISY-Ex--10/4432--SE
Linköping 2010

TEKNISKA HÖGSKOLAN
LINKÖPINGS UNIVERSITET

Department of Electrical Engineering
Linköping University
S-581 83 Linköping, Sweden

Linköpings tekniska högskola
Institutionen för systemteknik
581 83 Linköping

Modern Stereo Correspondence Algorithms

Investigation and evaluation

Anders Olofsson

17 June 2010
Linköping

LiTH-ISY-Ex--10/4432--SE

Supervisor:	Apostolos Georgakis, PhD Ericsson Research, Kista, Stockholm
Examiner:	Professor Robert Forchheimer Linköping University, Linköping

Presentationsdatum
2010-06-17
Publiceringsdatum (elektronisk version)
2010-07-01

Institution och avdelning
Institutionen för systemteknik

Department of Electrical Engineering

Linköpings universitet

Språk
___ Svenska
_X_ Annat (ange nedan)
Engelska
Antal sidor
102

Typ av publikation
____Licentiatavhandling
_X_ Examensarbete
____C-uppsats
____D-uppsats
____Rapport
____Annat (ange nedan)

ISBN

ISRN LiTH-ISY-Ex--10/4432--SE

Serietitel

Serienummer/ISSN

URL för elektronisk version
http://www.ep.liu.se

Publikationens titel
Modern Stereo Correspondence Algorithms: Investigation and Evaluation

Författare
Anders Olofsson

Sammanfattning

Many different approaches have been taken towards solving the stereo correspondence problem and great progress has been made within the field during the last decade. This is mainly thanks to newly evolved global optimization techniques and better ways to compute pixel dissimilarity between views. The most successful algorithms are based on approaches that explicitly model smoothness assumptions made about the physical world, with image segmentation and plane fitting being two frequently used techniques.

Within the project, a survey of state of the art stereo algorithms was conducted and the theory behind them is explained. Techniques found interesting were implemented for experimental trials and an algorithm aiming to achieve state of the art performance was implemented and evaluated. For several cases, state of the art performance was reached.

To keep down the computational complexity, an algorithm relying on local winner-take-all optimization, image segmentation and plane fitting was compared against minimizing a global energy function formulated on pixel level. Experiments show that the local approach in several cases can match the global approach, but that problems sometimes arise – especially when large areas that lack texture are present. Such problematic areas are better handled by the explicit modeling of smoothness in global energy minimization.

Lastly, disparity estimation for image sequences was explored and some ideas on how to use temporal information were implemented and tried. The ideas mainly relied on motion detection to determine parts that are static in a sequence of frames. Stereo correspondence for sequences is a rather new research field, and there is still a lot of work to be made.

Nyckelord
stereo correspondence, stereo matching, cost function, energy minimization, graph cuts, belief propagation, RANSAC plane fitting, mean-shift image segmentation

Abstract

Many different approaches have been taken towards solving the stereo correspondence problem and great progress has been made within the field during the last decade. This is mainly thanks to newly evolved global optimization techniques and better ways to compute pixel dissimilarity between views. The most successful algorithms are based on approaches that explicitly model smoothness assumptions made about the physical world, with image segmentation and plane fitting being two frequently used techniques.

Within the project, a survey of state of the art stereo algorithms was conducted and the theory behind them is explained. Techniques found interesting were implemented for experimental trials and an algorithm aiming to achieve state of the art performance was implemented and evaluated. For several cases, state of the art performance was reached.

To keep down the computational complexity, an algorithm relying on local winner-take-all optimization, image segmentation and plane fitting was compared against minimizing a global energy function formulated on pixel level. Experiments show that the local approach in several cases can match the global approach, but that problems sometimes arise – especially when large areas that lack texture are present. Such problematic areas are better handled by the explicit modeling of smoothness in global energy minimization.

Lastly, disparity estimation for image sequences was explored and some ideas on how to use temporal information were implemented and tried. The ideas mainly relied on motion detection to determine parts that are static in a sequence of frames. Stereo correspondence for sequences is a rather new research field, and there is still a lot of work to be made.

Acknowledgements

I would like to begin with thanking Professor *Robert Forchheimer* for influencing me to do a thesis work related to computer vision and information coding. He has given me a lot of good advice, and not only regarding the thesis work, but for many other things as well.

I would also like to thank my supervisor, *Apostolos Georgakis*, for his guidance and support during the work. He has provided me with valuable insight about working in the field of research.

My opponent and long time friend, *Ulf Kargén*, also deserves to be mentioned here as he suggested several good ideas on how to improve the report. He has also shown great patience in long discussions over the phone.

I am also grateful of how my two dear sisters, *Erica* and *Carina*, have taken care of me during this spring. They have occasionally been spoiling me with dinners and cultural events.

I would also like to thank all of *my wonderful friends*, who have managed to keep up my mood during periods of stress. Hopefully, you know who you are.

Finally, and not to forget, a thanks goes to the *people who brought cake* to the Ericsson office in Kista on Fridays.

Anders Olofsson

June 2010

Contents

1 Introduction

This report is part of a master's thesis work carried out at the department of *Visual Technology* at *Ericsson Research* in Kista, Stockholm, Sweden. A central topic of the thesis work has been *depth estimation*, which is about determining the distance to different objects present in a camera view. Ericsson is interested in this as the company is nowadays not only into telecommunication, but also into TV broadcasting services.

1.1 Depth estimation and 3DTV

Two areas closely related to depth estimation are *Free Viewpoint Television* (FTV) and *Multi-View Coding* (MVC). FTV is about giving television viewers the ability to display scenes from freely chosen, arbitrary view angles and MVC is about performing efficient predictive coding of neighboring, highly correlated camera views. Ericsson participates in the MPEG research activities related to FTV and MVC.

In the end, all work being done within the fields of FTV and MVC shares the same objective – to give the TV audience the opportunity to experience 3D. When looking at the increasing number of 3D cinemas around the world and the huge success of the 3D movie *Avatar*, there is no doubt about 3D being an interesting topic, and the fact that home entertainment systems able to show 3D content are starting to show up further indicates this.

Fundamentally, the 3D experience comes from the left and right eye seeing slightly different views. Thus, by broadcasting two separate views for the left and right eye, 3D can be perceived. In practice, however, it is more desirable to send only one camera view together with side information. A popular approach is to let ordinary 2D camera imagery be accompanied by *depth maps*, a format which is often referred to as '2.5D' or 'texture + depth' (see Figure 1 below for an illustration). Benefits of this format are the ability to synthesize novel views and that depth maps are highly compressible due to their characteristics.

(a) (b)

Figure 1. Illustration of the 2.5D 'texture + depth' format: a) An ordinary 2D camera image b) Depth map belonging to the image. In this example, pixels go darker as the depth grows.

In practice, depth maps are stored as grayscale images that show distance instead of texture. This means that an object located close to the camera turns out bright while a faraway located object looks darker (or vice versa). Also, the relation between intensity and distance has to be specified somehow (e.g. by defining the nearest and farthest distance values of the intensity range $0 - 255$).

1.2 Acquiring depth information

Depth information for a 2D image may be acquired in several ways. One example is by using a *laser range camera* and another way is using a stereo image pair in combination with triangulation. The latter method is normally referred to as *stereo vision, stereo matching* or *stereo correspondence*.

In the work related to FTV and MVC, a method based on stereo correspondence is currently used to generate depth maps. Therefore, the thesis work has been focusing on using stereo correspondence for computing depth.

In principle, stereo correspondence is about finding matching pixels in two given input images taken from slightly horizontally separated points. The technique will be discussed more thoroughly in Chapter 2.

1.3 Objective of the work

The main objective of the thesis work has been to look closer into and investigate modern stereo correspondence algorithms in order to get an orientation within the field.

Moreover, algorithmic components found interesting have been implemented for experimental trials. Stereo correspondence for image sequences has also been looked into and explored to some extent.

The technique that is currently used for depth estimation in FTV and MVC related research is rather slow, and it has been desirable to find faster techniques that give comparable results.

1.4 Disposition

In Chapter 2, the stereo correspondence problem is defined. The connection between stereo correspondence and depth estimation is given and some of the challenges that arise in stereo correspondence are described.

Chapter 3 aims to give a theoretical introduction to the techniques that are found in modern stereo correspondence algorithms.

A survey of modern stereo correspondence algorithms is presented in Chapter 4, and this survey has served to influence the contents of chapter 3.

An implementation description of a stereo algorithm follows in Chapter 5 and experimental results from the implementation are presented in Chapter 6.

Lastly, stereo correspondence for image sequences is treated in Chapter 7. Some existing approaches are presented, and own ideas are also implemented and tested.

1.5 Related work

1.5.1 Stereo correspondence

A lot of research has been done when it comes to stereo vision. Some surveys have been written to give an overview of popular methods and to compare their performance. Examples of surveys produced within the last decade are [1], [2] and [3].

A thorough taxonomy of dense stereo correspondence algorithms is [1]. The taxonomy gives a good overview of modern stereo algorithms and their intrinsic components. A continuously updated homepage [4] with a table of performance comparisons between different proposed algorithms comes along the taxonomy. As of today, results from over 80 algorithms are published on the page along with links to papers presenting the algorithms.

The homepage also provides several stereo image pairs together with ground truth data (see Section 2.2.3). This opens up for the ability to evaluate own algorithms against already existing ones. Results may also be submitted to the homepage in order to take part of the comparison table.

Lastly, there is also a C++ software framework freely distributed on the homepage, where many algorithmic components have been implemented.

1.5.2 Stereo correspondence for image sequences

Less research has been made for sequences of stereo images than for single stereo image pairs.

Some related work that has been found is [5], which describes an algorithm for simultaneously determining optical flow and disparity for stereo sequences by minimizing a global energy function. This algorithm forms the base of [6], which uses the optical flow computed as in the previously mentioned algorithm to make temporally consistent disparity output sequences. Yet another algorithm trying to make use of optical flow is [7], which minimizes an energy function that depends on both disparity and flow. By looking at the outline of these algorithms, they seem rather computationally demanding.

A lot of research in stereo for sequences is carried out by the automotive industry, with detection of vehicles and pedestrians as objective. In [8], the effect of using edge maps for dark scenes (unlit roads) is investigated. Another paper [9] gives an interesting comparison between different disparity estimation techniques for stereo sequences and compares different prediction error metrics that can be used for quality assessment. More information on prediction error metrics for stereo and optical flow is to be found in [10], which discusses some metrics and their benefits and drawbacks.

Lastly, [11] presents ways to improve disparity estimates using temporal information and gives some results. One technique that is explored is bilateral filtering of disparity, performed both spatially and temporally.

2 Stereo correspondence

Stereo correspondence is a traditional problem within computer vision and has been an active research topic for decades. Even though the task might seem simple at a first glance, there are several non-trivial problems to deal with. The basics behind stereo correspondence and some of its related problems will be discussed within this chapter.

2.1 Stereoscopic images

2.1.1 Left and right view images

A *stereo image pair* consists of two images of the same scene, taken from slightly horizontally separated points – from the *left view* and the *right view*.

The effect of taking images from horizontally separated points is demonstrated in Figure 2. Objects near the camera will be depicted more to the right in the left image – and more to the left in the right image. Faraway objects will be located at approximately the same place in both images that make up the stereo pair.

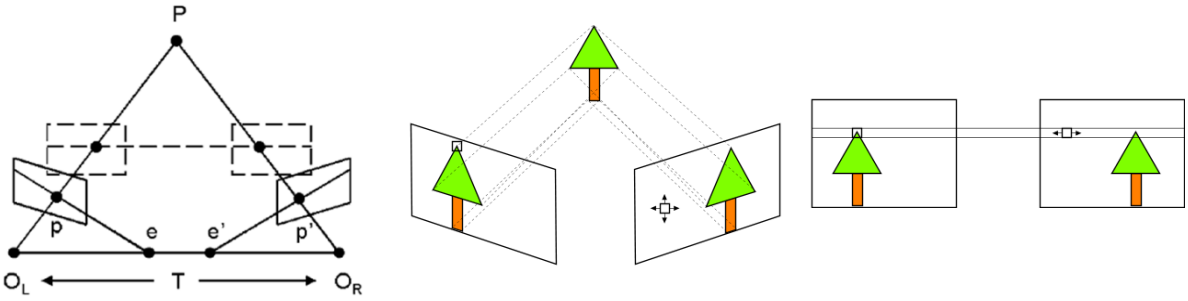
Figure 2. The principle of stereoscopic images: A scene is depicted in two pictures taken from two horizontally separated camera positions, resulting in a left and right image. Objects close to the camera will be placed more to the right in the left image – and more to the left in the right image. Faraway objects, such as the sun and the cloud, will be located at approximately the same position in both images.

The fact that the horizontal displacement of an object between the left and right view depends on the distance from the object to the camera viewpoints is exploited in stereo correspondence, where the goal is to find matching pixels in the left and right view images of a stereo image pair.

2.1.2 Image rectification

To greatly reduce the complexity of pixel matching, it is necessary to perform image rectification. The purpose of image rectification is to make the *epipolar lines* of two camera images horizontally aligned. This may be accomplished by using linear transformations that rotate, translate and skew the camera images. In the transformations, internal camera parameters and information about mutual camera positions and camera orientation are used.

Figure 3 illustrates the process. The 3D point P is projected onto the left and right camera image planes at points p and p'. Epipolar lines are drawn from p and p' along the image planes to the baseline T between the camera focal points O_L and O_R. These lines intersect the baseline in the *epipoles* e and e'. Two matching points p and p' are always located somewhere on the epipolar lines $\overline{ep}$ and $\overline{e'p'}$, respectively. This geometric constraint forms the basis for image rectification.



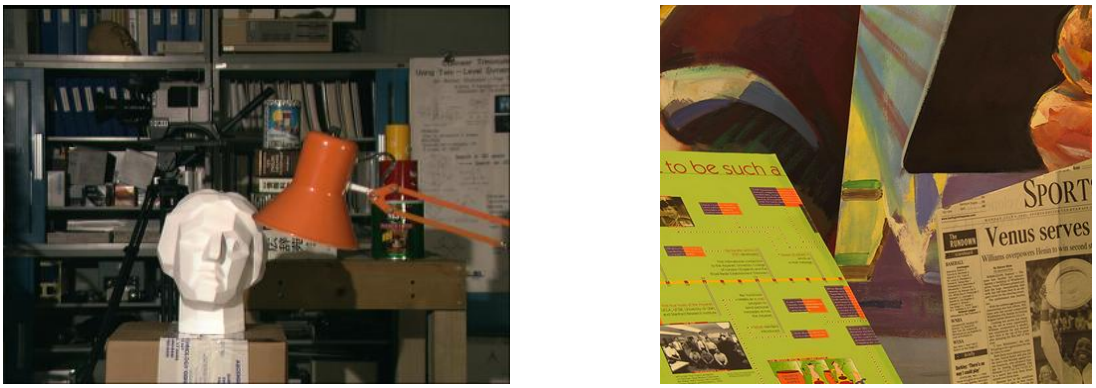
(a) (b) (c)

Figure 3. The principle of image rectification: a) The original camera image planes are drawn with solid borders and the rectified image planes are drawn with dashed borders. Epipolar lines are also drawn from the projected points p and p' to epipoles e and e'. [2] b) Camera images before rectification – pixel matching is a 2D search problem c) Camera images after rectification – now pixel matching can be done through a one dimensional line search.

After image rectification has been carried out, the epipolar lines of two projected points are parallel and horizontally aligned along the new image planes. The stereo matching problem is therefore reduced to a one dimensional search along horizontal lines instead of a two dimensional search. More information on image rectification can be found in [12] and [13].

2.1.3 Examples of images that are commonly in stereo research

In the ongoing stereo research of today, four rectified stereo image pairs are extensively used for benchmarking. These image pairs are named *Tsukuba*, *Venus*, *Cones* and *Teddy* and may be downloaded from the *Middlebury Stereo homepage* [4]. The left views of the image pairs are shown in Figure 4.



(a) (b)

(c)

(d)

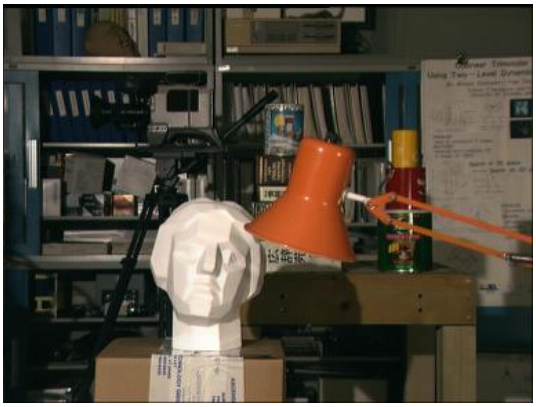
Figure 4. Stereo image pairs that are extensively used in research to compare performance of proposed algorithms: a) Tsukuba b) Venus c) Cones d) Teddy

2.2 Disparity

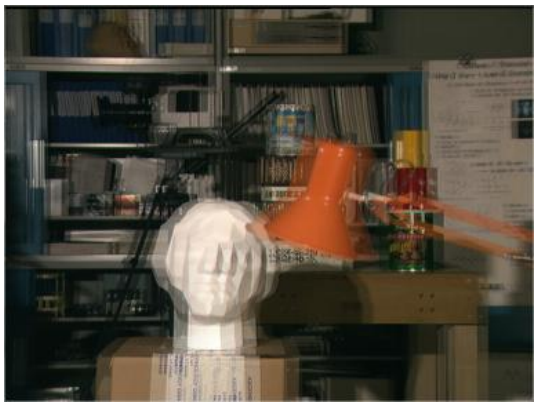
2.2.1 Disparity and disparity map for a reference view

In stereo correspondence, the target is to find matching pixels of two given input images and the result from finding matching pixels is normally saved in a *disparity map*. The term disparity can be looked upon as horizontal distance between two matching pixels and the disparity map defines a value of this horizontal pixel distance for each image pixel coordinate. Hence, it may be seen as a function $d(x, y)$ of image pixel coordinates. The *Tsukuba* stereo image pair is shown together with a disparity map in Figure 5.

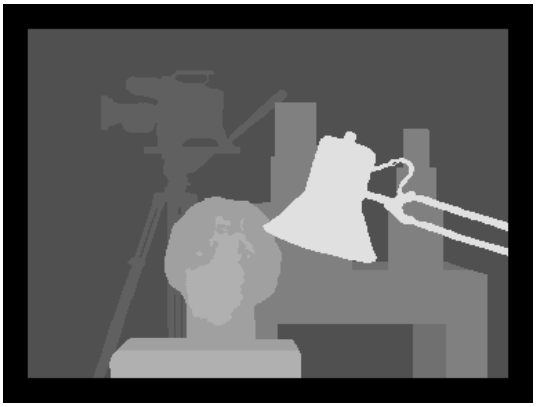
(a)



(b)



(c)



(d)

Figure 5. The Tsukuba stereo image pair: a) Left view b) Right view c) Superimposed left and right view, for illustration purpose. Note the higher disparity for the lamp and the head, relatively to objects in the background. d) Disparity map (a so called ground truth map, scaled with a factor of 16 to make use of the whole intensity range)

The left and right images of a stereo pair are sometimes denoted as *reference image* and *search image*. The reference image is defined by which view the disparity map is generated for. In Figure 5 (d), the viewpoint of the disparity map is the same as for the left image, and the left image is therefore the reference image in this case[1].

2.2.2 Relations for horizontal coordinates of a stereo image pair

By using a disparity map, pixel coordinates of objects in stereo images can be related to each other as follows:

$$x_{RIGHT} = x_{LEFT} - d_{LEFT}(x_{LEFT}, y) \tag{2.1}$$

$$x_{LEFT} = x_{RIGHT} + d_{RIGHT}(x_{RIGHT}, y) \tag{2.2}$$

$$x_{SEARCH} = x_{REF} + s \cdot d_{REF}(x_{REF}, y), \qquad s = \pm 1 \tag{2.3}$$

$$y = y_{LEFT} = y_{RIGHT} \tag{2.4}$$

In the above relations, $d_{LEFT}(x, y)$ and $d_{RIGHT}(x, y)$ denote disparity maps for the left and right views. It is also possible to write the relations as in (2.3), with the difference that the direction of the coordinate shifting has to be specified.

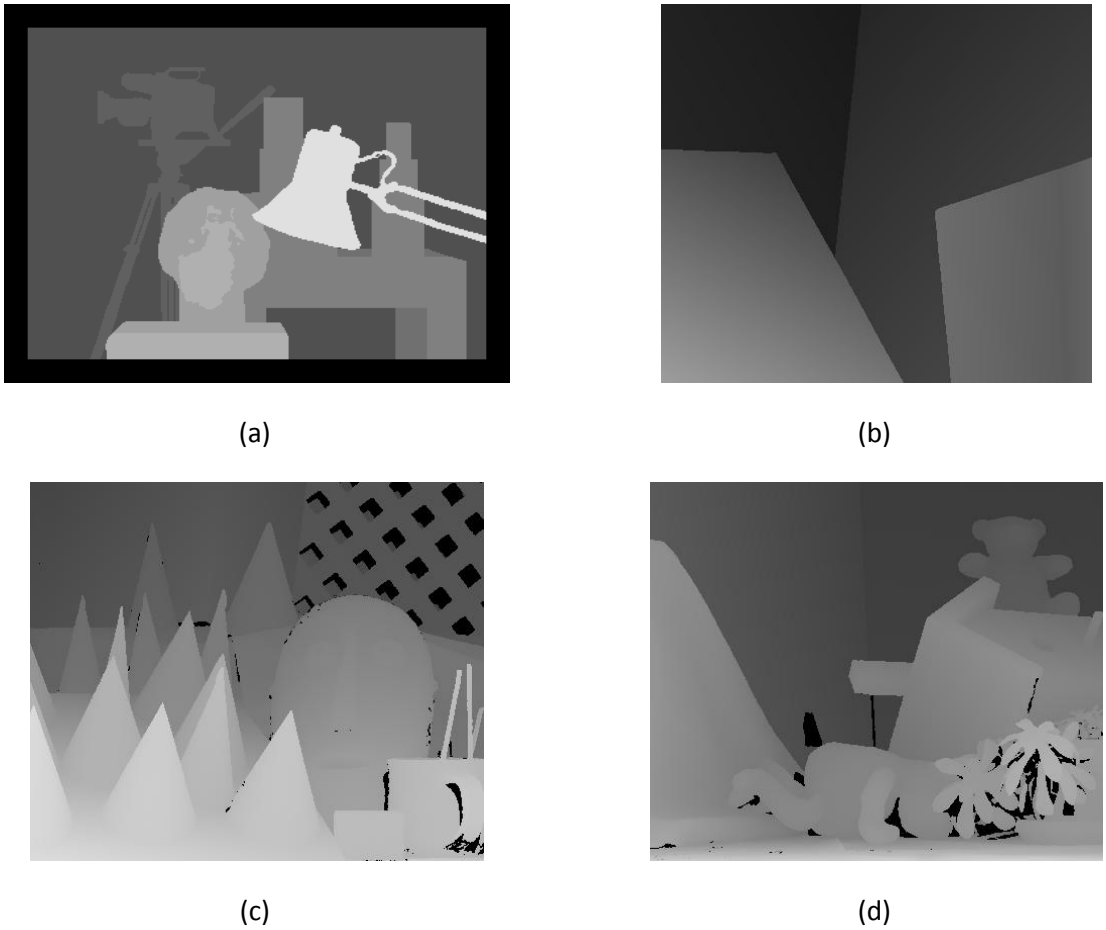
2.2.3 Ground truth disparity maps

The disparity map displayed in Figure 5 (d) shows *true* disparities for corresponding pixels. A disparity map with 'true' disparity values is referred to as *ground truth* and may be very useful when evaluating different approaches to calculate disparity.

[1] Selecting the left image as reference image tends to be the more common choice in stereo correspondence algorithms.

One straight-forward way to create ground truth disparity maps is through the use of range cameras. Another way used by the authors of [1] is to let scenes be composed of piecewise planar surfaces only, which are manually labeled. Corresponding points of the planar surfaces are identified in high resolution stereo images, and planes which model the disparity of the surfaces are fitted. This results in ground truth maps with subpixel disparity values for downsampled versions of the images. To provide more challenging stereo images with ground truth data, the same authors later used a technique based on structured light (see [14] for a description of this method).

In the figure below, ground truth disparity maps belonging to the image pairs in Figure 4 are displayed (note that the ground truth disparity maps are scaled to use a larger range of the available intensity interval).



(a) (b)

(c) (d)

Figure 6. Ground truth data: a) Tsukuba ground truth, consisting of fronto-parallel planar surfaces b) Venus ground truth, consisting of planar surfaces c) Cones ground truth data, created from a technique based on structured light d) Teddy ground truth, made in the same way as (c)

In Appendix A it is possible to see the left view images placed next to their corresponding ground truth disparity maps.

2.2.4 Relation between disparity and depth

By using a rectified stereo image pair, triangulation can be performed to calculate the depth. The idea is illustrated for an arbitrarily located 3D point P in Figure 7.

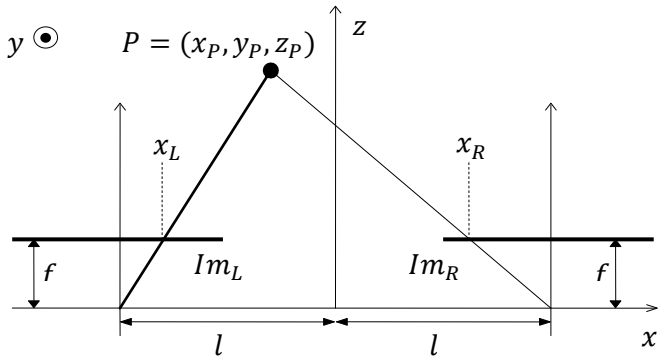


Figure 7. The disparity is given by the difference $d = x_L - x_R$, where x_L is the x-coordinate of the projected 3D coordinate x_P onto the left camera image plane Im_L and x_R is the x-coordinate of the projection onto the right image plane Im_R.

From the figure it is seen that the following relations, all of which are independent of y, hold:

$$d = x_L - x_R = f\left(\frac{x_P + l}{z_P} - \frac{x_P - l}{z_P}\right) = \frac{2fl}{z_P} \tag{2.5}$$

$$z_P = \frac{2fl}{d} \tag{2.6}$$

This means that once the disparity is computed, depth may in turn be found when given camera parameters such as focal length f and baseline distance $T = 2l$. Note that this simple way of relating disparity and depth demands that the *pinhole camera model*[2] is fairly valid.

Depth and disparity are sometimes used synonymously, but as this project is closely related to stereo correspondence, disparity will be used from this point and on.

2.3 Stereo matching constraints

Apart from the epipolar constraint used in image rectification, there are several other constraints that may be exploited in stereo correspondence. The constraints are sometimes implemented and used explicitly, but are normally used implicitly. Some of the most commonly referred constraints will be brought up here.

2.3.1 Pixel intensity similarity

The *similarity constraint* is very fundamental and states that pixel intensities belonging to the same point of an object in the left and right images should be almost equal for corresponding pixels. Thus, their color and intensity value must not vary significantly between different viewpoints.

2.3.2 Uniqueness

The *uniqueness constraint* is also essential and means that for every pixel in the left input image there is only one matching pixel in the right image. Even though this may seem natural, it will be seen in Section 2.4.1 that this constraint is not trivial as it does not hold at all times.

[2] A good source for further reading on camera models is [12].

2.3.3 Smoothness and continuity

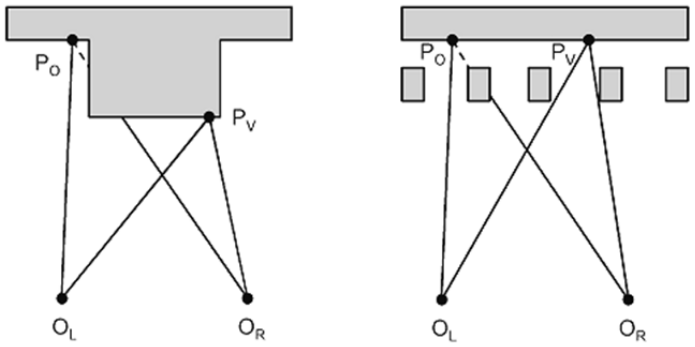
A constraint that it referred to very often in modern stereo correspondence research is the *smoothness constraint*, sometimes also called the *continuity constraint*. This constraint postulates that the depth of physical surfaces in the real world is a smoothly varying property, except for at object boundaries where discontinuities may be present [15].

2.4 Fundamental problems in stereo correspondence

There are a lot of challenges when solving stereo correspondence problems and some of the most important problems encountered in stereo matching will be described here.

2.4.1 Occlusion

One problem which needs to be dealt with is *occlusion*. Occlusion occurs when a point in the 3D space in front of the cameras gets depicted in one of the images and is blocked from being depicted in the other one. See Figure 8 below for an illustration of the problem.



(a)

(b) (c)

Figure 8. The occlusion problem: a) The points P_V are visible to both cameras and disparity estimation is therefore possible for these points. However, the points P_O are occluded for the right camera, and no matching pixels will be found [2]. b) and c) Occlusion in the Tsukuba image pair: Yellow areas in the left image are occluded from the right camera and areas of the same color in the right image are occluded from the left camera.

For the observant reader, it may be obvious that the earlier mentioned uniqueness constraint does not hold for occluded areas. This is used in stereo algorithms, and will be described in Section 3.5.

There is no general solution to the occlusion problem and as occlusion may lead to spurious matches, one goal in stereo correspondence research is to somehow minimize the impact of occlusion on the final result.

2.4.2 Uniform texture and lack of texture

Another challenge is to handle surfaces with uniform texture and surfaces that lack texture. When surfaces with these properties are encountered in a stereo image pair, it immediately becomes complicated to decide which pixels in the left and right images that are corresponding to each other. This, in turn, leads to an elevated risk of spurious matches – with resulting mismatch errors in the calculated disparity map.

2.4.3 Sensor noise and bias

Camera sensor noise is yet another issue for stereo matching in that two matching pixels that *should* have the same intensities may not. This problem is especially apparent in poorly textured image regions or in images taken under poor lighting conditions, thus having low signal-to-noise ratio.

If two different cameras are used to capture the stereo images there might also be a difference in gain or a bias in the intensity of matching pixels in the left and right image. Depending on the matching technique being used, the images may have to be preprocessed to correct this issue in order to prevent mismatching.

2.4.4 Example of a problematic stereo image pair

Of course, several problems may be simultaneously present in a stereo image pair. Several of the discussed problems are present in Figure 9, which shows two stereo images taken in a dark garage.

Left image Right image

Figure 9. Example of a problematic stereo image pair taken in a dark garage. Occlusion is caused by the pillar, pixels belonging to the untextured floor are hard to match and the poor lighting in the garage makes the noise clearly visible.

3 Introduction to stereo correspondence algorithms

This chapter will introduce some concepts that are common in stereo algorithms of today. Its purpose is to give an introduction to the area before looking closer into some modern stereo algorithms, and before presenting the implementation part of the project.

3.1 Pixel and block matching

3.1.1 Pixel dissimilarity and cost function

In order to find corresponding pixels in a search and reference image, there is obviously a need for a pixel similarity measure. It is however more common to use the term *dissimilarity measure* or *matching cost*, which increases as the similarity between two compared pixels decreases.

A common notation for representing matching cost is through a function $C(x, y, d)$ of reference image coordinates and disparity. The cost function returns the value of the dissimilarity for a coordinate (x, y, d) in the *disparity space*. The disparity space is made up by the available image pixel coordinates and the *disparity search range*. Normally, a disparity search range has to be specified manually and depends upon the characteristics of the input image pair.

3.1.2 Commonly occurring measures and cost aggregation

Over the years, a large amount of dissimilarity measures have been presented. Common measures that are used for pixel-wise comparison are *absolute intensity difference* (AD), *squared intensity difference* (SD) and *absolute gradient difference* (GRAD). By extending the comparison to square window regions centered about the search and reference pixels, these measures are turned into *sum of absolute intensity differences* (SAD), *sum of squared intensity differences* (SSD) and *sum of absolute gradient differences* (SGRAD). Summing costs over window regions is referred to as *cost aggregation*.

The main benefit of cost aggregation is that the high noise sensitivity of pixel-wise comparison is reduced. However, area based methods tend to generate less detailed results. Therefore, a tradeoff between noise sensitivity and loss of detail must be made when selecting aggregation window size.

The cost functions for the above discussed dissimilarity measures may be written as:

$$C_{AD}(x, y, d) = |I_L(x, y) - I_R(x - d, y)| \tag{2.7}$$

$$C_{SD}(x, y, d) = |I_L(x, y) - I_R(x - d, y)|^2 \tag{2.8}$$

$$\begin{aligned} C_{GRAD}(x, y, d) = {} & |\nabla_x I_L(x, y) - \nabla_x I_R(x - d, y)| \\ & + \left|\nabla_y I_L(x, y) - \nabla_y I_R(x - d, y)\right| \end{aligned} \tag{2.9}$$

$$C_{SAD}(x, y, d) = \sum_{(u,v)\in W(x,y)} |I_L(u, v) - I_R(u - d, v)| \tag{2.10}$$

$$C_{SSD}(x, y, d) = \sum_{(u,v)\in W(x,y)} |I_L(u, v) - I_R(u - d, v)|^2 \tag{2.11}$$

$$C_{SGRAD}(x,y,d) = \sum_{(u,v)\in W(x,y)} |\nabla_x I_L(u,v) - \nabla_x I_R(u-d,v)| + \sum_{(u,v)\in W(x,y)} \left|\nabla_y I_L(u,v) - \nabla_y I_R(u-d,v)\right| \quad (2.12)$$

Here, I_L and I_R are pixel intensity functions of the left and right image, respectively. $W(x,y)$ is some window that surrounds the position (x,y), and ∇_x and ∇_y are gradient operators.

3.1.3 The cost function as an image: Disparity Space Image

Sometimes, the cost $C(x,y,d)$ is referred to as *disparity space image* (DSI) [1]. This is due to the fact that for each fixed integer disparity value d, the function represents an image visualizing the cost for every pixel location (x,y). In Figure 10 below are three slices from the cost function $C_{SAD}(x,y,d)$ created from the Tsukuba image pair for disparity values $d = 5, 11, 14$ using a $9x9$ pixel square shaped aggregation window.

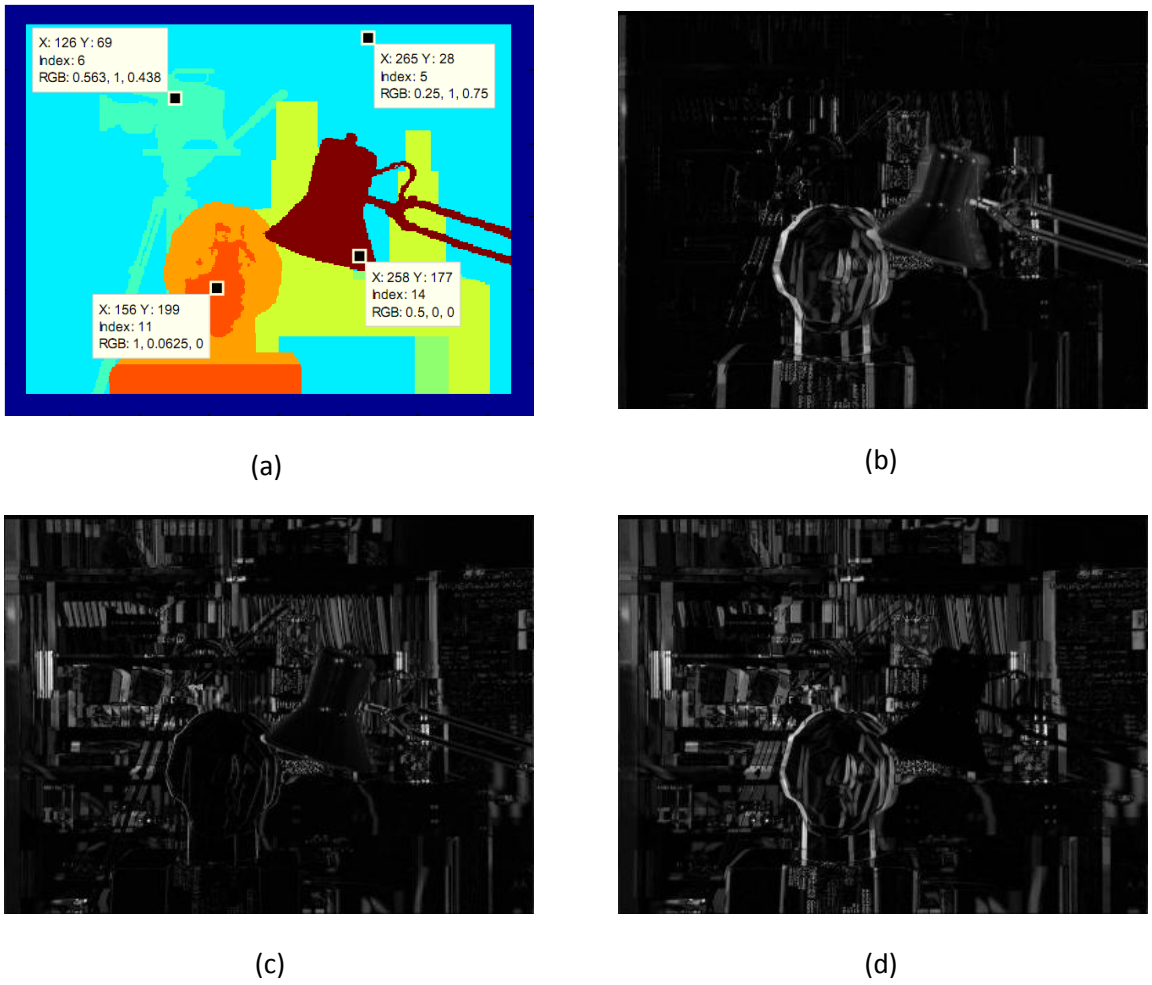


Figure 10. Three slices of the matching cost $C_{SAD}(x,y,d)$ of the Tsukuba image pair: a) Ground truth disparity map b) $d = 5$ c) $d = 11$ d) $d = 14$. Dark areas have low cost and from the slices it can be seen that the background has a disparity of around 5 pixels, the sculpture 11 pixels and that the lamp has a disparity of 14 pixels.

3.1.4 The dissimilarity measure of Birchfield-Tomasi

A widely adopted technique for calculating dissimilarity between pixels in a reference and search image is the *dissimilarity measure of Birchfield-Tomasi* (BT) [16]. This dissimilarity measure is an alternative to regular absolute differences with the underlying to use linear interpolation of the left and right view intensity functions for better handling of subpixel disparity shifts. As a result, the impact of image sampling effects is reduced.

Mathematically, the BT dissimilarity measure is written as:

$$d(x_L, x_R) = \min\{\overline{d}(x_L, x_R, I_L, I_R), \overline{d}(x_R, x_L, I_R, I_L)\} \tag{2.13}$$

$$\overline{d}(x_L, x_R, I_L, I_R) = \min_{x_R - 1/2 \le x \le x_R + 1/2} \left| I_L(x_L) - \hat{I}_R(x) \right| \tag{2.14}$$

$$\overline{d}(x_R, x_L, I_R, I_L) = \min_{x_L - 1/2 \le x \le x_L + 1/2} \left| \hat{I}_L(x) - I_R(x_R) \right| \tag{2.15}$$

Here, x_L is an integer coordinate on a scanline of the left image, x_R is an integer coordinate on the same scanline of the right image, $I_L(x)$ and $I_R(x)$ are intensity functions describing intensity values of integer coordinates along the scanlines, and $\hat{I}_L(x)$ and $\hat{I}_R(x)$ are linearly interpolated functions of the intensity values.

A demonstration of when the BT measure proves useful is shown in Figure 11, where the intensity function is shifted with a disparity of 0.4 pixels. In the example, (2.14) is computed – i.e. x_L is held fixed and the right intensity function is interpolated.

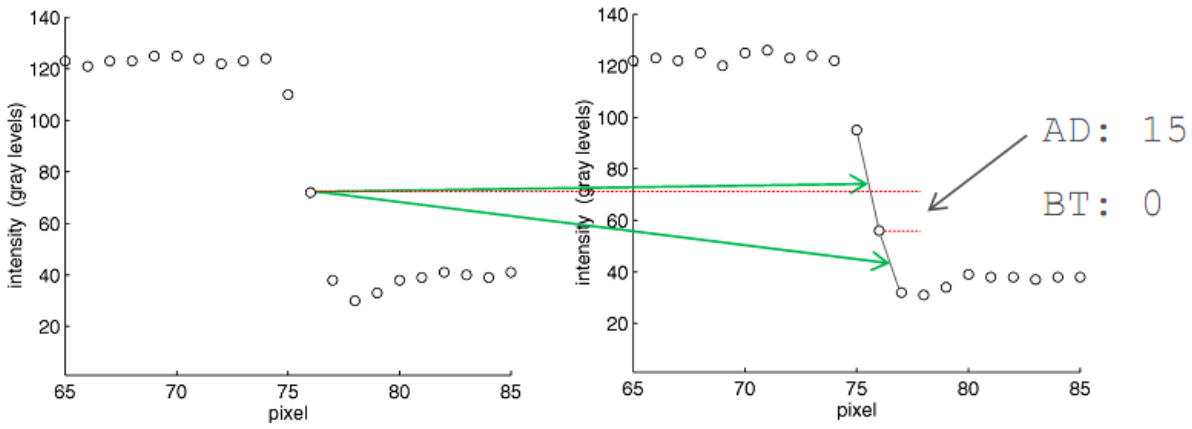


Figure 11. Example of when the dissimilarity measure of BT is helpful. The intensity functions of the left and right plot are shifted with a disparity of 0.4 pixels, and the error should therefore be zero. Note that BT reduces the problem that arises for ordinary AD.

As piecewise linear functions take extreme values in their discontinuous points or ending points, the calculation of $\overline{d}(x_L, x_R, I_L, I_R)$ is carried out with the use of a few addition, multiplication and comparison operations. Thus, there is not much computational burden added when using BT.

3.1.5 Two recent adaptive window approaches

A way to aggregate cost, which has recently started to show up in algorithms, is to set weights of pixels within aggregation windows based on both color intensity similarity with and distance to the

center pixel. One of the approaches to do this is based on the *bilateral filtering* technique which was introduced into computer vision in the late nineties [17].

In bilateral filtering, the support-weights are defined in the following way:

$$w(i,j,x,y) = \exp\left(-\frac{\|C(i,j) - C(x,y)\|^2}{\gamma_r} - \frac{\|(i,j) - (x,y)\|^2}{\gamma_s}\right) \tag{2.16}$$

Here, $C(i,j)$ is the color vector of pixels in some color space (or plain intensity values), γ_r a parameter that determines how color differences affect the result and γ_s a parameter that decides the impact of spatial distance from the center pixel on the weights.

This filtering technique was applied to stereo matching in [18]. Their approach is based on separately calculating the support weights both for the reference and the search window:

$$w_{L,R}(i,j,x,y) = \exp\left(-\frac{\|C(i,j) - C(x,y)\|}{\gamma_r} - \frac{\|(i,j) - (x,y)\|}{\gamma_s}\right), \tag{2.17}$$

The weights w_L and w_R are multiplied and the matching cost is aggregated as follows, where $d(i,j,d)$ is the used dissimilarity measure:

$$C(x,y,d) = \frac{\sum_{(i,j)\in W(x,y)} w_L(i,j,x,y) w_R(i-d,j,x-d,y) d(i,j,d)}{\sum_{(i,j)\in W(x,y)} w_L(i,j,x,y) w_R(i-d,j,x-d,y)} \tag{2.18}$$

The window $W(x,y)$ is normally a rather large square window centered about (x,y). Window sizes as large as $31x31$ pixels are not unusual in this approach.

The above discussed approach has been further developed in [19]. Instead of using the negative exponential of color dissimilarity and spatial distance to calculate weights, the *geodesic color distance* to the center pixel is defined and used. This distance between two pixels $c = (x_c, y_c)$ and $p = (x_p, y_p)$ is expressed as:

$$D(c,p) = \min_{P\in P_{c,p}} d(P) \tag{2.19}$$

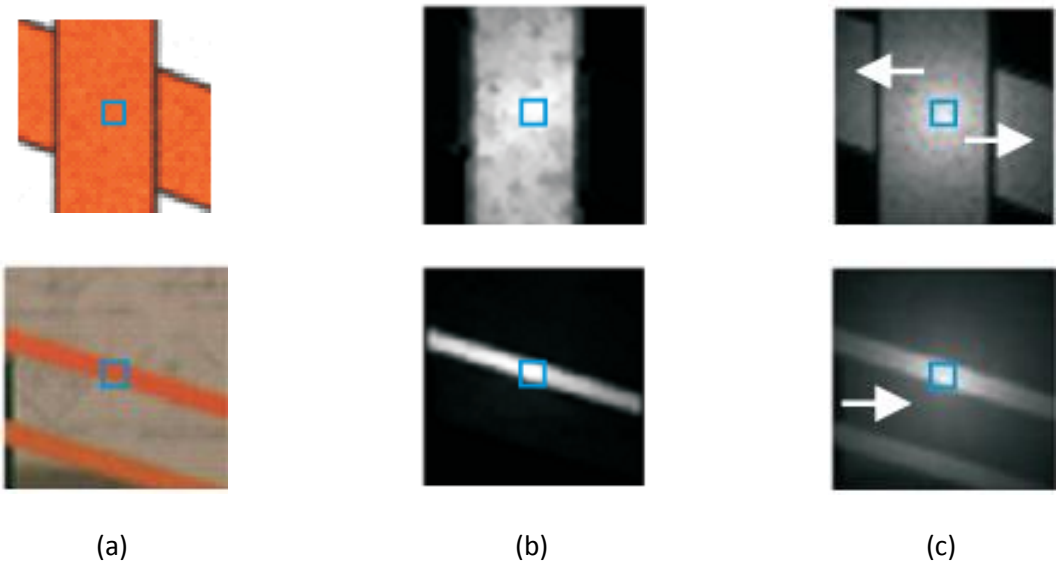
Here, $P_{p,c}$ denotes the set of all possible *paths* between the center pixel c and some other pixel p. A path is defined as a sequence of eight-connected neighboring pixels $\{(x_c, y_c), \dots, (x_p, y_p)\}$ between c and p, and the total distance d for a certain path P_i is calculated according to:

$$d(P_i) = \sum_{k=2}^{n} d_{p1,p2}((x_{k-1}, y_{k-1}), (x_k, y_k)) \tag{2.20}$$

$$d_{p1,p2}(p_1, p_2) = \|C(p_1) - C(p_2)\| \tag{2.21}$$

Again, $C(p) = C(x_p, y_p)$ is a color vector (or an intensity value). Finally, the weights of the support window are set based on the shortest distance found in (2.19):

$$w(p,c) = \exp\left(-\frac{D(p,c)}{\gamma}\right) \tag{2.22}$$



(a) (b) (c)

*Figure 12. Some examples of support windows created by recent adaptive window approaches [**19**]: a) Square window regions in the color images b) Support windows generated by the geodesic color path technique c) Support windows generated by the bilateral filtering approach (the arrows point out differences between the two methods)*

Some results of the two described methods for creating adaptive support windows are shown in Figure 12. A benefit of adapting the window weights is that the local matching can yield more accurate results, while a drawback is that the computational complexity goes up. The computational burden for the geodesic color path technique is very heavy, and the authors are using an approximate method to find the shortest path. To tackle the increased computational burden of these adaptive window techniques, the calculations may possibly be implemented on a GPU as in [20].

For yet another recent local method exploiting color similarity through the use of image segmentation, see Section 3.4.4.

3.1.6 Some words on other measures

The dissimilarity measures and cost aggregation methods mentioned here merely make up a fraction of all approaches that exist. Other variants include normalized measures, such as *normalized cross correlation* (NCC) and *normalized sum of squared differences* (NSSD). Also, there exist numerous local transforms that may be applied on areas surrounding the pixels that are to be compared, such as the *rank transform*, *census transform* and *fourier transform*. However, the need for normalization can be reduced through preprocessing of the images and, according to [2], local transforms rarely seem to give any advantages over the more common difference and gradient based measures.

For further reading, a comparison of some dissimilarity measures and their performance is presented in [21], in which the authors also look into cases when the left and right cameras are biased or have different gain.

3.2 Local and global methods

In stereo research, algorithms often tend to be classified as belonging to a family of either *local methods* or *global methods*. Also, methods in between the two classes exist. The classification relates to the approach used when assigning disparities to the reference image pixels, as will be seen.

3.2.1 Local methods

Local methods work in the way that they only use information located in a close neighborhood of pixels that are compared to each other. A common approach for local methods is to assign the disparity value that minimizes the cost function for each individual pixel coordinate of the reference image. This approach is often referred to as *Winner-Take-All* (WTA) optimization and the process is illustrated in Figure 13 for a pixel on the left edge of the red lamp in the Tsukuba image pair. In the example, a C_{SAD} cost function using a $9x9$ square window centered about the compared pixels was chosen and the disparity search range was set to $0 - 20$ pixels.

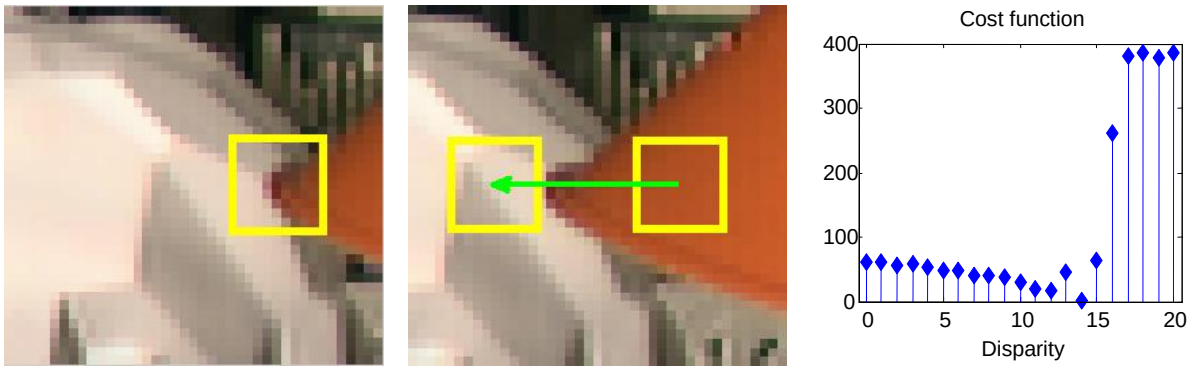


Figure 13. Local matching in the Tsukuba image pair: a) Reference image (left view) b) Search image (right view). Disparities are calculated for pixels along the green line. c) Cost for the different disparities. Note the rise of the cost when the window enters the white area of the sculpture behind the lamp. Clearly, the disparity seems to be 14 pixels for the corresponding pixels in this example.

When using local *Winner-Take-All* optimization, the disparity map is defined as:

$$d(x,y) = \arg\min_{k} C(x,y,k) \tag{2.23}$$

Strong benefits coming with local methods are their simplicity and that they run fast, which make them suitable for real-time implementations. Drawbacks are lack of occlusion handling and high noise sensitivity.

3.2.2 Global methods

In contrast to local methods, global methods aim to determine disparities for all reference image pixels at once. This is achieved by minimizing an *energy function* using some optimization technique[3]. The energy function used in stereo matching usually consists of a correspondence data term and a smoothness term:

[3] The energy terminology is borrowed from the field of statistical mechanics, where similar labeling problems are solved to find the spin of particles [22].

$$E(d) = E_{data}(d) + \lambda \cdot E_{smooth}(d) \tag{2.24}$$

In (2.24), $d = d(x, y)$ is the disparity map for the reference image and λ is a parameter that adjusts smoothness of the result. The data term is normally a sum of costs for a given disparity map, based on some matching cost like the ones presented in Section 3.1:

$$E_{data}(d) = \sum_{(x,y)\in Im} C(x, y, d(x, y)) \tag{2.25}$$

Aggregation of cost is normally skipped when global optimization is used.

The smoothness term is often a function depending on differences in disparity and intensity of neighboring pixels according to:

$$\begin{aligned} E_{smooth}(d) = &\sum_{(x,y)\in Im} \rho\big(d(x, y) - d(x + 1, y), I(x, y) - I(x + 1, y)\big) \\ &+ \rho\big(d(x, y) - d(x, y + 1), I(x, y) - I(x, y + 1)\big) \\ = &\sum_{(i,j)\in N} V_{ij}\big(d_i, d_j\big) \end{aligned} \tag{2.26}$$

Here, N is the set of all four-connected neighboring pixels where a neighboring pixel pair is only included once. The idea is to let ρ increase monotonically with disparity difference to penalize a discontinuous result, but at the same time be able to reduce this penalty for disparity discontinuities located at color edges. This is based on the observation that depth discontinuities often coincide with color/intensity edges in the real world. The sought behavior can be accomplished by letting ρ be a product of two functions:

$$\rho(d, i) = \rho_d(d) \cdot \rho_i(i) \tag{2.27}$$

In (2.27), ρ_d increases with $|d|$ and ρ_i decreases with $|i|$. The function ρ_d is often based on truncated linear disparity difference magnitude:

$$\rho_d(d) = \min\left(|d|, \alpha\right) \tag{2.28}$$

The reason for truncating the penalty is that the result would otherwise be too smooth, especially near object boundaries where large disparity jumps are supposed to be present. Not truncating the penalty could in such areas lead to several smaller jumps. The parameter α is normally set to some fraction of the disparity search range.

The intensity function ρ_i is based on image intensities of adjacent pixels. One example of how this function may look is the following (originally presented in [22]):

$$\rho_i(i) = \begin{cases} c & when \quad |i| < T \\ 1 & when \quad |i| \geq T \end{cases} \tag{2.29}$$

Here, c is a constant with a value of less than 1 and T is a threshold value that determines what is to be considered an intensity edge in the reference image. As desired, the penalty for disparity jumps is reduced at intensity edges.

Although a formulation of the stereo correspondence problem on the form of an energy function was proposed already in the eighties [23], it is not until recently that tractable ways of performing the optimization have been found. One way of solving the task was introduced in 1998 and is based on *graph cuts* [24] (graph cuts and another global energy optimization technique will be treated in the next section).

While computational complexity rises for global methods, benefits compared to local methods are better handling of occlusion and uniform texture and a way to adjust smoothness of the result – i.e. to explicitly make use of the smoothness constraint described in section 2.3.3. Also, global methods are far more tolerant to noise.

3.3 Global optimization techniques

In recent stereo correspondence work, the most commonly used optimization methods for minimization of energy functions are based on *graph cuts* (GC) and *belief propagation* (BP). A promising new algorithm for solving energy minimization problems, originating from belief propagation, is *convergent tree-reweighted message passing* (TRW-S).

The mentioned global optimization techniques have been designed to solve labeling problems and are suitable to minimize energy functions like (2.24). A survey [25] published in 2007 compares several global optimization techniques, both new and older ones, and concludes that the GC and TRW-S techniques are the top performers.

In the following sections, basic principles behind graph cuts optimization and message passing algorithms such as BP and TRW-S will be given.

3.3.1 Graph cuts

A very popular graph cuts energy minimization algorithm was first published in 2001 and is described in [26]. In the paper, two graph formulations that can be used to minimize energy functions of type (2.24) are proposed along with algorithm descriptions. These are based on α, β*-swap moves* and α*-expansion moves*[4]. The text here uses notation from the original paper [26], where pixels are defined as single letters, e.g. p or q, and their assigned labels are written as f_p or f_q.

After a α, β-swap move, the result is an optimal relabeling of pixels labeled α or β before the move. The new pixel labeling is created by leaving labels unchanged, by changing α-labels to β-labels or by changing β-labels to α-labels.

The α-expansion move simply expands the set of pixels with label α in an optimal way so that the energy is minimized.

The described moves are performed in a loop over all labels until the energy stops decreasing. An iteration over all possible labels is referred to as a *cycle* and the order in which labels are visited may either be predetermined or random.

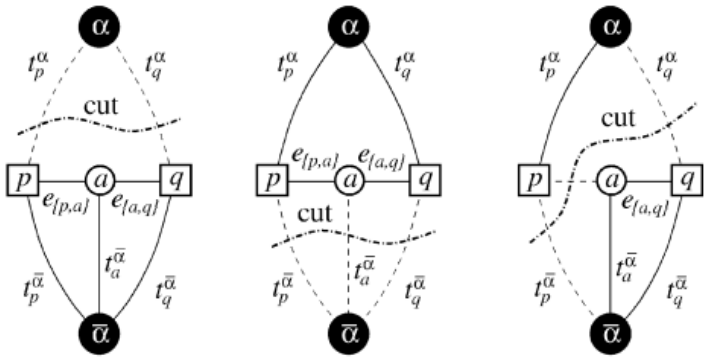
[4] A move is the terminology for relabeling a subset of pixels

(a) (b) (c)

Figure 14. Examples of results from moves in the graph cuts algorithms of [26]: a) Original pixel labeling, three labels are present b) How the result might look like after a α, β-swap move. Note that only pixels with labels α and β are affected by the move. c) How the result might look like after a α-expansion move. Here, pixels having any label in the original labeling might get assigned α as its new label.

In order to perform a move, the energy function is formulated as a graph problem for the considered label α (for expansion) or labels $\{\alpha, \beta\}$ (for swap). The new labeling after the move is found by computing the *minimum cut* of the defined graph, often simply referred to as the *graph cut*.

Of the two moves, the α-expansion is strongly favored in the survey [25] and also tends to occur more frequently than the α, β-swap move in stereo algorithms. For this reason, only the expansion move will be discussed hereafter.



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

(a) (b)

Figure 15. A small two pixel example of the graph structure used to find a α-expansion move in graph cuts optimization. [26] a) Pixels are interconnected via edges modeling the smoothness costs $V(f_p, f_q)$. All pixels are also connected to the terminal nodes via edges that specify the data costs. A cut in the graph separates the pixels in two sets – one keeps its previous labels and one is relabeled with the new label α.The energy is calculated by summing costs over the cut edges. b) The edge weights for the example graph in (a)

The graphs for computing an expansion move consist of a set of nodes, each representing a pixel. When the energy function is of the form (2.24) and formulated on a pixel level, the resulting graphs will be two dimensional grid graphs.

Two terminal nodes are introduced in the graphs and connected to all pixel nodes via edges called *t-links*. A cut over the edge between a pixel and the $\overline{\alpha}$-terminal means that the pixel keeps its original label, and a cut over the edge connecting the pixel to the α-terminal means that the pixel is assigned α as label.

For pixels with some label f_p different from α prior to the move, the edges are weighted with a data cost retrieved from the cost function. In the example graph shown in Figure 15, the cost for assigning a pixel p its original label f_p is written $D_p(f_p)$ and the cost for assigning α as the new label is $D_p(\alpha)$[5].

The cost for edges between the $\overline{\alpha}$-terminal node and pixel nodes already having the label α prior to the expansion move is set to infinity to assure that these pixel nodes keep their α-label throughout the optimization.

All pixel nodes are also interconnected via edges called *n-links* and nodes called *auxiliary nodes* (the circles in Figure 15 (a) that containin an '*a*'). The edges between pixel nodes and auxiliary nodes are weighted with a cost computed from the smoothness energy term in (2.24). The auxiliary nodes are also connected to the $\overline{\alpha}$-terminal with an edge cost set to the smoothness cost between its neighboring pixels when their labels are left unchanged. The notation $V(f_p, f_q)$ in the figure is similar to the notation of (2.26) (where neighboring pixels are called i and j, though). By looking at Figure 15, things should become clearer.

With this graph configuration, solving for the optimal move equals solving for the minimum cut of the graph, which is done by computing the *maximum flow* between the terminals. This duality, referred to as the *max-flow min-cut* theorem, was originally presented in [27]. The maximum flow problem is well known and many efficient algorithms to solve it have been developed over the years [28].

The described graph cuts algorithms solve binary labeling problems exactly and multilabel problems approximately. However, it is proven in [26] that the solution is always a local minimum close to the global minimum. In practice, the resulting energy is often in fact only fractions of a percent larger than the global optimum energy [25].

3.3.2 Belief propagation algorithms

Belief propagation is an algorithm that was mainly developed to find marginal probabilities in Bayes networks. However, the algorithm can also handle other graphical models such as *Markov Random Field* (MRF) models which are of certain interest in the optimization of global energy functions found in computer vision.

A MRF is an undirected graph model in which nodes represent random variables. An important property of a *pair-wise* MRF is that a node variable is conditionally independent of all other variables given its neighbors. The joint probability $P(x_1, \dots, x_n)$ of a pairwise MRF model may therefore be written in a decomposed form as:

[5] To use the earlier presented notation for cost, we have $D_p(f_p) = C(x_p, y_p, f_p)$, with $p = (x_p, y_p)$.

$$P(x_1, \dots, x_n) = \frac{1}{Z}\prod_i \phi_i(x_i, y_i) \prod_{i,j \in N} \psi_{ij}(x_i, x_j) \tag{2.30}$$

Here, x_n represents the nodes of a graph and N describes node neighborhoods defined through pair-wise dependencies between node variables. The *potential* $\phi_i(x_i, y_i)$ represents the probability for a certain state $x_i \in \chi_i$ in node i based on an observation y_i and the potential $\psi_{ij}(x_i, x_j)$ denotes the conditional dependency between neighboring nodes. The observation variable y_i is rarely written out explicitly. [29]

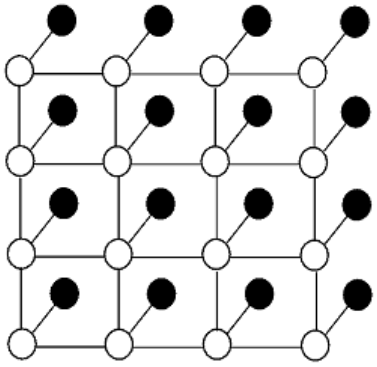


Figure 16. An example of a two dimensional grid MRF [29]. The white circles are the graph nodes x_i and the black circles are observations y_i tied to the nodes. The node variables in this example could for instance represent disparities of image pixels, while the observation variables could be matching cost.

The usefulness of this probability approach comes from that an energy function $E(d)$ may be described as a two dimensional grid graph where nodes are four-connected to each other. The data cost function sets the individual node probabilities and the conditional probabilities are defined through the pair-wise energy terms. The connection between an energy function and the joint probability of a pair-wise MRF can be described as:

$$E(d) = E_{data} + E_{smooth} = \sum C\big(x, y, d(x, y)\big) + \sum V_{ij}\big(x_i, x_j\big) \tag{2.31}$$

$$P(d) = \frac{1}{Z}\exp(-E) = \frac{1}{Z}\prod \mathrm{e}^{-C(x,y,d(x,y))} \prod \mathrm{e}^{-V_{ij}(x_i, x_j)} \tag{2.32}$$

Here, Z is a normalization constant that makes the sum of all probabilities equal to one. From the relations, it is seen that minimizing the energy function E corresponds to maximizing the joint probability P, i.e. finding the most probable disparity field $d = d(x, y)$.

Belief propagation comes in two versions, with the *sum-product* version targeting to find all marginal probabilities and the *max-product* version aiming to find the most probable solution (i.e. having the lowest energy). Here, sum-product version will be explained briefly through a small *loop free* example (i.e. for a graph without cycles).

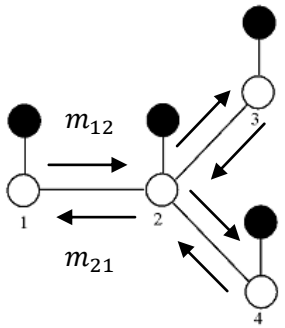


Figure 17. An example of a loop free MRF [29]: Messages are sent back and forth between the nodes as shown by the arrows.

Message passing is central to belief propagation algorithms. Basically, message passing is about neighboring nodes sending messages containing probabilities for the available node states back and forth to each other. A message $m_{ij}(x_j)$ from node i to node j is a vector of the same size as the number of available states for node j, where each component of the vector is proportional to how likely node i thinks that node j should be in the corresponding state. Messages are used to calculate node *beliefs*[6]. Message updates and beliefs are computed by the following equations:

$$m_{ij}^{t}(x_j) = \sum_{x_i} \phi_i(x_i)\psi_{ij}(x_i, x_j) \prod_{n \in N \setminus j} m_{ni}^{t-1}(x_i) \tag{2.33}$$

$$b_i(x_i) = k \cdot \phi_i(x_i) \prod_{n \in N} m_{ni}^{T}(x_i) \tag{2.34}$$

$N \setminus j$ denotes the neighboring nodes of i, except node j, and m_{ni}^{T} are the final messages after T iterations. The constant k makes the sum of beliefs equal to one.

For node 1 in the example graph shown in Figure 5, the belief is calculated according to:

$$b_1(x_1) = k \cdot \phi_1(x_1) \cdot m_{21}^{2}(x_1) \tag{2.35}$$

The message vector m_{21}^{2} is computed in the following way:

$$m_{21}^{2}(x_1) = \sum_{x_2} \phi_2(x_2)\psi_{21}(x_1, x_2) m_{32}^{1}(x_2) m_{42}^{1}(x_2) \tag{2.36}$$

$$m_{32}^{1}(x_2) = \sum_{x_3} \phi_3(x_3)\psi_{32}(x_3, x_2) m_{23}^{0}(x_3) \tag{2.37}$$

$$m_{42}^{1}(x_2) = \sum_{x_4} \phi_4(x_4)\psi_{42}(x_4, x_2)\, m_{24}^{0}(x_4) \tag{2.38}$$

The messages $m_{ij}^{0}(x_j)$ are initialized to one. If substituting the messages into equation (3.20), the result will be the marginal probability:

[6] Belief is synonymous to marginal probability

$$b_1(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4} k\phi_1(x_1)\,\phi_2(x_2)\psi_{21}(x_1,x_2)\phi_3(x_3)\psi_{32}(x_3,x_2)\phi_4(x_4)\psi_{42}(x_4,x_2)$$
$$= \sum_{x_2}\sum_{x_3}\sum_{x_4} p(x_1,x_2,x_3,x_4) = p(x_1) \qquad (2.39)$$

Before starting the message passing algorithm, the messages have to be initialized somehow. One way is to assign random values to the message vector components and another way is to assign equal values to all components, e.g. ones. Also, the message passing order has to be defined. The starting node has to be specified, as well as a schedule for directions of the message passing. For trees it is common to pass messages from the root to the leaves, and back (the root may be selected freely). For grid graphs, one schedule for passing messages is left-right-up-down.

The main benefit with message passing is that marginal probabilities can be computed for large graphs with a computational burden linearly proportional to the number of nodes. The messages contain global information, which is used for local computations and then passed on. For trees and graphs without loops, BP algorithms give exact results with two message passes.

When loops may be present, as in grid MRF models, there is a risk of messages circulating back and forth infinitely. One way of dealing with this is to keep track of the energy and terminate the algorithm when its value starts to oscillate. Algorithms based on *generalized belief propagation* can handle loopy graphs more robustly, but the result from belief propagation on loopy graphs is always *approximate*. [29]

A numerous amount of algorithms derived from the belief propagation concept exist. The *loopy belief propagation* algorithm suggested by [30] and the *convergent tree-reweighted message passing* of [31] are popular versions in today's computer vision applications (the TRW-S algorithm differs in that it puts different weights on the incoming messages $m_{ki}^{t-1}(x_i)$ in the message update equation (2.33)). For more information on belief propagation algorithms, please refer to [29].

3.4 Image segmentation and its usages in stereo algorithms

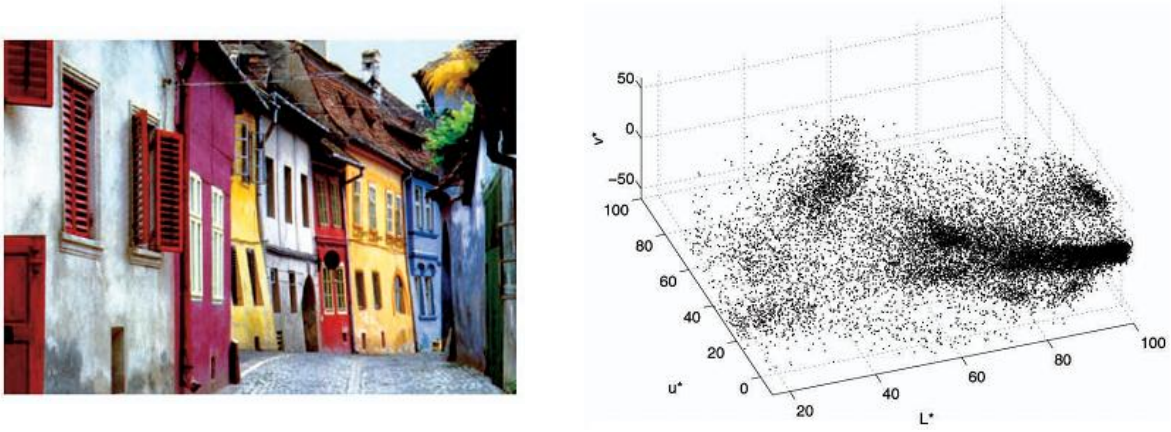
Image segmentation is a technique which has been successfully introduced into stereo correspondence algorithms during the last decade. The goal of image segmentation is to reduce the number of colors in the input reference image and then group neighboring pixels of similar color together to form bounded segments. The *mean-shift* color segmentation technique, used in many modern stereo algorithms, has proven to be a good way of dividing an image into regions with borders that coincide with object boundaries.

3.4.1 Mean-shift image segmentation

Mean-shift segmentation is a *nonparametric*[7] *mode*[8] *finding* technique. In [32], the technique is used to perform *feature space analysis* of the color space belonging to an image, which is done with the purpose to segment the image into delineated clusters. Image data is modeled as samples from a multivariate distribution with unknown probability density function (p.d.f.) and the goal is to find its modes [17].

[7] *Non-parametric* methods make no assumptions on the statistical properties of analyzed data

[8] A *mode* is a local maximum of the probability density function



(a) (b)

Figure 18. A color image (a) with its corresponding Luv color space (b). [32]

In Figure 18 above, a color image is shown together with its corresponding *Luv color space*. In contrast to the RGB space, pixels are described by *lightness* and *chrominance* components, i.e. their intensity and color. The authors of [32] model the Luv space data points as samples from a random multivariate distribution, and to find its modes they take an approach based on *kernel density estimation*.

In statistics, kernel density estimation is a way to estimate an unknown p.d.f. given a number of data samples from the distribution. The *univariate kernel density estimator* will serve as an example to explain the underlying principle, and is formulated as:

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K(x - x_i) \tag{2.40}$$

A common kernel is the Gaussian kernel, which is expressed as:

$$K(x) = \frac{1}{\sqrt{2\pi}h}\exp\left(-\frac{x^2}{2h^2}\right) \tag{2.41}$$

Here, h is a parameter that controls the bandwidth of the kernel (which is the variance for a Gaussian kernel). A density estimate created with this approach is illustrated in Figure 19 for a case with six data points originating from some random variable with a univariate distribution. Note that three modes can clearly be distinguished.

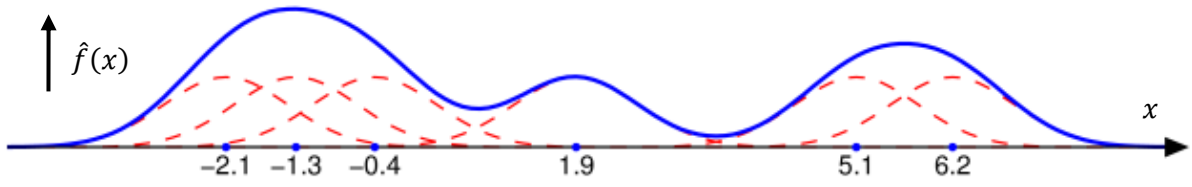


Figure 19. Example of kernel density estimation using a Gaussian kernel with bandwidth (variance) $h = 0.5$ [33]: The red dashed lines are the individual contributions from each kernel at the data points in the estimate's sum. The blue line is the total sum of all kernels, i.e. $\hat{f}(x)$.

In the case where there is a set of d-dimensional data points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$, the authors of [32] use a simplified *multivariate kernel density estimator* given by:

$$\hat{f}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} K_h(\boldsymbol{x} - \boldsymbol{x}_i), \qquad K_h(\boldsymbol{x}) = \frac{c}{h^d} k\left(\frac{\|\boldsymbol{x}\|^2}{h^2}\right) \tag{2.42}$$

The kernel $K_h(\boldsymbol{x})$ is shaped by a *kernel profile* $k(x)$, defined for $x \geq 0$, and the normalization constant c is set so that $\int_{\mathbb{R}^d} K_h(\boldsymbol{x})d\boldsymbol{x} = 1$ holds. The *Epanechikov* kernel is desirable in segmentation tasks, but a normal kernel could also be used [32]. Their corresponding profiles are:

$$k_E(x) = \max(0,1-x), \mathrm{x} \geq 0 \tag{2.43}$$

$$k_N(x) = \exp\left(-\frac{1}{2}x\right), x \geq 0 \tag{2.44}$$

Having defined the estimator, the target is now to find the local maxima of the density. This can be accomplished without calculating the density itself, but by instead computing its gradient:

$$\nabla\hat{f}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} \frac{2c}{h^{d+2}}(\boldsymbol{x} - \boldsymbol{x_i})k'\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|^2}{h^2}\right) \tag{2.45}$$

As we are searching for maxima, normalization constants may be disregarded, and the gradient expression can be rewritten in the following way:

$$g(x) = -k'(x), \qquad G(\boldsymbol{x}) = -k'\left(\frac{\|\boldsymbol{x}\|^2}{h^2}\right) \tag{2.46}$$

$$\begin{aligned}\nabla f(\boldsymbol{x}) &= \sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{x})G(\boldsymbol{x}_i - \boldsymbol{x}) \\ &= \left[\sum_{i=1}^{n} G(\boldsymbol{x} - \boldsymbol{x}_i)\right]\left[\frac{\sum_{i=1}^{n} \boldsymbol{x_i} G(\boldsymbol{x} - \boldsymbol{x_i})}{\sum_{i=1}^{n} G(\boldsymbol{x} - \boldsymbol{x_i})} - \boldsymbol{x}\right]\end{aligned} \tag{2.47}$$

Out of this expression the *mean-shift vector* is defined as:

$$\boldsymbol{m}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} \boldsymbol{x_i} G(\boldsymbol{x} - \boldsymbol{x_i})}{\sum_{i=1}^{n} G(\boldsymbol{x} - \boldsymbol{x_i})} - \boldsymbol{x} \tag{2.48}$$

Thus, the mean-shift vector is a weighted mean value of data points, computed in $\boldsymbol{x}$. As seen, it always points in the direction of the gradient, i.e. towards local maxima of the density estimate. This result was first presented in [34]. Popular choices of kernel profiles result in the function $g(x)$ being a unit ball or a Gaussian [17].

For the use of the mean-shift technique in image segmentation, it is often not meaningful to cluster together pixels of similar color that are located spatially far apart. Therefore, the used kernel could also include a spatial term in order to prevent this:

$$K_{h_s,h_r}(\boldsymbol{x}) = \frac{c}{h_s^2 h_r^p} k\left(\left\|\frac{\boldsymbol{x}^s}{h_s}\right\|^2\right) k\left(\left\|\frac{\boldsymbol{x}^r}{h_r}\right\|^2\right) \tag{2.49}$$

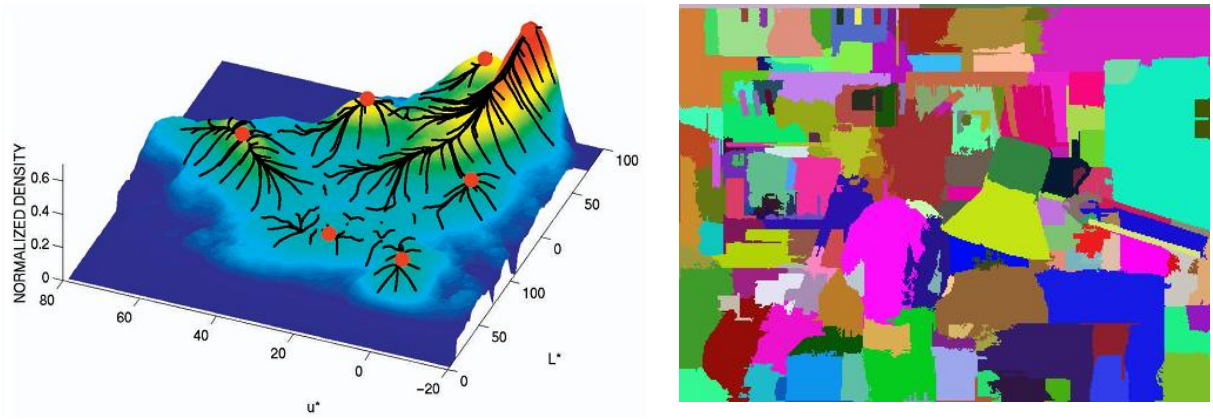
The feature vector $\boldsymbol{x}$ now contains a spatial part $\boldsymbol{x}^s$ and a range part $\boldsymbol{x}^r$, where the spatial part is a two-dimensional vector and the color range part is normally one or three-dimensional depending on whether a color or grayscale image is segmented. As the spatial range and color range may have different scales, two bandwidth parameters are used.

Selecting the bandwidth of the mean-shift algorithm is not trivial, and there have been some attempts made to find out good ways on how to do this [32] [17]. However, the algorithm is quite insensitive when it comes to the bandwidth selection and it is often possible to achieve a decent result from using only one fixed parameter set.

The segmentation is done by starting the mean-shift in all available data points $\boldsymbol{x}_i$ and move in the direction of the mean-shift vector until convergence according to:

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \boldsymbol{m}(\boldsymbol{y}_k), \qquad \boldsymbol{y}_0 = \boldsymbol{x}_i, \qquad i = 1, \dots, n \tag{2.50}$$

All data points leading to convergence in the same mode of the p.d.f. are then clustered together to form bounded segments. Clusters belonging to local convergence points located closer to each other than the specified bandwidths are concatenated, and remaining small clusters consisting of less than a threshold of M pixels may be merged if desired.



(a) (b)

Figure 20. Mean-shift segmentation: a) The modes found in the Luv feature space belonging to the color image in Figure 18 are shown as red dots [32]. b) Segmenting the Tsukuba image with $(h_s, h_r, M) = (15,15,40)$ *results in 164 different segments, here randomly colored for convenience.*

3.4.2 Plane fitting of disparities within segments

The underlying idea of performing plane fitting within segments is based on the assumption that depth varies continuously almost everywhere and that depth discontinuities occur primarily at color edges (see section 2.3.3). By fitting planar surfaces in segments of homogenous color, local piece-wise smoothness is forced.

Regarding plane fitting, robust methods have to be used in order to get a satisfactory result. As the initial disparity estimate which provides data points for the plane fitting is often contaminated with

noise, least squares methods are not directly applicable due to their *outlier*[9] sensitivity. Techniques such as *Random Consecutive and Sample* (RANSAC) [35] can be used prior to least squares to remove outliers from the model fitting step, as they would otherwise introduce a bias. Recently, techniques based on histogram analysis of slope values found from data points have also been proposed in stereo algorithms (such as [36] and [37]).

The RANSAC algorithm

The RANSAC algorithm was originally presented in [38]. RANSAC alternates between two steps – hypothesizing and testing – and to do this, a model definition is needed along with a function that returns the distance between a data point and the model.

The hypothesis step consists of randomly selecting the minimum amount of sample points required to fully define the model parameters, i.e. three points for a plane, and then the model is fitted using these points. The testing step measures the distance between all other data points and the model, and points located within a certain distance threshold are considered to be inliers, while the rest are defined as outliers. Finally, the model giving the highest rank of inliers is assumed to be the most representative one. As an optional step, the model can be refitted using all the found inliers.

In the case of plane fitting, the used model is the well known plane equation and the distance function is the orthogonal projection of a line drawn from a point on the plane to some data point $\boldsymbol{d}_i = (x_i, y_i, z_i)$. The model manifold $\boldsymbol{\mathcal{M}}(\boldsymbol{\theta})$ and the distance function e are defined as:

$$\boldsymbol{\mathcal{M}}(\boldsymbol{\theta}) = \{\boldsymbol{d} \in \mathbb{R}^3 : \theta_1 x + \theta_2 y + \theta_3 z + \theta_4 = (\boldsymbol{d}\ 1) \cdot \boldsymbol{\theta} = 0, \|\boldsymbol{\theta}\| = 1\} \tag{2.51}$$

$$e\big(\boldsymbol{d}, \boldsymbol{\mathcal{M}}(\boldsymbol{\theta})\big) = \frac{[(\boldsymbol{d}\ 1) \cdot \boldsymbol{\theta}]}{\|\boldsymbol{\theta}_{1:3}\|} \tag{2.52}$$

Let the input data set be defined as $D = \{\boldsymbol{d}_1, \dots, \boldsymbol{d}_N\}$. For a plane $\boldsymbol{\mathcal{M}}(\boldsymbol{\theta})$ created from three randomly selected data points, the *consensus set* (CS) defines the set of all data points that can properly be modeled by this plane:

$$S(\boldsymbol{\theta}) = \{\boldsymbol{d} \in D : e\big(\boldsymbol{d}, \boldsymbol{\mathcal{M}}(\boldsymbol{\theta})\big) < \delta\} \tag{2.53}$$

The threshold parameter δ can either be specified directly from the nature of the problem, or calculated by specifying the variance σ_e of the expected errors $e\big(\boldsymbol{d}, \boldsymbol{\mathcal{M}}(\boldsymbol{\theta})\big)$ for *inliers*, assuming they are independent Gaussian variables (see [35] for details).

Several ways to rank a consensus set exist. In the original RANSAC proposal, the ranking was based on the number of elements in S, and the model parameters giving the largest consensus set were considered the best. Other approaches exist, e.g. MSAC, which also regards the actual error values [39]. These two ranking methods can be compared against each other by formulating them as cost minimization problems:

[9] An outlier is a data point which deviates too much from a specified model to be considered as a sample from it.

$$\boldsymbol{\theta} = \arg\min_{\|\boldsymbol{\theta}\|=1} C_{\boldsymbol{\mathcal{M}}(\boldsymbol{\theta})} = \arg\min_{\|\boldsymbol{\theta}\|=1} \sum_{\boldsymbol{d}\in D} \rho(\boldsymbol{d}_i) \tag{2.54}$$

$$\rho_{RANSAC}(\boldsymbol{d}) = \begin{cases} 0, & e\big(\boldsymbol{d}, \boldsymbol{\mathcal{M}}(\boldsymbol{\theta})\big) < \delta \\ 1, & otherwise \end{cases} \tag{2.55}$$

$$\rho_{MSAC}(\boldsymbol{d}) = \min\left(e\big(\boldsymbol{d}, \boldsymbol{\mathcal{M}}(\boldsymbol{\theta})\big), \delta\right) \tag{2.56}$$

The main benefit of the MSAC approach to other ranking methods is that no additional computation is needed.

In RANSAC, the number of iterations is a self-adapting quantity which depends upon two variables, namely the probability q of finding a good sample set for the initial model creation and the *alarm rate probability threshold* ϵ which specifies the maximum allowed probability to only pick bad sample sets. The algorithm is run until the number of iterations T results in $(1-q)^T \leq \epsilon$. How to estimate the unknown probability q and T is discussed further in [35].

Plane fitting based on x and y slant values

The methods of plane fitting found in the well performing, recent stereo algorithms [36] and [37] are based on calculating slant values for every pixel pair within an image segment. The calculation of horizontal and vertical slants is divided into two separate steps. In the papers belonging to the stereo algorithms, the technique is not described very well, so only a vague description is possible to give here.

First, for every pixel pair (x_i, x_j) on the same line, having the disparities (d_i, d_j), the slant value $s = (d_j - d_i)/(x_j - x_i)$ is calculated. For a line having N pixels, $N(N-1)/2$ slant values are computed and inserted into a list. This procedure is performed for every horizontal and vertical line of pixels within segments. The found slant values are then processed in order to find the most representative value of how the plane is slanting in the x and y direction.

The authors of [36] merely mention that they insert the values into a list, which is sorted. After that, they perform convolution with a Gaussian kernel to find an estimate. No information is given on filter parameters or which criteria that is used for selecting the best slant value.

In [37], the selection of the winning slants is described as the result from creating a histogram of the occurring slant values followed by Gaussian filtering of the histogram peaks. However, nothing is said regarding how filter parameters such as length and standard deviation are set. Filtering a histogram with a Gaussian kernel resembles the kernel density estimation technique described in Section 3.4.1.

This plane fitting approach will be investigated more closely in the experimental part of this project in hope of finding a successful plane fitting algorithm based on calculating horizontal and vertical slant values.

3.4.3 Setting smoothness terms based on segmentation results

Segmentation information can be used in global optimization to make disparity discontinuities coincide with segment borders. For an energy function of type $E(d) = E_{data}(d) + E_{smooth}(d)$, the smoothness term can be expressed as:

$$E_{smooth}(d) = \sum_{(i,j)\in N} V_{ij}(d_i, d_j) \tag{2.57}$$

$$V_{ij}(d_i, d_j) = \begin{cases} \lambda \cdot \rho_d(d_i - d_j) & for\ s_i = s_j \\ \gamma \cdot \lambda \cdot \rho_d(d_i - d_j) & for\ s_i \neq s_j, \quad with\ 0 < \gamma < 1 \end{cases} \tag{2.58}$$

This means that when two neighboring pixels i and j belong to the same image segment $s_i = s_j$, the smoothness parameter is set to λ. When the pixels are located in separate segments, the cost is lowered through multiplication of the parameter with a constant $\gamma \in (0,1)$, thus encouraging disparity jumps at segment borders.

3.4.4 Adaptive support-weights in cost aggregation

Various techniques using segmentation results for cost aggregation have been presented. The concept is used to generate an initial disparity estimate in e.g. [37] where the technique proposed in [40] was implemented.

The cost aggregation of [40] is done by using a rather large $51x51$ pixel square window, and weighting pixels differently whether they belong to the same segment as the center pixel or not. Pixels located in the same segment as the center pixel are weighted with 1, and pixels outside are weighted with a small value δ. In the paper describing the technique, this value was 0.01.

3.5 Handling of occlusion and outliers

Firstly, occlusion is implicitly dealt with in stereo algorithms. One example is the replacing of noisy disparity values though the use of plane fitting. However, there are also several methods developed to explicitly handle the occlusion problem. Among these methods, *left-to-right and right-to-left cross-checking* is considered to be both simple and effective relative to other approaches [2].

Another way to measure reliability is to look at the cost function and determine the difference between the winning disparity cost and the second lowest cost [41]. Moreover, cost truncation can also help to improve the result at disparity discontinuities, where occlusion is a problem. Nothing speaks against using several approaches in conjunction with each other to handle occlusion.

3.5.1 Cross checking between left and right disparity estimates

Cross-checking between left and right images is carried out by estimating two disparity maps, one for each view of the input stereo image pair. Warping of the disparity values in these to their expected location in the other view is then performed. Reliability of disparity values is determined by comparing disparities of the originally estimated maps for each view against the disparities in the warped versions originating from the other view. Pixels fulfilling the equalities below are consistent and therefore not considered to be occluded:

$$d_{LEFT}(x, y) = d_{RIGHT}(x - d_{LEFT}(x, y), y) \tag{2.59}$$

$$d_{RIGHT}(x, y) = d_{LEFT}(x + d_{RIGHT}(x, y), y) \tag{2.60}$$

Pixels with disagreeing disparity values in the above comparisons are considered to be unreliable. The disagreement may very well be explained by occlusion, but this method also handles mismatches resulting from other reasons such as noise or lack of texture. Unreliable pixels can be excluded from

contributing to the final result, which may improve the quality (disparity in their positions can be determined by e.g. plane fitting).

3.5.2 Comparing winning disparity cost against second lowest matching cost

This approach is for instance used in [41], where pixels are marked as *stable* or *unstable* based on the outcome of the comparison between the lowest and second lowest matching cost value. Unstable pixels are treated differently from stable ones so that their effect on the final result is limited.

Let the lowest cost for some coordinate (x_i, y_i) be denoted $C_i^0 = C(x_i, y_i, d_0)$ and the second lowest cost for the same pixel be referred to as $C_i^1 = C(x_i, y_i, d_1)$. The pixel in consideration can then be classified as stable or unstable by calculating the following quantity:

$$\frac{C_i^1 - C_i^0}{C_i^1} \tag{2.61}$$

If the quantity (2.61) has a value over a certain threshold, the pixel is labeled as stable, and otherwise it is classified as unstable.

3.5.3 Cost truncation

When aggregating cost, the result may be affected by outlier pixels. If the aggregated cost of a correct disparity value that *would* have been chosen in WTA optimization if the images were noise free is influenced by one or more outliers, another disparity value may be chosen instead. This problem is especially apparent in disparity discontinuous areas, where foreground and background pixels are present in the aggregation window at the same time [1].

Cost truncation might also help for global optimization. Think of an unreliable pixel with a high data cost for all disparity values. If the consistently high data costs for an unreliable pixel are truncated and limited to that all take the same value, the disparity of this pixel will instead be guided by the smoothness cost. Thus, the disparity of the pixel will instead be propagated from more reliable neighboring pixels.

Truncating the cost function reduces the influence of outliers, and ideally the cost should be truncated right above the noise level. The truncation operation is defined as:

$$C(x, y, d) = \min\left(C(x, y, d), \alpha\right) \tag{2.62}$$

In [36], the threshold α is set by using reliable disparity estimates which have been validated by cross checking. The value is set slightly above the actual cost values of reliable estimated disparities. However, no details are given on the implementation and how the threshold is determined.

A straightforward implementation of cost truncation is to manually set the threshold α. However, the parameter is highly dependent on the input image characteristics, such as noise level. Thus, the parameter either has to be set manually for every used image pair, or set to a rather high value, which lowers the improvement of the result.

4 Survey of modern stereo correspondence algorithms

4.1 Introduction to the survey

A lot of research has been done on stereo matching and, as a result, many approaches to tackle the problem exist. This survey aims to describe approaches of recent algorithms that are *known* to perform well on benchmarking stereo image sets. Also, techniques found in *Depth Estimation Reference Software* (DERS) which is used in the MPEG activities will be explained.

The performance evaluation table of the *Middlebury Stereo* homepage [4] has been used in the process of stereo algorithm selection (see Figure 21 below). The selected algorithms have been studied more thoroughly and inspired the contents of this chapter. They have also influenced the selection of theory in the previous chapter.

Algorithm	Avg. Rank	Tsukuba ground truth nonocc	Tsukuba ground truth all	Tsukuba ground truth disc	Venus ground truth nonocc	Venus ground truth all	Venus ground truth disc	Teddy ground truth nonocc	Teddy ground truth all	Teddy ground truth disc	Cones ground truth nonocc	Cones ground truth all	Cones ground truth disc	Average percent of bad pixels (explanation)
AdaptingBP [17]	4.5	1.11 9	1.37 5	5.79 11	**0.10** 1	0.21 3	1.44 2	4.22 3	7.06 4	11.8 4	2.48 2	7.92 7	7.32 3	4.23
CoopRegion [41]	4.5	**0.87** 1	**1.16** 1	**4.61** 1	0.11 2	0.21 2	1.54 4	5.16 10	8.31 7	13.0 7	2.79 6	7.18 4	8.01 9	4.41
DoubleBP [35]	5.9	0.88 3	1.29 2	4.76 3	0.13 5	0.45 12	1.87 8	3.53 2	8.30 6	**9.63** 1	2.90 7	8.78 16	7.79 6	4.19
OutlierConf [42]	7.0	0.88 2	1.43 7	4.74 2	0.18 11	0.26 7	2.40 14	5.01 6	9.12 10	12.8 6	2.78 5	8.57 12	6.99 2	4.60
SubPixDoubleBP [30]	9.2	1.24 16	1.76 16	5.98 12	0.12 4	0.46 13	1.74 7	**3.45** 1	8.38 8	10.0 2	2.93 9	8.73 15	7.91 8	4.39
WarpMat [55]	11.3	1.16 10	1.35 4	6.04 13	0.18 12	0.24 5	2.44 15	5.02 7	9.30 11	13.0 9	3.49 17	8.47 11	9.01 22	4.98
Undr+OvrSeg [48]	14.8	1.89 37	2.22 33	7.22 29	0.11 3	0.22 4	**1.34** 1	6.51 16	9.98 12	16.4 19	2.92 8	8.00 8	7.90 7	5.39
GC+SegmBorder [57]	15.6	1.47 28	1.82 18	7.86 34	0.19 13	0.31 8	2.44 15	4.25 4	**5.55** 1	10.9 3	4.99 45	**5.78** 1	8.66 17	4.52
AdaptOvrSegBP [33]	16.7	1.69 31	2.04 28	5.64 9	0.14 6	**0.20** 1	1.47 3	7.04 27	11.1 15	16.4 21	3.60 21	8.96 19	8.84 19	5.59
GeoSup [64]	17.8	1.45 27	1.83 20	7.71 33	0.14 7	0.26 6	1.90 9	6.88 24	13.2 29	16.1 16	2.94 10	8.89 18	8.32 14	5.80

Figure 21. The top ten dense stereo matching algorithms listed in the evaluation table on the Middlebury Stereo homepage [4].

A majority of the studied stereo correspondence algorithms perform the following steps:

- Image segmentation
- Computation of a cost function
- Disparity computation
- Refinement of computed disparity

First of all, some background on how to evaluate performance will be presented. After that, each of the selected algorithms will be treated more thoroughly under its own subsection.

To learn about stereo correspondence concepts that have been left out of this survey, both well known and more obscure techniques are described in [1], [2] and [3]. Also, the Middlebury Stereo homepage [4] is a good source of algorithms, now containing more than 80 submitted results.

4.2 Performance evaluation of algorithms

The authors behind the taxonomy [1] and the Middlebury Stereo homepage have developed an objective performance measure that compares pixel subsets of disparity maps coming from different algorithms against the same pixel subsets in ground truth data.

The subsets are *non-occluded pixels* ($\mathcal{N}$), *all pixels* ($\mathcal{A}$) and *disparity discontinuity pixels* ($\mathcal{D}$). They are determined by performing warping of ground truth disparity maps or by selecting pixels near discontinuities where the change in disparity is larger than a specified threshold value. In Figure 22 below, the subsets as computed for the *Tsukuba* stereo images are shown.

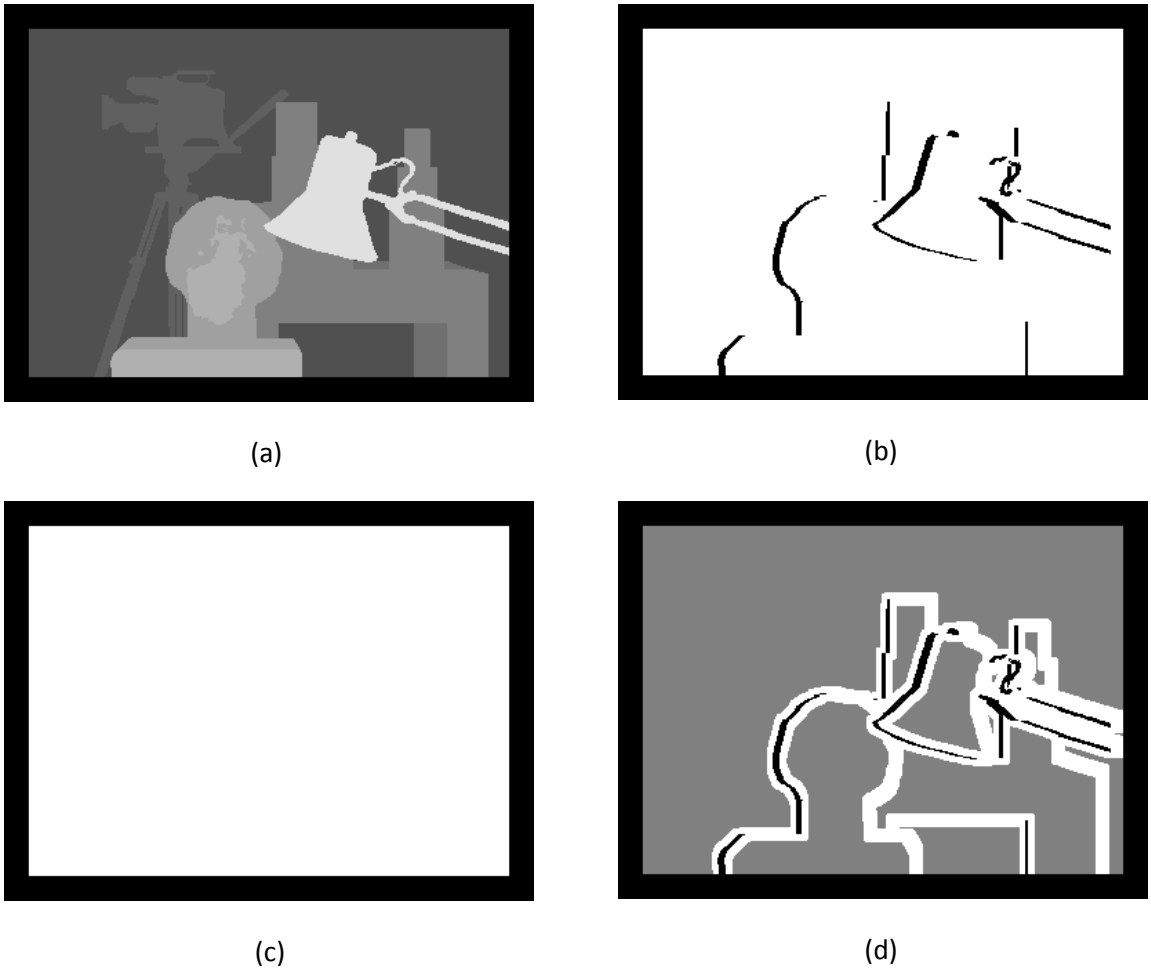


Figure 22. Examples of the pixel subsets used in performance evaluation of stereo algorithms: a) Ground truth b) Non-occluded pixels c) All pixels d) Pixels of discontinuous regions. Errors are evaluated in white areas.

The quantitative measure that has been used to evaluate performance of algorithms is the *percentage of bad pixels* in subsets of estimated disparity maps, which is defined as:

$$B_{\mathcal{S}} = \frac{1}{N_{\mathcal{S}}} \sum_{(x,y)\in\mathcal{S}} \left| d_{estimate}(x,y) - d_{groundtruth}(x,y) \right| > \delta \qquad (4.1)$$

Here, $\mathcal{S}$ denotes the desired subset of pixels. $N_{\mathcal{S}}$ is the number of pixels in $\mathcal{S}$ and δ is the error tolerance. Throughout this project, a tolerance δ of 1.0 has been used as this corresponds to the value used in [1] and several other studies.

Later in the report, the percentage of bad pixels for the three subsets will simply be denoted $\mathcal{N}$, $\mathcal{A}$ and $\mathcal{D}$ for non-occluded pixels, all pixels and disparity discontinuous pixels (instead of e.g. $B_{\mathcal{N}}$ – for simplicity).

4.3 Selected algorithms

To select algorithms to look further into, the performance comparison table of [4], shown in Figure 21, was used. Algorithms in the table are sorted by their average rank based on their performance for the four stereo image benchmarking sets presented in Section 2.1.3. The top five algorithms are looked closer into in this survey.

As the algorithms are described in a belonging paper often having a long title, they are abbreviated here when referred to. The abbreviations used are the same as in the performance comparison table. Names of the papers and their corresponding abbreviations are given in the table below.

Average rank	Abbreviation	Paper
4.5	AdaptingBP	*Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure* [36]
4.5	CoopRegion	*A Region Based Stereo Matching Algorithm Using Cooperative Optimization* [37]
5.9	DoubleBP	*Stereo Matching with Color-Weighted Correlation, Hierarchical Belief Propagation, and Occlusion Handling* [41]
7.0	OutlierConf	*Stereo Matching: An Outlier Confidence Approach* [42]
9.2	SubPixDoubleBP	*Spatial-Depth Super Resolution for Range Images* [43]

Table 1. List of the top five algorithms that are studied closer in this survey.

4.3.1 AdaptingBP

The AdaptingBP algorithm is designed in the same way as several other well performing algorithms. Mean-shift image segmentation is used in combination with plane fitting and a subsequent refinement of the fitted planes.

The cost function of this algorithm is a combination of absolute intensity differences and absolute gradient differences:

$$C(x, y, d) = (1 - \omega) \cdot C_{SAD}(x, y, d) + \omega \cdot C_{SGRAD}(x, y, d) \quad (4.2)$$

Cost aggregation is done with a $3x3$ pixel square window and an optimal weighting factor ω is computed by maximizing the number of reliable matches in a right-to-left cross checking.

After this step, a SNR value is calculated through image warping based on reliable disparity estimates that passed the cross checking test. The cost function is then truncated using a threshold set slightly above this value. It is not specified in detail how this is done.

Plane fitting is firstly performed through calculation of slant values for all pixel pairs of an image segment, using a disparity estimate created from WTA optimization of the cost function. Slants are calculated as described in Section 3.4.2. No specific information is given on how the slant values are actually chosen.

The planes are refined in the way that the total cost for all pixels in each segment is calculated for all available plane parameter sets. If any plane parameter set lowers the cost for a segment, the

parameters of the segment are changed to these. Segments sharing the same plane parameters after this step are then grouped together and a second plane fitting is performed for each of the larger, grouped clusters of initial segments.

Finally, optimization of plane parameter assignment is done using loopy belief propagation. The energy function is defined as the sum of total pixel matching cost for assigning planes to segments, in combination with a smoothness cost which is set to depend on segment color similarity and border lengths between the neighboring segments.

4.3.2 CoopRegion

This algorithm also uses mean-shift segmentation and plane fitting. The left image is segmented and the segmentation information is incorporated already in the cost aggregation step as suggested in [40] and described in Section 3.4.4. A large $51x51$ pixel window is used, and pixels that belong to the same segment as the window's center pixel are weighted much higher than pixels of other segments in the aggregation.

An initial plane fitting is then performed using an approach similar to the one described for AdaptingBP. But here, the slant values within segments are instead histogram categorized. Gaussian filtering is then performed on the histogram functions and the slant values corresponding to the peaks of the filtered histograms are selected. No plane parameter refinement is performed prior to optimization. Refinement is instead performed iteratively throughout the optimization step.

The algorithm uses *cooperative optimization* to find optimal plane parameters for every segment, with an energy function that also has an occlusion term in addition to data and smoothness terms. The cooperative optimization works in the way that a single segment is targeted at a time, and optimization of its plane parameters is performed considering a local neighborhood around the segment. Obtained results are used when optimizing the next segment and the optimization is performed iteratively over all segments until it converges.

Data cost is computed by direct pixel-wise matching where the segment plane parameters are used to compute corresponding pixels. Occlusion cost is computed by projecting pixels onto the search image view. When coordinates are projected onto several times, and when coordinates are not projected onto at all, occlusion penalty cost is added. Smoothness cost is added when there is a disparity difference for border pixels of two neighboring segments.

4.3.3 DoubleBP

The DoubleBP algorithm also relies on image segmentation in combination with plane fitting, followed by an energy minimization using belief propagation.

Cost aggregation is performed using adaptive window support weights based on the bilateral filtering technique as discussed in Section 3.1.5. Also, the pixel dissimilarity measure of Birchfield and Tomasi is used. A cost function is computed for both the left and right view to make right-to-left cross checking possible. Lastly, the data cost is truncated to half the average of the cost function to reduce impact of outliers.

After the cost computation, pixels are considered occluded, stable or unstable. Occluded pixels are those who fail a traditional right-to-left cross checking. Stable pixels are those for which there is a

significant difference between the lowest and the second lowest data cost and all other pixels are marked as unstable.

Plane fitting is carried out through the use of RANSAC. If the percentage of stable pixels in an image segment is above a threshold, the disparity values of these are not replaced with the values of the fitted plane. If the percentage is low, on the other hand, all disparity values are determined by the fitted plane. Occluded and unstable pixels are always disregarded and not used in the plane fitting process, and their disparity values are replaced with the values of the plane.

Optimization is carried out through the use of belief propagation, and energy is minimized on a pixel level. The data term of the energy function is set differently depending on the pixel category, to make disparity of stable pixels propagate into regions of occluded and unstable pixels. The smoothness term is linearly proportional to disparity difference of adjacent pixels, and truncated (as in Section 3.2.2).

The plane fitting and optimization steps are performed iteratively to update the left view disparity estimate until convergence.

4.3.4 OutlierConf

This algorithm also has an approach rather similar to other top performers through the use of mean-shift image segmentation, plane fitting and pixel reliability classification.

Two cost functions are computed using the adaptive support weight approach based on bilateral filtering. After that, the cost functions are truncated to half their average. This is similar to what is done in DoubleBP.

A continuous pixel confidence measure is introduced, determining the reliability of pixels. The confidence measure is calculated after an initial disparity estimation using belief propagation. Pixels that are not consistent in a cross checking are labeled as outliers. Moreover, the difference in final belief/cost between pixels that passed the crosschecking and beliefs for pixels that failed it are used to compute a continuous confidence measure. The measure affects a weighting between data cost and smoothness cost in the final optimization step.

The energy function for this algorithm is defined to create left and right disparity estimates simultaneously while considering smoothness within and consistency between the estimates.

In addition to the data cost functions, smoothness cost functions are determined by comparing inlier pixel disparity values against outlier pixel disparity within small local regions, considering color similarity. If an outlier has similar color as inliers and a color different from other outliers within a segment, cost is lowered to encourage a change of its disparity to the value of the inliers. This smoothness cost remains fixed throughout the optimization and its impact is set from the confidence measure.

Another smoothness term of the energy function is based on cross checking. If the disparity values of corresponding left and right disparity map pixels are not consistent, a penalty cost is added. Also, a traditional smoothness penalty is added for the case when two neighboring pixels within a view are assigned different disparities.

Plane fitting is optional and based on RANSAC. It is used to create a modified, regularized data cost where cost for disparities is increased proportionally to their distance from fitted planes within segments.

Worth noting is that this algorithm can perform rather well without the use of image segmentation and image segmentation. However, when image segmentation and plane fitting is enabled, the result is improved.

4.3.5 SubPixDoubleBP

SubPixDoubleBP is a bit different compared to the other algorithms. It is not an ordinary stereo matching algorithm, but instead exploits the bilateral filtering technique to increase the resolution of low resolution disparity estimates, originating from e.g. range cameras.

The disparity refinement is carried out through the use of a color image depicting the same view, but having a higher resolution than the original disparity estimate.

A bilateral filter is applied on a cost function $C(x, y, d)$ created from an upsampled version of the initial disparity estimate according to:

$$C_i(x, y, d) = \min(\alpha, |D_{i-1}(x, y) - d|^2), \qquad i = 1, \dots, n \tag{4.3}$$

D_0 denotes an initial upsampled version of a low resolution disparity estimate. This estimate is updated by iteratively filtering C and performing a simple winner-take-all optimization afterwards on the filtered cost function. Also, sub-pixel computation is performed using quadratic polynomial interpolation for the cost values near the winner of the optimization.

In the presence of two stereo images the algorithm simplifies to iteratively updating a disparity estimate using the same cost function as in DoubleBP.

4.4 Depth Estimation Reference Software (DERS)

The depth estimation reference software used within MPEG is a bit different than the other investigated algorithms, mainly in that it is designed for multi-view and uses three input views instead of two. Also, DERS is used to perform disparity estimation for image *sequences*. How the additional third view is used as well as features related to estimating disparity for sequences will be treated later in Chapter 7.

A large amount of options exists in DERS. However, some options require manual input data (such as manually created edge maps), and these will not be discussed here.

The main outline of disparity estimation in DERS is the following steps:

- Image segmentation (optional)
- Pixel/block matching
- Cost adjustment for temporal enhancement (optional, explained in Chapter 7)
- Disparity computation using graph cuts optimization
- Refinement of created disparity maps by using plane fitting (optional)
- Post processing – $3x3$ median filtering

Disparity computation with subpixel precision is optional.

4.4.1 Image segmentation

For the image segmentation step, mean-shift can be used. The bandwidth parameters are fixed and set within the software source code. Other segmentation approaches also exist, such as *K-means clustering* [17].

4.4.2 Pixel/block matching

There also exist several options for cost computation within DERS. The simplest approach is to compare absolute intensity differences pixel-wise. Other options are $3x3$ square window block matching and the adaptive weight approach based on bilateral filtering as discussed in Section 3.1.5.

4.4.3 Disparity computation using graph cuts

The cost function is fed to a graph cuts optimization framework that uses α-expansion moves. If image segmentation is turned off, the smoothness cost is set based only on disparity difference between adjacent pixels. Otherwise, neighboring pixels located at segment borders are assigned a lower smoothness cost by multiplying the original smoothness parameter with a user specified value of less than one, as described in Section 3.4.3.

4.4.4 Refinement using plane fitting (optional)

This step is invoked when image segmentation is activated. If segmentation is turned on, a plane is fitted for each segment using the method of least squares.

The usage of least squares differs from the approaches found in the other studied algorithms. This could be reasonable, though, as the graph cuts optimization filters out noise rather well prior to the plane fitting.

4.4.5 Post processing – median filtering

As a final post processing step, median filtering is performed, and the window size is hard coded to $3x3$ pixels.

4.4.6 Subpixel matching precision (optional)

To achieve subpixel disparity estimation, the search image is upsampled horizontally to either twice the original width or to four times the original width. This leads to half-pixel and quarter-pixel disparity estimation accuracy, respectively.

When this option is enabled, the number of disparity labels in the cost function is doubled or quadrupled, and the relation between actual horizontal disparity d and a cost function label l is adjusted so that $\Delta d \propto \Delta l/2$ holds for half-pixel accuracy and $\Delta d \propto \Delta l/4$ holds for quarter-pixel accuracy.

4.5 Conclusion on modern stereo correspondence algorithms

All the most well performing algorithms use image segmentation of some kind. The most popular approach is the mean-shift technique. Local color segmentation by the use of bilateral filtering is also rather popular, and sometimes both techniques are used simultaneously.

Apart from the new methods based on slant value calculation, RANSAC seems to be the most common method for performing plane fitting.

Most of the top performing algorithms minimize a global energy function, and fundamentally, the energy functions of use are quite similar. They all contain a data cost, which is weighted against a

smoothness cost. Many algorithms minimize an energy function formulated on pixel level, while some algorithms use energy functions formulated on segment level. Sometimes an occlusion term is also included, which adds a penalty based on left-to-right consistency or considers uniqueness in the mapping of matching pixels between the left and right view.

Up until now, belief propagation and graph cuts are the most popular energy optimization techniques in use, with the first technique being more popular. However, the survey of global optimization techniques [25] states that BP is both slower and less accurate than GC. For message passing, TRW-S is instead promoted.

5 Stereo algorithm implementation

One goal within the project has been to implement a well performing stereo correspondence algorithm. For this, the previously investigated algorithms have served as sources of inspiration.

It has been desirable to implement techniques that are not too computationally demanding and the best performing algorithm on the Middlebury Stereo homepage at the time of this project, AdaptingBP [36], looked rather interesting in this aspect. It uses local approaches to a large extent and finishes with a global optimization formulated on segment level rather than pixel level. Thus, this algorithm has in particular influenced the implementation work.

Some of the algorithmic components that have been implemented are:

- Error evaluation as defined on the Middlebury Stereo homepage [4]
- Image segmentation
- Pixel matching with/without the Birchfield-Tomasi dissimilarity measure
- Cost aggregation using square windows and segment shaped areas
- Winner-take-all optimization
- Left-to-right cross checking
- Truncation of matching cost
- Plane fitting and refinement/clustering
- Global optimization using graph cuts / convergent tree reweighted message passing

Many papers describing stereo algorithms unfortunately leave out important implementation details regarding for example parameters. As a contrast, each of the algorithmic steps will here be treated under its own section together with information on how parameters were set and other things worth noting.

The algorithm has been developed in MATLAB, sometimes invoking external libraries written in C++ and compiled to MEX-format. The reason for choosing MATLAB is that experimenting and testing is simplified as no compilation has to be done. Also, many useful tools and functions exist within the MATLAB environment. The implementation is to be looked upon as a framework for experiments.

5.1 Error evaluation using ground truth data

The error evaluation as it is performed on the Middlebury Stereo homepage [4] was implemented by using the masks made available on the homepage. The masks specify what pixels to include in the error evaluation for three different types of areas:

- Non-occluded areas
- All areas
- Discontinuous areas

5.2 Pixel matching

Pixel matching has been implemented using absolute intensity differences and absolute gradient differences. To perform cross checking at a later stage, four cost functions are created and stored for both the left and right view:

$$C_{AD_L}(x, y, d) = |I_L(x, y) - I_R(x - d, y)| \tag{4.1}$$

$$C_{AD_R}(x,y,d) = |I_L(x+d,y) - I_R(x,y)| \tag{4.2}$$

$$\begin{aligned} C_{GRAD_L}(x,y,d) = {} & |\nabla_x I_L(x,y) - \nabla_x I_R(x-d,y)| \\ & + \left|\nabla_y I_L(x,y) - \nabla_y I_R(x-d,y)\right| \end{aligned} \tag{4.3}$$

$$\begin{aligned} C_{GRAD_R}(x,y,d) = {} & |\nabla_x I_L(x+d,y) - \nabla_x I_R(x,y)| \\ & + \left|\nabla_y I_L(x+d,y) - \nabla_y I_R(x,y)\right| \end{aligned} \tag{4.4}$$

When horizontal pixel coordinates point outside the available image coordinate range, they are set to the smallest or largest available coordinate value.

The BT dissimilarity measure was also implemented, and may optionally replace the absolute values in the above equations when desired.

5.3 Winner-take-all optimization

The implemented WTA optimization has been inspired from the AdaptingBP algorithm.

5.3.1 Cost aggregation

First, cost aggregation is performed, if desired. The aggregation is implemented as an averaging operation using a square window, with manually specified window size. To speed things up, the aggregation is done by performing 2D convolution on each disparity layer of the pixel matching cost function:

$$C_{SAD}(x,y,d) = C_{AD}(x,y,d) * W(x,y) \tag{4.5}$$

$$C_{SGRAD}(x,y,d) = C_{GRAD}(x,y,d) * W(x,y) \tag{4.6}$$

5.3.2 WTA optimization and cross checking

A weighting between SAD and SGRAD cost is calculated from the amount of consistency in a left-to-right and right-to-left cross-checking. The WTA-optimization and subsequent cross-checking is performed for a number of discrete weights $\omega_i \in [0,1]$, which are predetermined:

$$C_L^i(x,y,d) = (1-\omega_i) \cdot C_{SAD_L}(x,y,d) + \omega_i \cdot C_{SGRAD_L}(x,y,d) \tag{4.7}$$

$$C_R^i(x,y,d) = (1-\omega_i) \cdot C_{SAD_R}(x,y,d) + \omega_i \cdot C_{SGRAD_R}(x,y,d) \tag{4.8}$$

$$d_L^i(x,y) = \arg\min_d C_L^i(x,y,d) \tag{4.9}$$

$$d_R^i(x,y) = \arg\min_d C_R^i(x,y,d) \tag{4.10}$$

The disparity estimates $d_L^i(x,y)$ and $d_R^i(x,y)$ created for every weight are warped as described in Section 3.5.1, and the weight ω_i giving the largest number of valid cross matches is considered to be optimal.

Also, a bad pixel map $b(x,y)$ is computed. This map contains information on which pixels that failed the mutual left and right cross checking test. This information is used later to exclude bad disparity estimates from influencing the final result, and also for cost truncation.

5.3.3 Cost truncation

To limit the influence of outliers in later steps of the algorithm, truncation of the cost is performed. The implemented cost truncation makes use of the cross checking information and stores the WTA cost for reliable matches in a list. The list is then histogram analyzed, and the frequency of cost values belonging to the reliable matches is computed.

A cost truncation threshold is determined by summation of the number of occurrences for each of the unique reliable cost values, starting at the lowest cost. When the number of summed cost value occurrences corresponds to a certain percentage of all reliable matches, the summation is terminated. The threshold is set to the cost value just above the highest cost value whose number of occurrences took part in the summation.

5.4 Image segmentation

The mean-shift segmentation has been implemented using a MATLAB MEX wrapper written by Shai Bagon [**44**] using C++ code from the open source project *EDISON* [45].

The mean-shift segmentation algorithm works as described in Section 3.4.1, and has three parameters that control the output; the spatial bandwidth h_r, the color bandwidth h_s and the minimum amount of pixels per segment M.

The wrapper function returns a segmented version of the input image and a label map $l(x, y)$, where each pixel has been assigned an integer label number corresponding to a segment of clustered pixels.

5.5 Plane fitting

Two plane fitting methods have been implemented to return vectors of plane parameters that are assigned to each of the segments that were defined in the image segmentation step. Only pixels that passed the cross-checking and thus not being marked as unreliable by the bad pixel map $b(x, y)$ are used to fit planes for the segments.

5.5.1 Plane fitting based on x and y slant values

This method is proposed in CoopRegion [37] and AdaptingBP [36], although vaguely described.

The calculation of slant values is carried out like described in Section 3.4.2. All the slant values were inserted into lists, one list for horizontal slants and one for vertical slants. The lists were processed to create histograms, giving the number of occurrences as a function of slant values. Several ideas on how to select the best x and y slant values were implemented:

- Creation of histogram functions and selecting the most frequently occurring slant values
- Creation of histogram functions followed by a smoothing of the number of occurrences using a Gaussian filter (i.e. a technique similar to kernel density estimation)
- Sorting of the slant lists, followed by Gaussian filtering of the slant values before creating histogram functions, and then selecting the most frequently occurring slant values
- Quantization of the slant lists before the above approaches using different quantization steps in order to group together similar slant values

5.5.2 Plane fitting based on RANSAC

The *RANSAC Toolbox* written by Marco Zuliani, who is also the author of [35], was used within the project. The toolbox contains a function for estimating planes when given a set of 3D data points.

The most important parameters that have to be specified are the expected variance σ of inlier data points and how consensus sets should be ranked (i.e. whether to use RANSAC or MSAC).

The algorithm determines which data points in the given set that are considered to be inliers, and these are used in a traditional least squares plane fitting.

5.6 Plane refinement

The plane refinement technique described in AdaptingBP was implemented. A faster version that compared the data cost when only trying *neighboring* and *second neighboring* segments' planes was also implemented. The idea for doing this is that it hopefully would not only be faster, but also more robust. When deciding which segments are neighbors, four-connectivity of segment pixels is regarded (and eight-connectivity is optional).

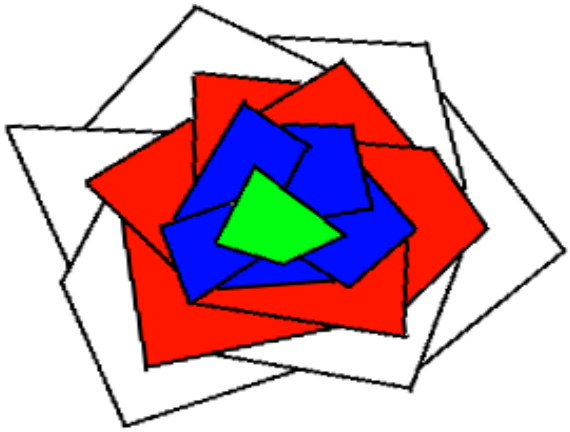


Figure 23. Neighboring segments: The green segment in the middle is neighbor with the blue segments, and 'second neighbor' with the red segments.

The refinement works by computing the cost value for assigning plane parameters to a segment, and then choosing the plane parameter vector that give the minimum cost (energy). This can be seen as an energy minimization of the following function, performed individually for each segment:

$$E_s(l) = \sum_{(x,y)\in s} C(x, y, d(x, y, l)), \qquad s \in S \tag{4.11}$$

$$d(x, y, l) = \max\left(\min\left((x\ y\ 1) \cdot \boldsymbol{\theta}_l, d_{max}\right), d_{min}\right) \tag{4.12}$$

Here, S is the set of all segments and $\boldsymbol{\theta}_l$ is the plane parameter vector belonging to a segment l, which was determined in the plane fitting step. The plane fitted disparity is limited to the disparity search range specified by d_{min} and d_{max}.

Trying all available plane parameters for every segment, i.e. allowing $l \in S$ when minimizing the above energy function for each segment, is similar to what is done in AdaptingBP. The second neighbor approach is achieved by reducing the allowed values for l to $l \in S_{N2}$, where S_{N2} denotes the set of all segments that are connected to each other as illustrated in Figure 23.

The result after the refining has been carried out is a clustered set of plane parameters. A second plane fitting is performed over the clusters that share the same plane parameters, again using the original noisy WTA disparity estimate. The idea is that the clustered segments lay on the same disparity level, even if they are of dissimilar color. Thus, the plane fitting will be performed over larger regions, in theory generating more reliable plane parameters.

5.7 Global optimization routines

5.7.1 Graph cuts

To implement graph cuts optimization, a MATLAB MEX wrapper written by Shai Bagon [44], utilizing C++ code of Olga Veksler et al [46][47][48] was used. The used graph cuts optimization function can handle a number of problem formulations:

- The minimization of a standard energy function $E(d) = E_{data}(d) + \lambda E_{smooth}(d)$ as defined in Section 3.2.2, but with a smoothness term that does only depend on disparity differences between neighboring pixels (and not on intensity differences).
- The minimization of an energy function $E(d)$ with a spatially varying smoothness parameter $\lambda = \lambda(x, y)$, which can be used to minimize energy functions where intensity differences are taken into consideration. The spatial variation of λ can e.g. be set using an edge map.
- The minimization of an arbitrarily shaped graph with unique pair-wise energy terms, i.e. a totally user defined graph and not a grid graph, which is assumed in the previous two problem formulations. This approach can be useful when minimizing energy functions formulated on a segment level instead of on a pixel level.

Both the α-expansion and α, β-swap moves are supported. However, only the expansion move algorithm is of interest (for the reasons discussed in Section 3.3.1).

5.7.2 Convergent tree reweighted message passing

The minimization routine using convergent tree reweighted message passing (TRW-S) [31] was implemented using a MATLAB MEX wrapper written by Oliver Woodford [49][50], in turn making use of a C++ routine written by Vladimir Kolmogorov.

The wrapper function takes as argument an entirely user defined and arbitrarily shaped graph, where smoothness terms (edge costs) have to be specified in one of the following ways:

- A matrix of edge costs for all label combinations, which is used to set costs for all edges in the graph
- Several matrices of edge costs for label combinations, with the possibility to set unique costs between every pair of nodes. For every node pair, an index referring to a matrix of smoothness terms is specified.

An interesting feature with this algorithm is that each node of the graph may have an individual number of labels (as opposed to the graph cuts algorithm).

5.8 Optimal plane reassignment

Optimal plane reassignment as described in AdaptingBP was implemented using TRW-S. A graph linking neighboring segments together was set up, and the smoothness cost was set to depend on

the plane parameters, the color similarity of and the common border length of neighboring segments.

This approach can be expressed as minimizing an energy function using global optimization:

$$E(\mathcal{L}) = E_{data}(\mathcal{L}) + E_{smooth}(\mathcal{L}) \tag{4.13}$$

In this case, the labeling problem consists of assigning optimal plane parameter values $\boldsymbol{\theta}_{l_i}$ to every segment s_i. The assignments are defined by the labeling $\mathcal{L}(s_i) = l_i$.

The data term E_{data} is calculated locally for every segment in the same way as for the plane refinement step (4.11). Every local contribution is then summed over all segments to form a global data cost.

The smoothness term E_{smooth} is the sum of all smoothness costs between neighboring segments. As in AdaptingBP, the smoothness cost between two neighboring segments is calculated as:

$$\begin{aligned} E_{smooth}(s_i, s_j) = \lambda_{smooth} \cdot \min(|l_i - l_j|, 1) \cdot borderlength(s_i, s_j) \\ \cdot similarity(s_i, s_j) \end{aligned} \tag{4.14}$$

The border length calculation is not strictly defined in the AdaptingBP paper or in the paper which originally proposed the smoothness term of this type [51]. As the number of common border pixels may be different depending on which of the neighboring segments that is counted from (for example, think of a line-shaped segment surrounded by another segment's pixels), the mean value of the two neighboring segments' number of four connected border pixels was used.

The color similarity of two segments was implemented as proposed in [51]:

$$similarity(s_i, s_j) = \left(1 - \frac{\min(\|cmean(s_i) - cmean(s_j)\|, 255)}{255}\right) \cdot 0.5 + 0.5 \tag{4.15}$$

In (4.15), $cmean(s_i)$ is the vector of mean R, G and B color values of pixels within a segment s_i.

In addition to this approach, which is similar to the one of AdaptingBP, the ability of TRW-S to handle an individual amount of labels for every graph node was used to implement an approach based on second neighbors as discussed earlier. In this implementation, the possible plane parameters that can be assigned to a segment are limited to its own parameters and the parameters of its neighbors and second neighbors.

5.9 Disparity optimization using graph cuts

Graph cuts optimization was implemented to optionally make use of image segmentation data. The energy function is on standard form, specified on a pixel level, and uses the segmentation data to set a spatially varying smoothness parameter:

$$E(d) = E_{data}(d) + E_{smooth}(d) \tag{4.16}$$

The data term $E_{data}(d)$ is simply the sum of local costs for the disparity map $d = d(x, y)$ as specified by the computed cost function $C(x, y, d)$, and the smoothness energy term using segmentation information can be written as:

$$E_{smooth}(d) = \sum_{(i,j)\in N} V_{ij}(d_i, d_j) \tag{4.17}$$

$$V_{ij}(d_i, d_j) = \begin{cases} \lambda \cdot \rho_d(d_i - d_j) & when\ s_i = s_j \\ \gamma \cdot \lambda \cdot \rho_d(d_i - d_j) & when\ s_i \neq s_j \end{cases} \tag{4.18}$$

$$\rho_d(d) = \min(|d|, \alpha) \tag{4.19}$$

N is the set of all four connected neighboring pixels, with each pair of neighbors included once. For two neighboring pixels i and j located within the same image segment $s_i = s_j$, the smoothness cost is set to λ. For neighboring pixels belonging to different image segments, the parameter is lowered by multiplication with a constant $\gamma \in [0.5,1]$ with the purpose to encourage disparity jumps at segment borders. This smoothness energy term was implemented by performing edge detection on the label map $l(x, y)$ outputted from the image segmentation routine and defining a spatially varying $\lambda(x, y)$ to return $\gamma \cdot \lambda$ for segment border pixels and λ for all other pixels.

6 Middlebury images – experimental results

This part of the report contains experimental results for the implemented stereo algorithm. To reduce the extent of this chapter, mainly experimental results for the *Cones* stereo image pair are presented. Additional results for the other available stereo pairs of the Middlebury Stereo homepage [4] are given in 0.

(a) (b)

Figure 24. The Cones stereo image pair retrieved from the Middlebury Stereo homepage: a) The left (reference) image b) Ground truth disparity map, where disparity has been scaled with a factor of 4. The black regions are regions with unknown disparity due to occlusion.

Throughout the experiments, the same error measures as those on the Middlebury Stereo homepage will be computed and used. These are the percentages of bad pixels in *non-occluded areas*, *all areas* and *discontinuous areas*. For clarity, these errors will be abbreviated $\mathcal{N}$, $\mathcal{A}$ and $\mathcal{D}$. For a definition on how the number of bad pixels is computed, see Section 4.2.

6.1 Local disparity estimation

6.1.1 Reproduction of Middlebury results

This experiment aims to reproduce results created from the software framework released by Daniel Scharstein and Richard Szeliski on the Middlebury Stereo homepage [4]. The chosen experiment is WTA minimization of a $9x9$ SAD cost function. If the results agree, this means that the implemented error evaluation as well as the WTA routine works as expected.

	Tsukuba			Venus		
	$\mathcal{N}$	$\mathcal{A}$	$\mathcal{D}$	$\mathcal{N}$	$\mathcal{A}$	$\mathcal{D}$
Middlebury testbed	8.64%	10.67%	25.66%	13.60%	15.06%	33.80%
This implementation	8.56%	10.54%	26.94%	10.97%	12.32%	38.08%

Table 2. Percentage of bad pixels for Tsukuba and Venus when using WTA minimization of a 9x9 SAD cost function.

The results seem to agree fairly well. The error evaluation routine has also been tested by downloading other algorithms' results from the Middlebury Stereo homepage, calculating their errors and comparing the results with the errors given on the homepage. In this case, the computed errors were identical.

6.1.2 The implemented winner-take-all routine

To examine the effect of different window sizes, compare AD with BT and to see the effect of using a gradient based cost function, exhaustive testing was performed. The parameter ranges were:

- Window sizes: $3x3$, $5x5$, $7x7$, $9x9$, $11x11$, $13x13$, $15x15$, $19x19$
- Dissimilarity measure of Birchfield-Tomasi: On/Off
- Gradient based matching cost: On/Off
- Gradient weights: $0 - 1.0$ in steps of 0.1

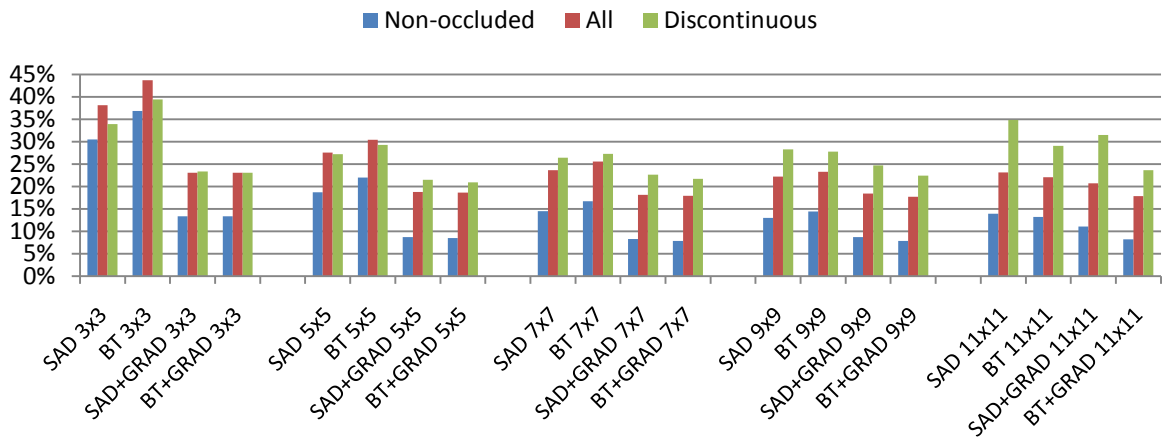


Figure 25. Percentage of badly estimated disparities in different regions of the Cones image pair when performing WTA-optimization using different window sizes.

6.1.3 Conclusion

First of all, it is clear that the gradient based cost function achieves better results than when only using intensity. The error typically drops several percent points.

It is also clear from the results that the BT dissimilarity measure helps in most cases, when it is used in combination with a gradient based cost function. The error typically drops $1 - 3$ percent points when BT is enabled. Note that when the cost function is based on intensity differences only, the results are ambiguous and BT does not seem to add much.

For all cases when the gradient weighted cost function was used, the weight values typically varied between 0.7-0.9, with very few exceptions. This means that it is probably only necessary to try a few weights, e.g. 0.5, 0.7 and 0.9.

Also, note how the error for discontinuous regions starts to rise as the aggregation window size increases. This is expected, as large windows will contain both foreground and background pixels when aggregating at disparity discontinuities.

6.2 Cost truncation

In this experiment, the implemented cost truncation is tested. Based on the previous results of the *local disparity estimation* experiment, BT was enabled and a gradient based cost function was used.

The parameters varied as follows:

- Window sizes: $3x3$, $9x9$
- Gradient weights: 0.7, 0.8, 0.9
- Automatic truncation threshold: On/Off
- Automatic truncation threshold percentage, when on: 70, 80, 90

6.2.1 Manual cost truncation

Cost truncation thresholds were set for two window sizes, $3x3$ and $9x9$ pixels. For these window sizes, the best available thresholds were found manually by looking at the number of bad pixels after WTA disparity estimation. When the number of bad pixels starts to increase, this is a sign indicating that the threshold is set too low.

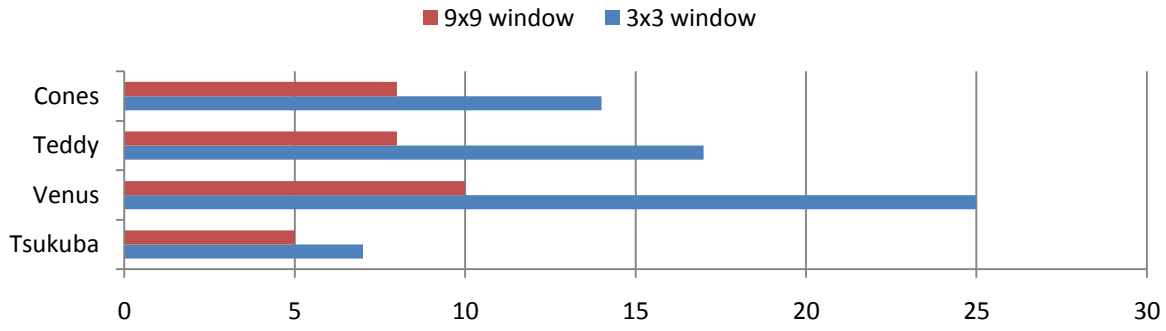


Figure 26. Manually determined cost truncation threshold values. The values were set so that a further decrease led to an increasing percentage of bad pixels.

The errors after WTA optimization for truncated costs were also compared against results for non-truncated costs and for results coming from automatically truncated costs, see next section.

6.2.2 Automatic cost truncation

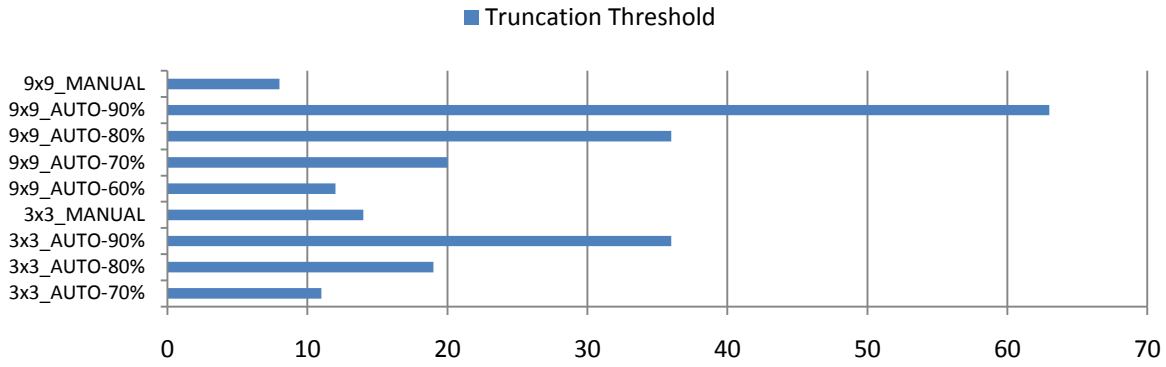


Figure 27. Results from automatically determining cost truncation values using different parameter values.

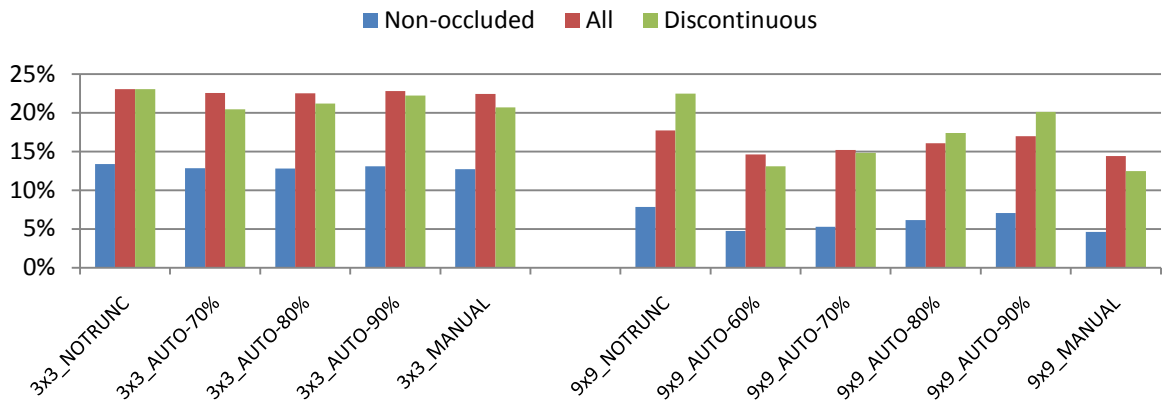


Figure 28. Percentage of bad pixels for a WTA estimated disparity when automatically determining cost truncation values.

6.2.3 Conclusion

The automatic cost truncation helps to reduce errors in areas where there are disparity discontinuities. Also, it bounds the cost function, which can be useful in e.g. global optimization and clustering.

If compared to the manually specified thresholds, good parameter values seem to be $80 - 90\%$ for $3x3$ window size, and $60 - 70\%$ for $9x9$ window size.

The errors drop when the cost function is truncated before aggregation and winner-take-all minimization. For a small $3x3$ window, the number of bad pixels typically drops $0.5 - 1.0$ percent points. For a larger $9x9$ window, the drop is $0.5 - 3.5$ percent points.

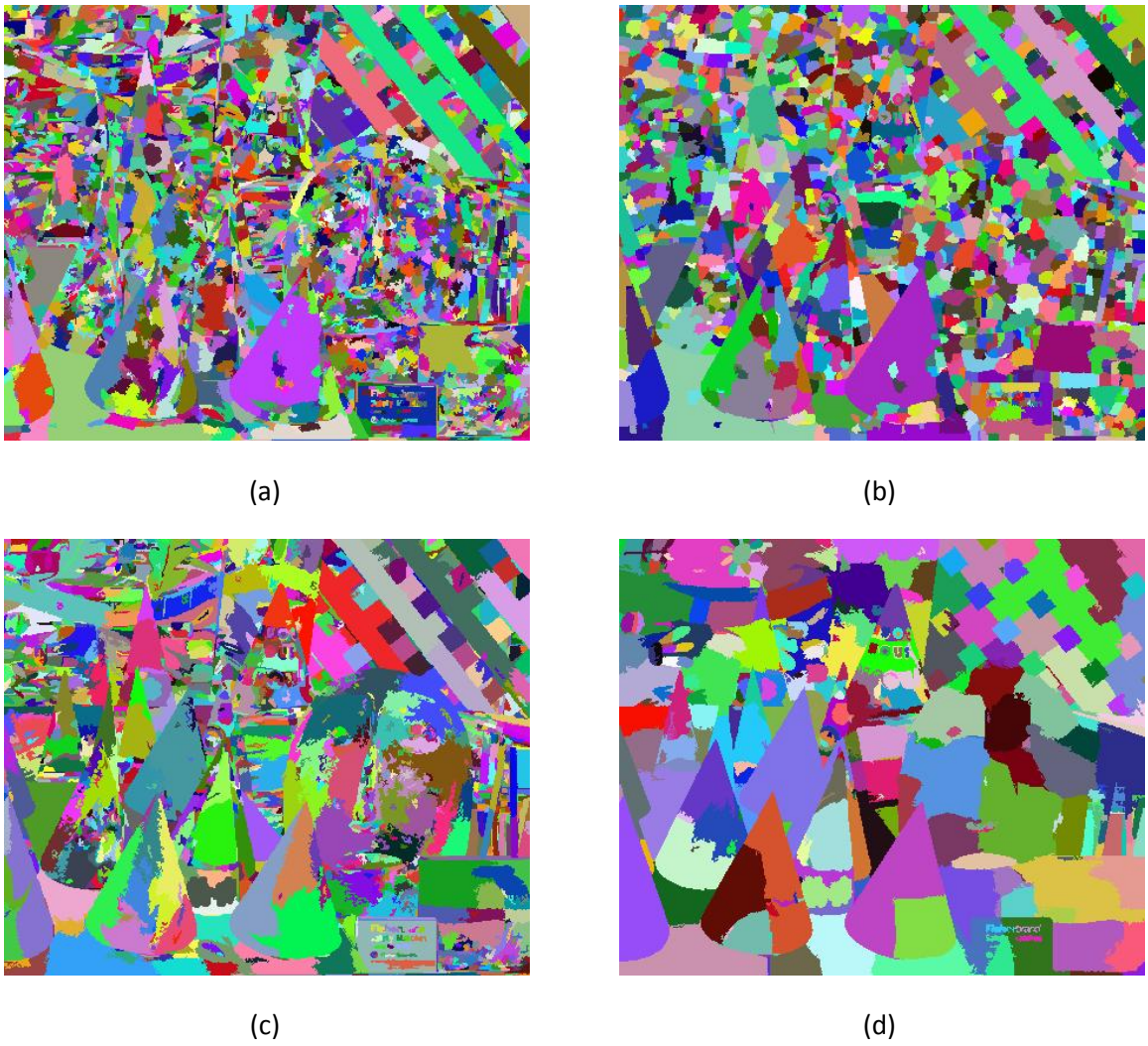
The drop at discontinuous regions is explained by that impact of cost that is aggregated from a different disparity than the aggregation window's center pixel is limited.

6.3 Image segmentation

6.3.1 Image segmentation parameter testing

The aim of this experiment is to get an understanding of how the image segmentation parameters affect the result. The spatial bandwidth h_s, the color bandwidth h_r and the minimum pixels per segment M were varied manually and the result was noted.

6.3.2 Results and conclusion



(a) (b)

(c) (d)

Figure 29. Image segmentation results for Cones attained from different parameters: a) $(h_s, h_r, M) = (5,5,11)$ *resulting in 2780 segments b)* $(h_s, h_r, M) = (5,18,11)$ *resulting in 1531 segments c)* $(h_s, h_r, M) = (18,5,11)$ *resulting in 1514 segments d)* $(h_s, h_r, M) = (18,18,11)$ *resulting in 348 segments*

The objective of the image segmentation is to delineate image regions with similar color. It is desirable to keep the number of segments as low as possible, while at the same time preserving correct object boundaries.

Reasonable parameter values were found to be $h_s,\ h_r \in [5,15]$ and $M \in [30,50]$.

6.4 Plane fitting

Based on the results of AdaptingBP [36], a small window size of $3x3$ pixels was used in the WTA optimization before plane fitting in these experiments. This small window size generates quite noisy disparity estimates, and the goal of performing plane fitting is to reduce the impact of the noise by robustly fitting planes in image segments using only disparity values that were validated in the cross checking.

6.4.1 Plane fitting using RANSAC/MSAC

In this experiment, the parameters varied as follows:

- RANSAC algorithm type: RANSAC, MSAC
- Error standard deviation: $0.1, 0.2, 0.3, 0.4, 0.5, 0.6$
- Probability that a data point is an inlier given that its distance to the fitted plane is below the threshold for the selected noise level, $P\left(\boldsymbol{d}_i \; inlier \middle| e\left(\boldsymbol{d}_i, \boldsymbol{\mathcal{M}}(\boldsymbol{\theta})\right) < \delta\right)$: 0.99
- Probability of only selecting bad minimal sample sets, ϵ: 10^{-6}

The relation between specified standard deviation and the maximum allowed distance from the plane surface for an inlier data point is given in the table below for the case when $P\left(\boldsymbol{d}_i \; inlier \middle| e\left(\boldsymbol{d}_i, \boldsymbol{\mathcal{M}}(\boldsymbol{\theta})\right) < \delta\right) = 0.99$.

σ	0.1	0.2	0.3	0.4	0.5	0.6
δ	0.3368	0.6737	1.0105	1.3474	1.6842	2.0211

Table 3. Relation between specified noise level and distance threshold. Data points that are at a distance of less than δ from a fitted plane are marked as inliers.

Other available parameters in the RANSAC Toolbox were set to their default values. Due to the random sample selection nature of RANSAC, five tests were conducted for each parameter setup, and the average errors were saved (as the error percentage fluctuates from one run to another, there would otherwise be a risk of drawing the wrong conclusions and not selecting the most appropriate parameters).

Results for the Cones image pair

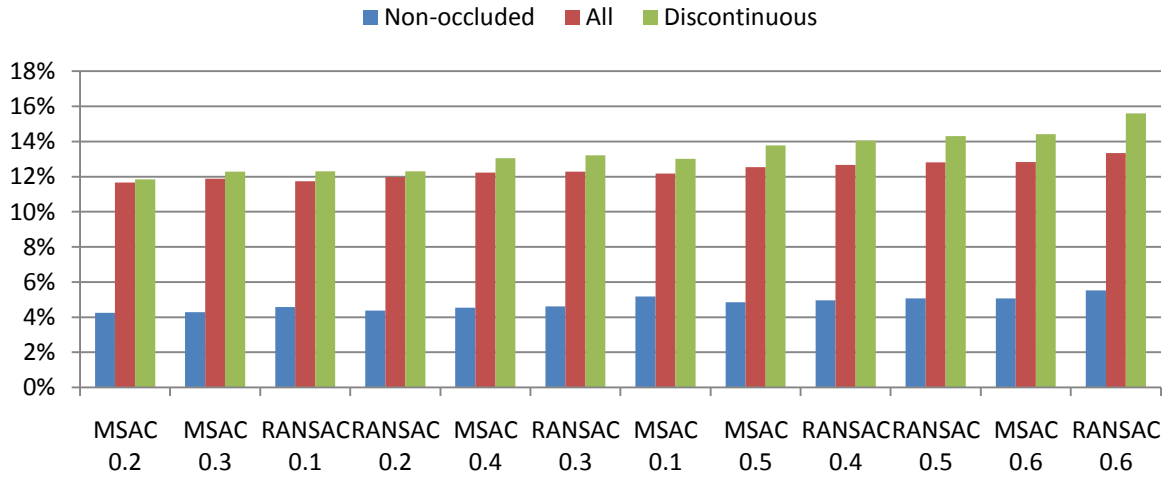


Figure 30. Plane fitting results for Cones, sorted by percentage of 'all' bad pixels.

Other window sizes

Experiments were also run with a window size of $7x7$ pixels. Generally, the error goes up when using window sizes larger than $3x3$ pixels. This is reasonably explained by the smoothing of the cost function that take place, which gives less accurate results at discontinuous regions.

6.4.2 Plane fitting using slant values

Experimental setup

Initially, some experiments were performed in order to select which approaches to proceed with. Two techniques were tried: Creating a table of quantized slant values and the corresponding number of occurrences, and creating a histogram with a fixed number of bins.

The vector containing the number of frequencies was filtered with Gaussian kernels to smoothen it. In the case where a frequency table was used, its length differed from time to time, and when a histogram was used, the length was manually specified. The length of the Gaussian filter was adaptive and specified in fractions of the length of the frequency table. In the case where a histogram was used, this is similar to kernel density estimation (as discussed in Section 3.4.1).

Tests were run over a large parameter space. For the case of a frequency table, the parameters were:

- Slant value quantization step: $0.0001, 0.001, 0.002, 0.003, 0.005, 0.01, 0.02$
- Gaussian filtering standard deviation: $0.5 - 19$ in steps of 1
- Gaussian filter lengths: $0.05 - 0.9$ (multiplied by the number of unique quantized slope values in the frequency table)

For the histogram technique, the parameters were:

- Number of bins: 500, 1000, 10000
- Gaussian filtering standard deviation: $0.5 - 9$
- Gaussian filter lengths: $0.01 - 0.9$ (multiplied by the number of histogram bins)

Results

The best result for each image pair from using individual parameters is presented in the diagram. RANSAC/MSAC results are also in the table for comparison.

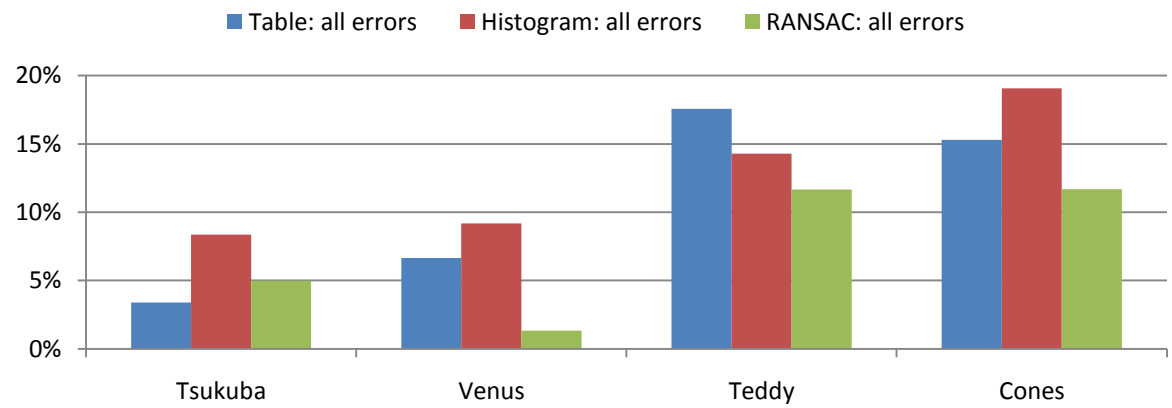


Figure 31. Comparison of plane fitting results when using approaches based on slant values against RANSAC/MSAC. The table shows the number of bad pixels for the category 'all pixels'.

The outcome of the plane fitting based on slant values was not successful. Except for one case, RANSAC/MSAC performed a lot better. Also, it should be noted that there was a high dependency on selecting the 'right' parameters for each image pair when using the histogram approaches compared to when using RANSAC/MSAC.

6.4.3 Comparison with least squares and result from cross checking

To determine the effect of cross checking, plane fitting was performed with and without excluding bad pixels by using cross checking information. For the sake of comparison, pure least squares plane fitting was also performed without removing outliers by using RANSAC. RANSAC was run with *MSAC* and $\sigma = 0.3$ and the percentage of *all* bad pixels was compared.

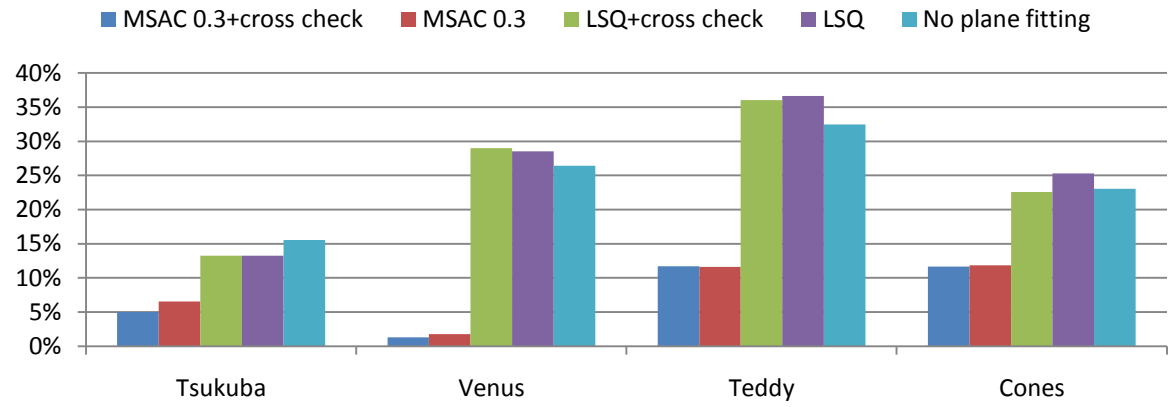


Figure 32. Comparison of plane fitting results with different options enabled/disabled. LSQ denotes pure least squares plane fitting.

6.4.4 Conclusion

Plane fitting definitely helps to increase the quality of the result. Consensus set ranking based on MSAC is preferred instead of pure RANSAC. A good parameter choice for the Middlebury image stereo pairs seems to be MSAC with σ set to 0.3.

Least squares plane fitting without any prior outlier removal does not improve the result. In fact, the result gets worse in most cases.

6.5 Plane refinement

Clustering of segment planes was implemented as described in Section 5.6, and tested.

Initial experiments were run, only testing neighboring segment plane parameters. This approach was faster, but had to be disregarded as the number of unique clusters stayed high.

Experiments were instead run trying all available plane parameters, and parameters of neighbors and neighbors' neighbors. In the clustering, an automatically truncated cost function was used and the truncation percentage parameter was varied.

After the clustering had been performed, a second plane fitting was performed over the grouped regions that had been assigned the same plane parameters, still using the initial disparity estimate created from the WTA optimization together with the bad pixel map outputted by the cross checking.

6.5.1 Results for clustering

Results from the clustering step when trying all available plane parameters as well as only plane parameters from local neighborhoods around segments are presented in Figure 33.

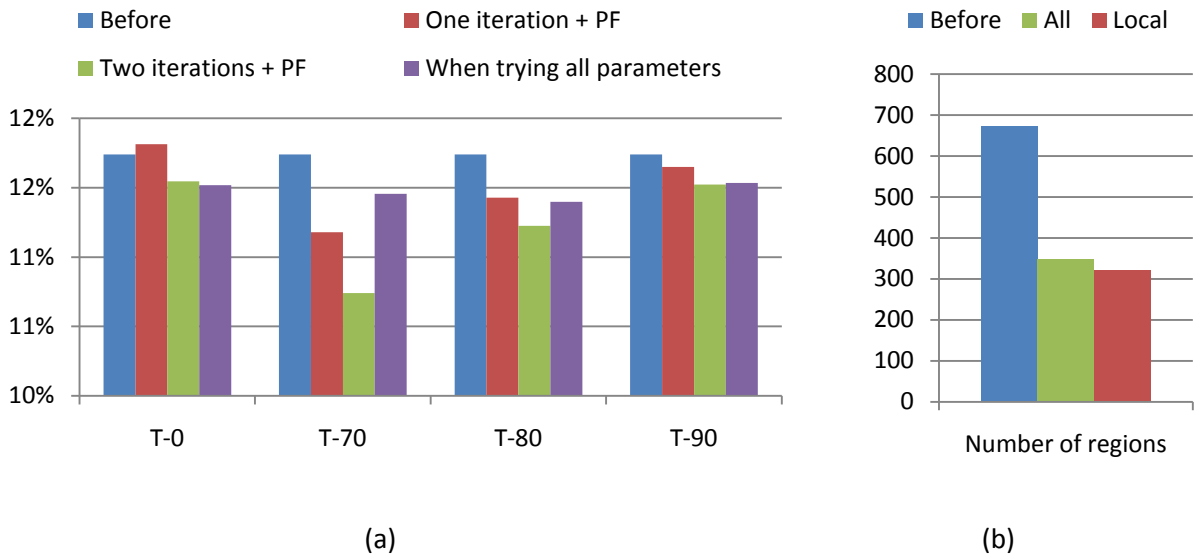


Figure 33. a) Percentage of 'all' bad pixels: comparison of clustering results with different cost truncation thresholds and one or two clustering iterations, the result from trying all parameters is for comparison b) The number of regions in the image before and after clustering, also comparing the approach (Local) to when trying all plane parameters (All).

6.5.2 Conclusion

Plane clustering helps improving the quality further, and at the same time, the number of segments is heavily reduced. The number of segmentation regions in the reference image typically drops with 50-80% depending on which input image pair that is used.

In the plane clustering, it was found that there is a clear need for cost truncation. This is likely explained by that outliers have a much higher cost than inliers, and that this can lead to the preferring of a rather bad plane with fewer outliers than some plane that would otherwise be better.

The implemented new clustering idea only looking at plane parameters of neighbors and 'second neighbors' gave better overall results than when trying all plane parameters. It also gave fewer regions after the clustering. The number of regions decreases the most in the first and second iteration, and two iterations was found to be sufficient.

6.6 Plane reassignment using TRW-S optimization

6.6.1 Experimental setup

While running some initial experiments for the final step of plane parameter reassignment, it was found that the results were more reliable when only reassigning using parameters originating from local neighborhoods around segments. The smoothness parameter λ was set empirically to 10 for *Venus* and to 1 for the other pairs.

6.6.2 Results

The results from plane reassignment using TRW-S is given in the table below. The error before reassignment is given inside the parentheses.

Image	λ	$\mathcal{N}$		$\mathcal{A}$		$\mathcal{D}$		$N_{segments}$	
Tsukuba	1	5.12%	(5.55%)	5.82%	(6.28%)	7.25%	(13.7%)	34	(117)
Venus	10	0.34%	(0.91%)	0.51%	(1.43%)	7.83%	(9.09%)	21	(93)
Teddy	1	4.96%	(5.41%)	8.57%	(9.00%)	13.4%	(13.8%)	41	(253)
Cones	1	3.51%	(3.61%)	9.91%	(10.8%)	10.4%	(10.4%)	39	(321)

Table 4. The percentage of bad pixels for the Middlebury stereo images after TRW-S plane reassignment has been performed. Values in parentheses are the errors before TRW-S reassignment, from the clustering step.

6.6.3 Conclusion

The plane reassignment step further clusters the segments and the number of segments that are assigned a unique plane parameter vector drops radically. From the initial image segmentation step the number of segments has now dropped with more than 90%.

The results for *Teddy* and *Cones* are now comparable with top ten results published on the Middlebury evaluation homepage and *Venus* is ranked within top 20. However, there seems to be a problem with the *Tsukuba* image pair. This problem is also apparent in the next section, where results from the graph cuts implementation are published.

6.7 Disparity estimation by graph cuts energy minimization

6.7.1 Experimental setup

The graph cuts implementation was tested with parameters varying as follows:

- Birchfield-Tomasi: On/Off
- Manual cost truncation limit: 30, Inf
- Gradient weight for cost function: 0, 0.75
- Smoothness coefficient (λ): 3, 7, 11, 16, 21
- Smoothness truncation parameter (α in (2.27)): 1, 3, 5
- Segment border smoothness lambda multiplier: 0.5, 0.75
- MSAC plane fitting (with $\sigma = 0.3$): On/Off

6.7.2 Results and conclusion for GC energy minimization without planefitting

Image	BT	C_{max}	ω	λ	α	γ	$\mathcal{N}$	$\mathcal{A}$	$\mathcal{D}$
Tsukuba	On	30	0	7	3	0,5	1.52%	3.48%	7.25%
Venus	On	30	0	16	3	0,75	0.60%	1.60%	7.83%
Teddy	On	Inf	0,75	7	5	0,5	6.77%	15.2%	19.3%
Cones	Off	30	0,75	7	3	0,5	3.54%	11.1%	10.4%

Table 5. Winning parameter values and bad pixels for the energy minimization using graph cuts optimization.

The results are not far from the top performing algorithms on the Middlebury Stereo homepage. For Tsukuba, Venus and Teddy, BT was clearly overrepresented among the best results. For Cones,

however, BT did not lead to any improvement. Gradient weighted cost was represented among all the top 30 results for Cones and Teddy, but hardly represented at all for the Tsukuba and Venus top results. Cost truncation was overrepresented for all pairs, except for Teddy.

6.7.3 Results for energy GC minimization with MSAC planefitting

Image	BT	C_{max}	ω	λ	α	γ	$\mathcal{N}$	$\mathcal{A}$	$\mathcal{D}$
Tsukuba	On	Inf	0,75	7	3	0,75	2.87%	3.53%	11.3%
Venus	On	30	0	11	3	0,75	0.45%	0.86%	5.43%
Teddy	On	30	0,75	3	5	0,5	4.33%	10.7%	11.9%
Cones	On	30	0,75	3	3	0,75	3.75%	11.8%	10.6%

Table 6.Winning parameter values and bad pixels for the energy minimization using graph cuts optimization followed by plane fitting.

Plane fitting improves the result significantly for *Venus* and *Teddy*, but makes the result significantly worse for *Tsukuba*. For *Cones*, the result is basically equal to when not using plane fitting.

6.7.4 Conclusion on global energy minimization using graph cuts optimization

The results are quite inconclusive when it comes to whether the mentioned features such as BT and gradient weighted cost help or not when using GC energy minimization for the general case.

For all image pairs except for *Venus*, the implemented graph cuts algorithm can match the top ten algorithm results of the Middlebury evaluation homepage.

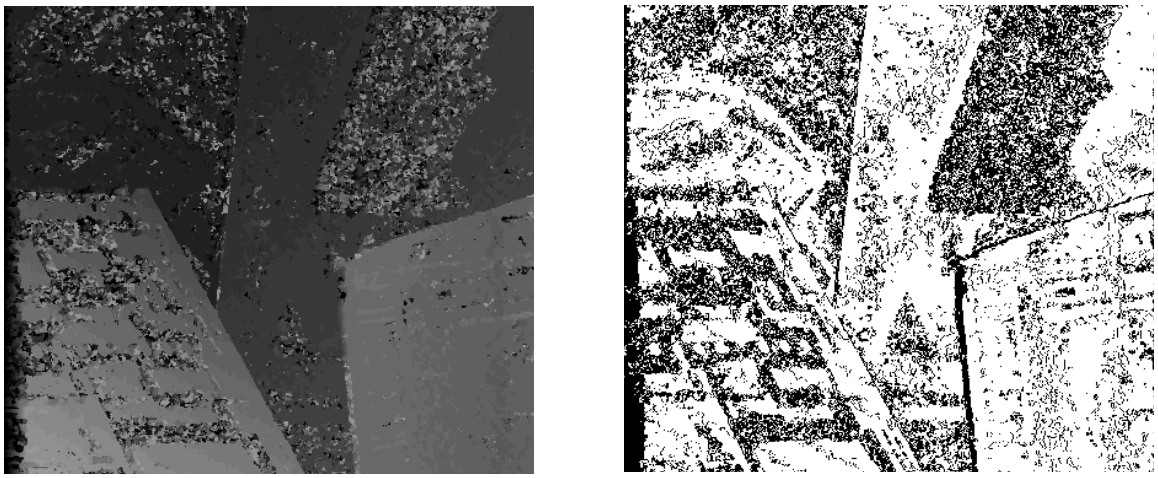
6.8 Intermediate results from the implemented algorithms

In this section, it will be shown what the different steps in the implemented stereo algorithms are actually doing and how the intermediate disparity estimates look.

6.8.1 Implementation using WTA-optimization, plane fitting and TRW-S reassignment

The stereo image pair *Venus* is used for illustration purpose, as the transformation from initial WTA estimate to the final result is quite large in this case.

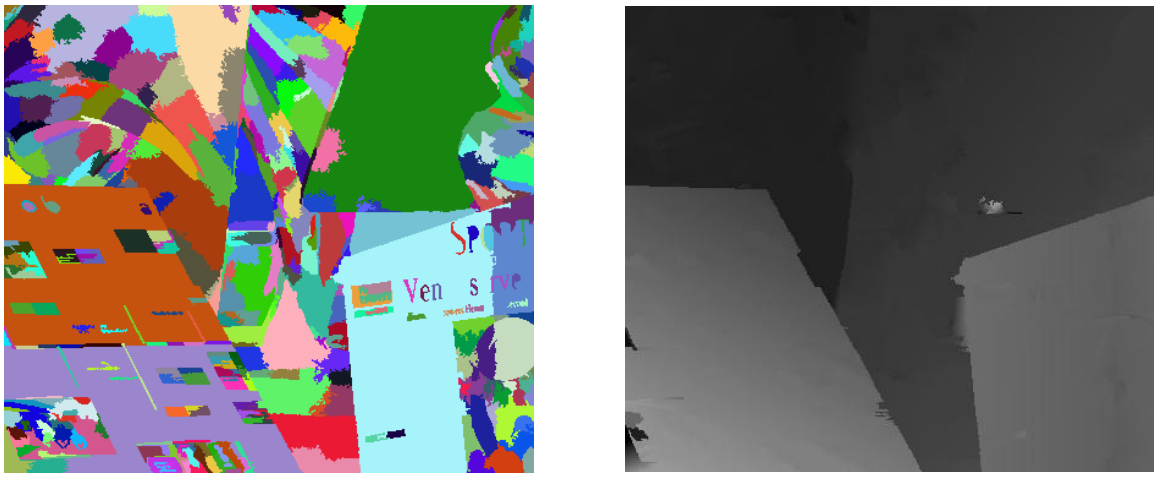
Winner-Take-All optimization and right-to-left cross checking



(a) (b)

Figure 34. Result after WTA-optimization and right-to-left cross checking: a) Disparity estimate b) Estimated disparities found to be unreliable by the cross checking are marked as black.

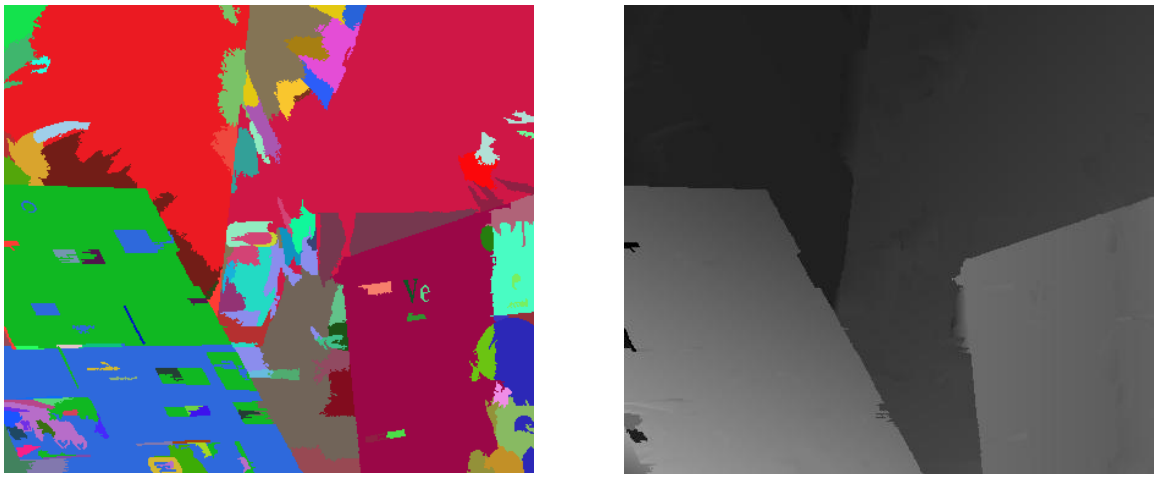
Image segmentation and plane fitting



(c) (d)

Figure 35. Results from the image segmentation and plane fitting: a) The delineated segments b) Disparity estimate after MSAC plane fitting

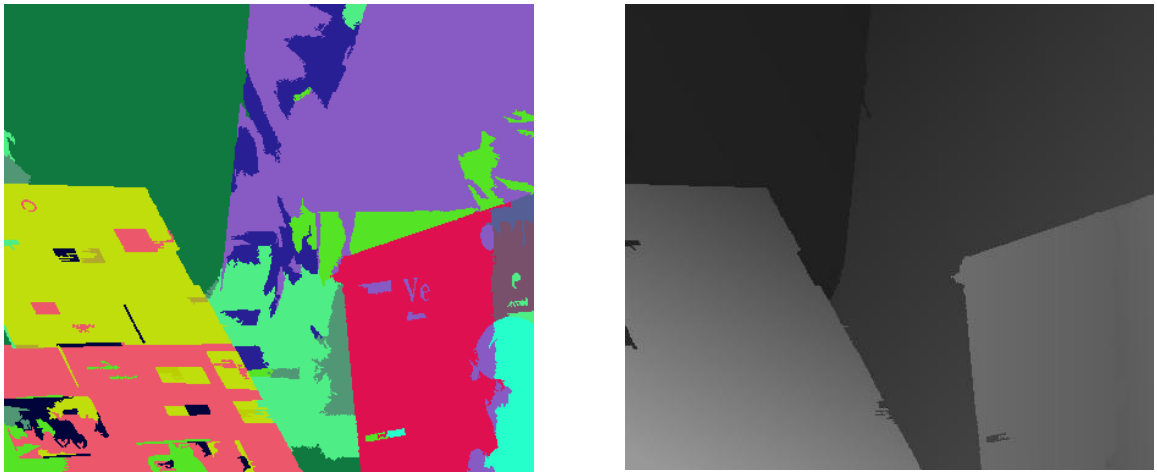
Plane refinement/segment clustering



(a) (b)

Figure 36. Results from the clustering step: a) The regions formed by the clustering b) Disparity estimate after MSAC plane fitting over the new regions shown in (a)

Plane parameter reassignment using TRW-S



(a) (b)

Figure 37. Results from the global optimization step using TRW-S: a) Regions with unique plane parameters b) Disparity estimate with the plane parameters as assigned to the clustered segments after TRW-S

Quantitative results for the different steps

Algorithmic step	$\mathcal{N}$	$\mathcal{A}$	$\mathcal{D}$	$N_{regions}$
WTA	25.2%	26.4%	30.2%	-
First plane fitting	1.15%	1.65%	8.88%	368
Clustering + second plane fitting	0.82%	1.35%	8.64%	96
TRW-S reassignment	0.39%	0.53%	4.47%	15

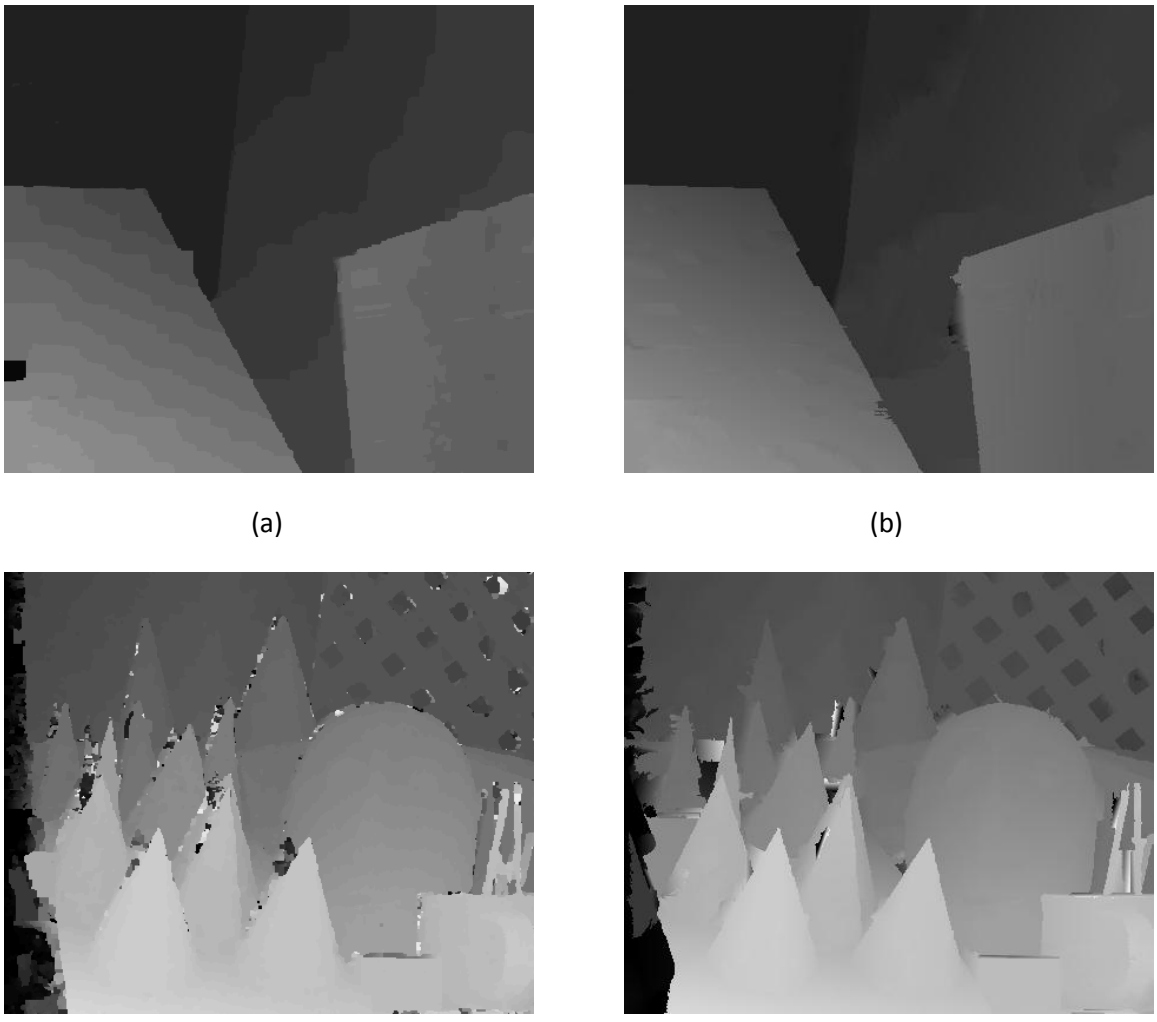
Table 7. Quantitative results (percentage of bad pixels) for the Venus image pair after each algorithmic step.

6.8.2 Implementation using graph cuts optimization

For the graph cuts implementation, there are no intermediate results to show in the same way as for the other algorithm implementation. There is firstly the disparity estimate produced after the optimization, and optionally a refined disparity estimate created from performing a subsequent plane fitting.

Disparity estimates after GC optimization and optional plane fitting

The *Venus* and *Cones* image sets are chosen to exemplify the results.



(a) (b)

(c) (d)

Figure 38. Results from the GC optimization: a) Venus c) Cones. Results after also performing a plane fitting: b) Venus d) Cones.

Quantitative results

Algorithmic step	$\mathcal{N}$	$\mathcal{A}$	$\mathcal{D}$
Venus, without/with plane fitting	0,60% / 0,45%	1,60% / 0,86%	7,83% / 5,43%
Cones, without/with plane fitting	3,54% / 3,75%	11,1% / 11,8%	10,4% / 10,6%

Table 8. Quantitative results (percentage of bad pixels) for the Venus image pair after each algorithmic step.

7 Real world images and sequences – implementation and results

This section will treat real world stereo images and present related findings. Implementation descriptions will be mixed with results as the nature of this part of the project was more trial and error than in the earlier work. First, a short introduction to disparity estimation for sequences will be given. Then, ideas that have been tried will be described along with results.

7.1 Disparity estimation for multi-view sequences using DERS

Here, features related to disparity estimation for sequences that can be found in *Depth Estimation Reference Software* (DERS) will be presented. Also, some information on how to use three camera views for disparity estimation will be given.

7.1.1 Temporal stability

When performing disparity estimation for image sequences, it is desirable that static parts of a scene remain at the same disparity, and thus are projected to the same coordinates of novel synthesized views throughout the sequence. This leads to a less flickering result, with a higher perceived quality.

In DERS, this is achieved by updating the cost function before graph cuts optimization using disparity values of the previous frame in combination with block based motion detection where a block motion map is created for the reference view.

Motion detection in DERS is performed by first applying a noise-reducing bilateral filter on the current and previous frame of the reference view. Then, the frames are divided into $16x16$ pixel blocks. Motion within a block is determined by subtracting intensities between a block of the current frame and the same block of the previous frame, followed by a computation of the *mean absolute difference* (MAD) value. If the MAD value of a block is above a user specified threshold, motion is considered to be present.

The data cost for disparities within static blocks is set to zero for the previous disparity values of pixels in these blocks, thus encouraging the selection of the same disparity values again when performing graph cuts optimization for the new frame. The cost for selecting other disparities than the previous ones within static blocks is also slightly raised to further increase the chance of selecting the same disparity again.

A good side effect, apart from achieving temporally stable results, is that the graph cuts optimization converges faster when using this method. In fact, experiments show a decrease of as much as $60 - 80\%$ in computational time.

7.1.2 Using three views instead of two

When estimating disparity maps for sequences using DERS, three camera views are used – *left*, *center* and *right* views. These cameras are normally placed horizontally aligned and are separated with equal baseline distances.

The center view is selected as reference view and the additional view is used when computing the cost function. By denoting the intensity functions of the three views I_L, I_C and I_R, the cost function of DERS may be written as:

$$C(x, y, d) = \min\left(|I_L(x + d, y) - I_C(x, y)|, |I_C(x, y) - I_R(x - d, y)|\right) \quad (7.1)$$

The largest benefit of introducing a third view into the cost function as in (7.1) is that the problem with occlusion is somewhat reduced.

7.2 Quality assessment for real world images using prediction

7.2.1 Implementation of a prediction error metric

In most cases, there is no available ground truth for real world stereo images. Quality assessment of disparity estimates using prediction was therefore implemented by warping reference image pixels to their corresponding positions in the search image view (which is referred to as *forward prediction*).

To compensate for occlusion, meaning that some regions are not projected onto, the error was set to the peak signal value for these areas. This means that an algorithm resulting in a low ratio of unique projection from the reference to the search view will be penalized.

After the warping, the *root mean square error* (RMSE) and the related *peak signal-to-noise ratio* (PSNR) are computed. These are defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{(x,y)} \left| I_{original}(x,y) - I_{predicted}(x,y) \right|^2} \tag{7.2}$$

$$PSNR = 20 \cdot \log_{10} \left| \frac{\max\limits_{(x,y)} I_{original}(x,y)}{RMSE} \right| \tag{7.3}$$

When color images are used, the RMSE value is computed by summing the square differences for all color channels, and dividing the total sum with three before computing the square root. The peak signal is computed by taking the maximum intensity value found in the available color channels.

7.2.2 Result and conclusion

(a) (b)

Figure 39. Forward prediction of Venus images: a) Forward prediction using ground truth b) Forward prediction using noisy disparity map created from 3x3 window block matching and WTA optimization

When not taking occlusion into consideration, and only computing the error for regions in the search image view that are actually projected onto, a noisy disparity estimate created from $3x3$ pixel block matching and WTA optimization performs better than the ground truth. This is also the case for

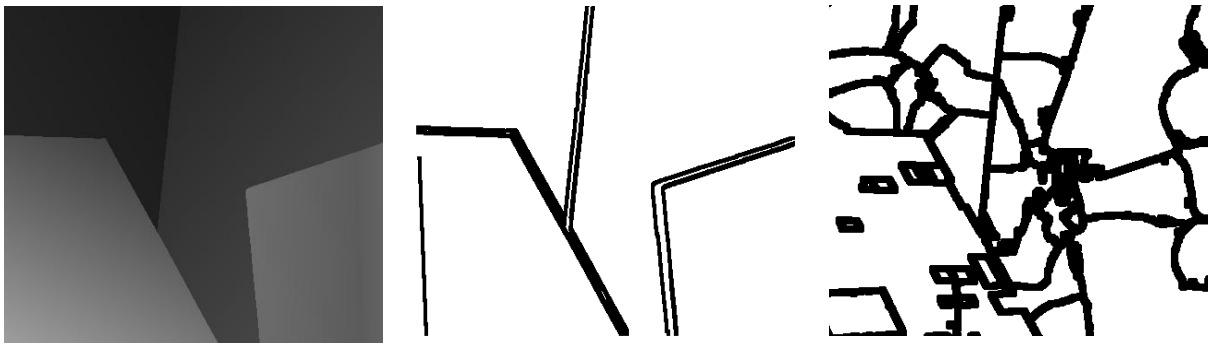
inverse prediction[10] without occlusion handling. The behavior is explained by that WTA optimization simply minimizes a cost function based on intensity differences, which is what is measured by the prediction error metric.

For the forward prediction, with occlusion handling, the PSNR value for ground truth disparity was 14.0 dB and the noisy WTA estimate gave a PSNR of 7.2 dB. Thus, this measure was considered to be reasonable.

An additional measure that can be considered is the amount of used information in the prediction. The ground truth warps 96% unique pixels from one view to another, while the noisy disparity estimate only warps 82% unique pixels.

Prediction error measures cannot be compared with measures based on ground truth data and should only be used for guidance. Subjective quality evaluation will therefore be necessary to see whether results are reasonable or not. More information on prediction error metrics is to be found in e.g. [9] and [10].

7.2.3 Prediction error for discontinuous regions



(a) (b) (c)

Figure 40. a) Venus groundtruth b) Discontinuous areas part of the groundtruth material released on the Middlebury stereo homepage c) Filtered edge map created by performing edge detection on a label map created from image segmentation

An attempt to measure error in discontinuous regions, similar to in the Middlebury evaluation, was also made. To do this, the resulting label map after heavy image segmentation was filtered with edge detectors. The edge map was then smoothened with an averaging filter of size $7x7$ to broaden the border regions, where the discontinuities reasonably should be present. However, the results from this approach were highly ambiguous. Image segmentation could be used rather successfully to find discontinuous regions, though.

[10] Inverse prediction simply 'pulls back' pixels from the search view to the reference view using the estimated disparity map. As a consequence, ghosting effects occur in disparity discontinuous regions, see [**10**] for further details.

7.3 Multi-view image sequences used in the experiments

Mainly, four multi-view image sequences were used in the experimenting. They are downscaled versions of original high resolution sequences released by *Nagoya university* and used in MPEG related research regarding FTV and MVC.

Figure 41. Multi-view sequences used in the experiments, originating from Nagoya university: a) Pantomime b) Lovebird1 c) Newspaper d) Bookarrival

The used image sequences are all filmed with static camera rigs. All sequences offer challenges, e.g. the background in *Pantomime*, the trees in *Lovebird1* and the poorly textured walls of *Newspaper* and *Bookarrival*.

Sequence name	Left camera	Center camera	Right camera	Resolution	Disparity search range
Pantomime	36	38	40	$640x480$	$1-17$
Lovebird1	3	4	5	$512x384$	$0-10$
Newspaper	0	2	4	$512x384$	$15-42$
Bookarrival	10	8	6	$512x384$	$14-36$

Table 9. Specifications for the multi-view sequences used in the experiments.

Disparity search ranges for the sequences were determined by manual inspection of the absolute differences of two adjacent camera views of the sequences, and measurement of the smallest and largest disparity values seen. After that, an extra margin of some $3-5$ pixels was subtracted from/added to the ranges.

7.4 Disparity for real world images using the implemented algorithms

The implemented stereo algorithms were tested on a frame of the *Pantomime* sequence.

7.4.1 The implemented stereo algorithm inspired by AdaptingBP

(a) (b) (c)

Figure 42. Pantomime: a) Result when using same parameters as for Middlebury images b) Result when increasing the block matching window size to $13x13$ and RANSAC σ to 0.7. c) A slice of the cost function for background disparity. The left clown in the right image leads to a high cost for correct disparity in the occluded region. The high cost for this occluded region at the 'correct' background disparity is illustrated in yellow/red.

The implemented algorithm based on image segmentation, plane fitting and reassignment of plane parameters is capable of giving results with only a few percent bad pixels for the Middlebury stereo sets. However, when run on a real world image pair, the situation is different. When running it on a frame from the *Pantomime* sequence, the results are rather poor in that the background gets assigned the same disparity as the clowns. This also the case when the block matching window size and RANSAC parameters are adjusted.

Most likely, the explanation lies within the characteristics of the Pantomime image pair combined with using an energy minimization approach. Note that the area immediately to the left of the clowns in the left view is occluded in the right view. Assigning a correct disparity to the background plane would lead to producing cost values based on absolute differences between the black background and the left part of the highly textured clowns, which are shown in Figure 42 (c). In Figure 42 (a) and (b), the cost has been reduced and an error is introduced at the right image border. As the background is untextured, this leads to a lower energy in total.

7.4.2 Simplified algorithm using WTA and plane fitting

While tuning the parameters of the previous experiment, it was noticed that the algorithm could produce rather good intermediate result after WTA optimization and the first plane fitting, if skipping the subsequent refinement steps. This also reduces the computational time of the algorithm.

A less complex algorithm, using only the WTA optimization and plane fitting step, was created for real world image pairs. As the final clustering and refinement steps of the originally developed algorithm will only fine tune the result, the idea would be that terminating after the first plane fitting could still give a decent result.

(a) (b)

Figure 43. Pantomime: a) Disparity estimate after WTA optimization and RANSAC plane fitting b) Left-to-right forward predicted view. Areas not projected onto are painted in yellow.

After some parameter testing, the result looked fairly well. However, the RANSAC algorithm sometimes lead to badly fitted planes for lowly textured segments such as the background in Pantomime.

7.4.3 Graph cuts energy minimization

Parameters were determined experimentally in order to find decent values and an example of the outcome is shown below.

(a) (b)

Figure 44. Pantomime: a) Disparity estimate after GC optimization b) Left-to-right forward predicted view

Energy minimization using graph cuts was performed for the same set of parameters as tried for the Middlebury images. One of the better results for *Pantomime* is shown in the figure above and, as can be seen, there are problems near the clowns. These problems are likely caused by similar reasons as those discussed in Section 7.4.1.

The cost for assigning an erroneous disparity to the background between the right clown's arm and body is likely lower than the smoothness energy increase that would be the consequence if the discontinuous disparity region would follow the entire side of the arm and body.

7.4.4 Local matching with segment based support windows

While experimenting with the Pantomime sequence, the idea to use support windows shaped like image segments was tried, because of the problematic textureless background. The segment based matching technique proposed in [40], which works as described in Section 3.4.4, was implemented. For implementation details and results for the Middlebury image sets, see Appendix C.

(a) (b)

*Figure 45. Pantomime: a) Result from segment-based matching as proposed [**40**] b) A problematic area, the dark part shows high values of the cost function for the disparity of the background. This dark area will be assigned the same disparity as the clown, even though it belongs to the background.*

The benefit of the technique is that for very large segments, such as the background in Pantomime, a large local matching window is used. Large segments are caused by textureless regions, so the idea seemed promising. However, the result was not as good as hoped for, as seen in Figure 45 (a). The borders of the clowns still cause problems, and the background gets poorly estimated although a large aggregation window of size $51x51$ pixels is used.

To reduce the problems caused by occlusion on the clowns' left sides, a version which uses segmentation data from *both* the left and right image was implemented. The segments that overlap the most for each disparity are assumed to be corresponding, and their common pixels are weighted higher in the aggregation than other pixels. This extended version of the algorithm performed better at the clowns' edges, but the result was still not good. Also, the computational complexity rises significantly when segmenting both views. Figure 46 illustrates the approach.

(a) (b)

Figure 46. Pantomime: a) In an extended version of the segment matching algorithm, occluded segments between the background and the clown were suppressed through the use of both left and right image segmentation. Here, the overlap of segments at background disparity is shown, and only pixels of the red segment are weighted high in cost aggregation of the background. b) Result from extended version using segmentation data from both the left and right view.

7.4.5 Conclusion

The implemented algorithms, which can match the state of art algorithms on the Middlebury Stereo homepage [4] for several image pairs traditionally used in research, fail to produce good result for a stereo image pair originating from the Pantomime sequence.

A simplified version using WTA optimization and plane fitting performed reasonably well, but sometimes had problems with a badly fitted plane for the large segment representing the background.

Graph cuts minimization of an energy function defined on pixel level gave decent results. This method basically has a built in handling of unreliable pixels. If the data cost is almost equally high for all disparities, as for poorly textured regions, the smoothness cost guides the disparity estimation.

The conclusion from testing the implemented stereo algorithms on real stereo image pairs is that many methods probably handle the stereo image pairs available on the Middlebury homepage rather well, but encounter severe difficulties when real world images are being processed.

As has been discussed in other similar work, e.g. [9], minimization of a global energy function formulated on a pixel level can handle real world stereo problems quite well in many cases.

7.5 Implementation of motion detection

In DERS, motion detection is currently performed by taking *mean absolute differences* (MAD) differences of relatively large blocks (of $16x16$ pixels). These large blocks are then either considered to be moving or static. An idea that emerged during the project was to implement more precise motion detection and try some possible usages for it.

Two approaches were tested. One approach is using image segmentation to define aggregation windows for computing MAD and the other is based on a simple square averaging window for aggregation.

7.5.1 Ideas on using motion detection

One possible usage for more precise motion detection is to decrease noise in an image sequence by averaging pixels over static regions. This could be seen as a denoising preprocessing step and should reasonably lead to better disparity estimation.

Another idea is to only reestimate disparity for areas where motion is present, thus reducing the computational time for generating a sequence of disparity images. This would also give more temporally consistent disparity sequences.

A third idea is to average disparity over time for static areas. Just as for the previous idea, this would lead to better temporal consistency, and hopefully to better estimated disparity for static regions as well.

7.5.2 Implemented motion detection approaches for computing MAD

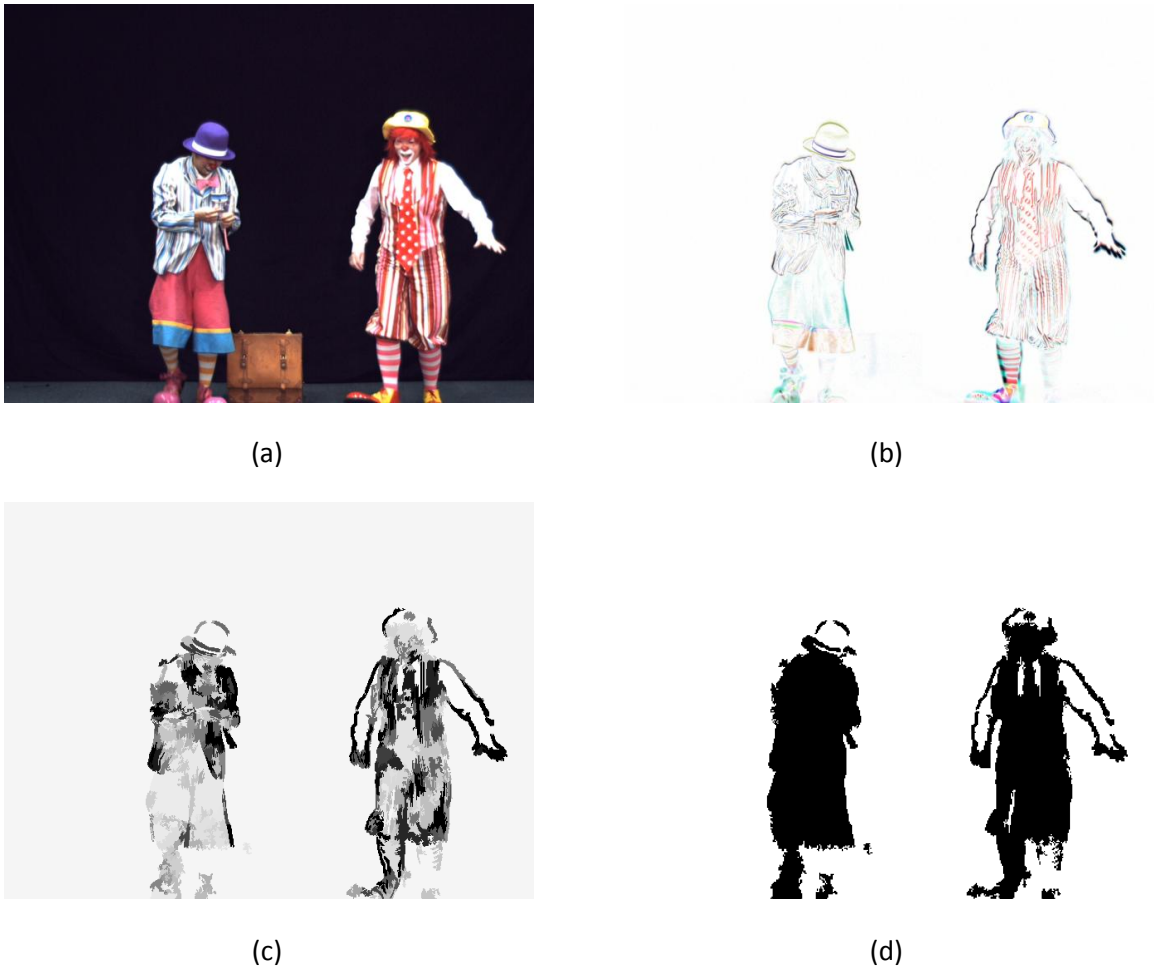
Segment based motion detection was implemented by performing image segmentation on the differences between adjacent frames. Motion was detected by using the resulting segments as blocks for the computation of MAD values. The MAD value of segments was then compared against a manually determined threshold set with a margin above the noise level for the video sequence, and segments where the threshold is exceeded were considered to contain motion. When segmenting differences between frames, it was only reasonable to compute the differences for adjacent frames as segmenting the differences between several pairs of frames would be too computationally demanding.

Square window based motion detection was implemented using a $9x9$ square averaging window to compute MAD values for all pixels. This window size was determined experimentally and as earlier in the project, convolution was used to perform the aggregation. Pixels where the manually determined MAD threshold was exceeded were considered to have changed/moved. When using this faster and simpler technique, the possibility to look at differences between the current and several previous frames was added.

Finally, both absolute intensity differences and absolute gradient differences between frames were tried. It was found that a combination of the two was necessary for motion detection to work properly on sequences such as *Lovebird1*.

7.5.3 Usage of MAD values: Motion maps and static maps

When the computed MAD values are compared against a threshold, pixels with values exceeding the threshold are labeled as moving and pixels with value below the threshold are labeled as static. This result is saved in a *motion map* and the principle is illustrated in Figure 47 for the approach when segmentation regions are used to compute MAD.



(a) (b)

(c) (d)

Figure 47. Pantomime sequence: a) Frame #25 b) Absolute differences between frame #24 and #25 c) MAD values for segmented absolute differences between the two frames d) Motion map showing segments with motion, found by comparing the MAD values against a threshold, in black

As it would be interesting to know the number of frames in a row that a certain pixel (x, y) has been considered as static by the motion detection, an idea to compute two motion information maps for every frame was implemented. These maps are here denoted *static maps* and store temporal consistency information for frames forward and backward in time. Every pixel of the computed static maps contains the number of previous and forward frames in which the pixel has been static. This information could be used for temporal averaging.

(a) (b)

Figure 48. Pantomime sequence: a) Segment-based motion information b) Square window based motion information

Some results from this implementation are given in Figure 48, where a backward static pixel map is shown for frame #10 using both segment-based and square window based motion detection. White means that an area has been static between all pair of frames until frame #10. Black means that motion has been present since the last frame, i.e. between frames #9 and #10 in the figure. The color scale ranging from yellow to red then gives the number of frames for which the pixel has been labeled as static. Note that the movement over time for the two clowns can be identified in the static pixel map.

7.5.4 Averaging of input sequence intensity values

The static maps were used to average static pixels over several frames in order to reduce the noise level. This could for instance be done prior to disparity estimation as a preprocessing step, with the goal to enhance quality.

For the experimenting, 50 frames of the Pantomime sequence were used, and averaging was performed with different time horizons for frame #25.

(a) (b)

(c) (d)

Figure 49. Pantomime sequence (zoomed), averaging of frame 25: a) No averaging b) Segment based, horizon 25 frames forward/backward c) Square window based, horizon 25 frames forward/backward d) Square window based, horizon 5 frames forward/backward.

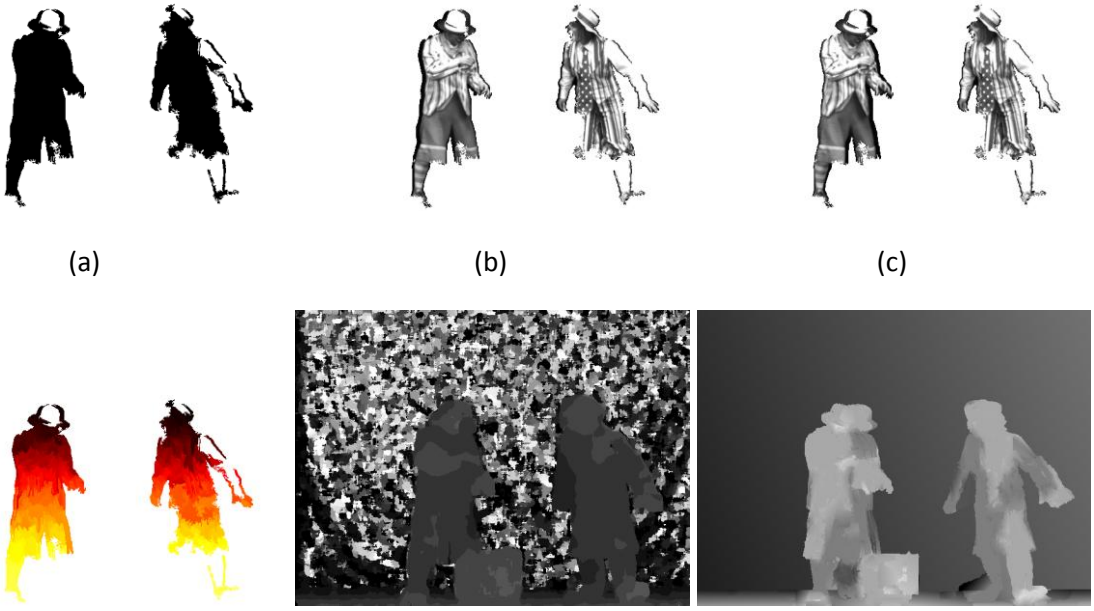
When segmentation based motion detection was used, the results indicate that the isolation of moving areas is actually too precise, and that this leads to artifacts as seen in Figure 49 (b).

Square window based motion detection seems to perform better, reasonably explained by that its less precise nature gives a margin around moving objects. Hence, bleeding object edges such as the clowns' bright clothes are not taking part in the averaging. A good time horizon would reasonably be a few frames forward/backward.

7.5.5 Segment-based motion detection, WTA-optimization and plane fitting

An idea to use segment-based motion detection and reestimate disparities using WTA optimization and plane fitting for changed segments only was tried. This would significantly reduce the time consumption of the plane fitting step. However, the idea failed.

Although the motion detection worked quite well, problems arose because of the smoothening effect of the clowns introduced by the cost aggregation (the so called *foreground fattening effect*).



(a) (b) (c)

(d) (e) (f)

Figure 50. Pantomime sequence: a) Detected motion b) and c) Visualisation of the clowns' movements between the two selected frames d) Segments created in the motion detection and labeled as moving e) WTA disparity estimate from 13x13 window cost aggregation f) After 10 frames: The foreground fattening effect apparent in (e) gives wrongly fitted planes in the black regions of (c), which leads to an erroneous result.

7.5.6 Square window based motion detection of differences between several frames

Up until now, motion detection had been using differences between adjacent frames only as this worked well for the Pantomime sequence. When trying it on other sequences, such as Lovebird1, the results indicated that differences between a current and several previous frames had to be used in order to successfully detect motion. An example of the problem is shown in Figure 51. Note how the center part of the man's body is registered as static between all pairs of frames, even though he clearly has moved.

(a) (b) (c)

Figure 51. Lovebird1 sequence (zoomed): a) Frame #1 b) Frame #10 c) Backward static pixel map for frame #10.

To correct the discovered problem, every frame was compared with several previous frames. This means that MAD values were computed for several unique pair of frames for each frame. To maintain a reasonable computational time, segment-based motion detection was abandoned.

The square window based motion detection was experimentally changed until it worked for all sequences, ending up in the following outline:

- Absolute differences between frames was based both on image gradient values and intensity values, weighted equally much. Also, color information had to be incorporated, and not just the Y-component of the YUV-format.
- The number of previous frames to use was experimentally determined to be 3. An average of the differences between frame pairs $(t, t-1)$, $(t, t-2)$ and $(t, t-3)$ was used when computing MAD for frame t.
- MAD values for different sequences were manually determined by investigating the noise level. As differences between the current and several previous frames were looked upon, the static map computation was changed to only look temporally backwards.
- The window size used for MAD filtering was $9x9$ pixels, and the created binary motion maps were smoothened with a $11x11$ pixel averaging window.

After these adjustments had been made, the motion detection worked rather well for all of the used sequences. Results for Pantomime and Lovebird1 are shown in Figure 52.

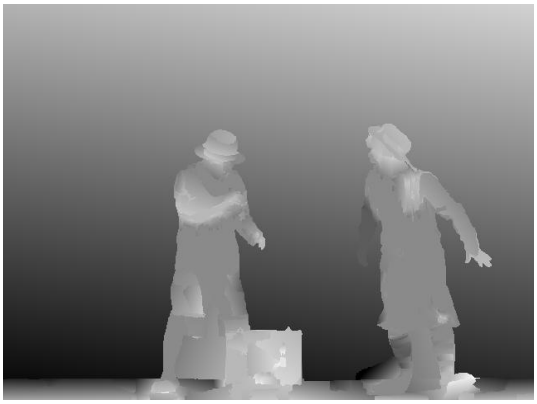
(a) (b)

Figure 52. a) Backward static pixel map for frame #10 of the Pantomime sequence b) Backward static pixel map for frame #10 of the Lovebird1 sequence

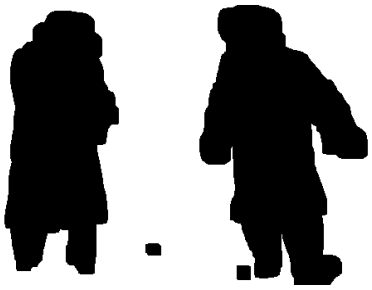
7.5.7 Square window based motion detection, WTA-optimization and plane fitting

A new attempt was made to reestimate disparity of moving regions only when instead using square window based motion detection as described in the previous section.

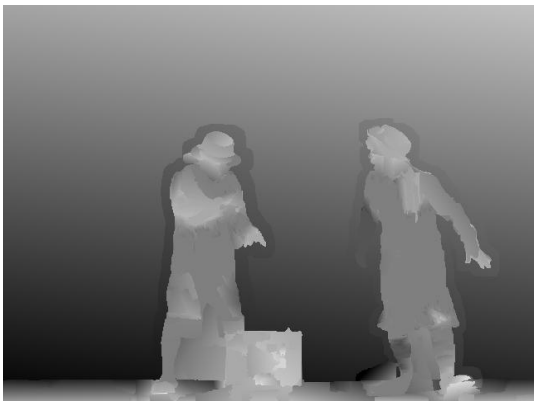
For this approach, image segmentation and plane fitting was used for the complete frame, though, as there was no good way available for doing this only for the parts that had changed. For instance, the foreground fattening effect discussed in Section 7.5.5 would again be a problem for the background close to the clowns if only reestimating disparity locally. The main benefit with this idea would therefore be temporal stability over time.



(a)



(b)



(c)

(d)

Figure 53. Pantomime sequence: a) Estimated disparities for the first frame, with a rather badly fitted plane b) Motion map showing regions of movement between the first and second frame c) The moving regions have been filled in with newly estimated disparities, note the border around the clowns d) The disparity map for frame #50, note that it is possible to see traces of how the clowns have moved over the static background.

The principles of the approach are shown for the Pantomime sequence in Figure 53. Also, an example is shown for the Lovebird1 sequence in Figure 54.

(a)

(b)

Figure 54. Lovebird1 sequence: a) Disparity map for frame #1 b) Disparity map for frame #50. Note that the disparity of the trees and the house in the background is identical to the case of (a), and that reestimation has only been done close to the walking couple.

7.5.8 Results and conclusion

Segment based motion detection provides adaptive window shapes for computing MAD values. This results in a better delineation of moving areas, because the edges of moving regions remain intact. The idea of segment based motion detection did not work well, though, and in fact it is desirable to have a margin around regions containing moving pixels. Thus, square window based motion followed by a smoothening of the computed motion maps is both preferable and less computationally expensive.

Also, looking only one frame backwards when computing differences was not enough, and instead three previous frames needed to be used in order to get more reliable results.

7.6 Implementation of multi-view disparity estimation

The stereo algorithms implemented earlier within the project were extended to use a cost function based on three input views. The cost function is computed in the same way as in DERS (see equation (7.1)). Apart from a changed cost function, the only modification done is that the cross checking was extended to use left-to-center and right-to-center comparison when using WTA optimization. For graph cuts, no further modifications were needed.

Three ideas were also tested to make disparity maps more temporally stable. One idea is similar to what was described in Section 7.5.7, another idea is based on averaging disparity of static regions and the third idea is based on global optimization. All ideas use the fast, square window based motion detection as described in Section 7.5.6 to determine static regions.

7.6.1 Global energy minimization where only changed pixels take part

To achieve temporally stable results similar to those outputted by DERS but with less computational complexity, an idea to only set up graphs for pixels of non-static regions was tried.

The energy function was the same as had been used before in the graph cuts stereo implementation, using image segmentation data to set smoothness cost. The TRW-S optimization algorithm was selected as arbitrarily shaped graphs are easier to specify with its interface than with the graph cuts algorithm. An initial experiment was run using the Pantomime sequence.

(a) (b)

Figure 55. a) Moving regions for which disparity should be updated b) Disparity created from TRW-S energy minimization

It was found that for larger moving regions, the TRW-S algorithm ran out of memory. This may be explained by that algorithms based on message passing tend to be more memory demanding than graph cuts algorithms.

The used graph cuts algorithm was investigated and it was found to support specifying indexes of pixels that should take part in the optimization. However, experiments showed that this feature did not work properly.

Due to time limitations, this part of the project had to be left as a topic for future work. Another idea that could be of interest when examining this approach further is the possibility to reduce the search range, and thus the number of labels for nodes of the created graphs.

7.6.2 Intensity and disparity averaging

The intensity averaging is implemented similarly to the averaging which is described in Section 7.5.4. Only previous frames were used, though, for the reasons discussed in Section 7.5.6. An idea of averaging disparity for static regions was also implemented.

For the implemented averaging, motion maps are used to average intensity or disparity using up to five previous values for static pixels.

7.6.3 Temporally stable disparity maps

To see the potential of reestimating disparity only for small moving regions instead of performing full scale disparity estimation, a feature to only fill in moving regions with new disparity when using WTA optimization and plane fitting was implemented as explained in Section 7.5.7.

7.6.4 Experimental results for the multi-view implementation

Experiments were run on real world multi-view sequences using various techniques. The sequences were *Pantomime*, *Lovebird1*, *Bookarrival* and *Newspaper*. Names of implemented techniques are abbreviated as follows:

- *AVD* – temporal averaging of disparity
- *AVI* – temporal averaging of intensity
- *PF* – plane fitting is used
- *XC* – cross checking is enabled
- *TS* – 'temporal stability', i.e. the technique where only moving regions are filled with new disparity values

Quality assessment was performed by PSNR calculation in combination with subjective quality evaluation. The PSNR results for different combinations of techniques are given in Figure 56.

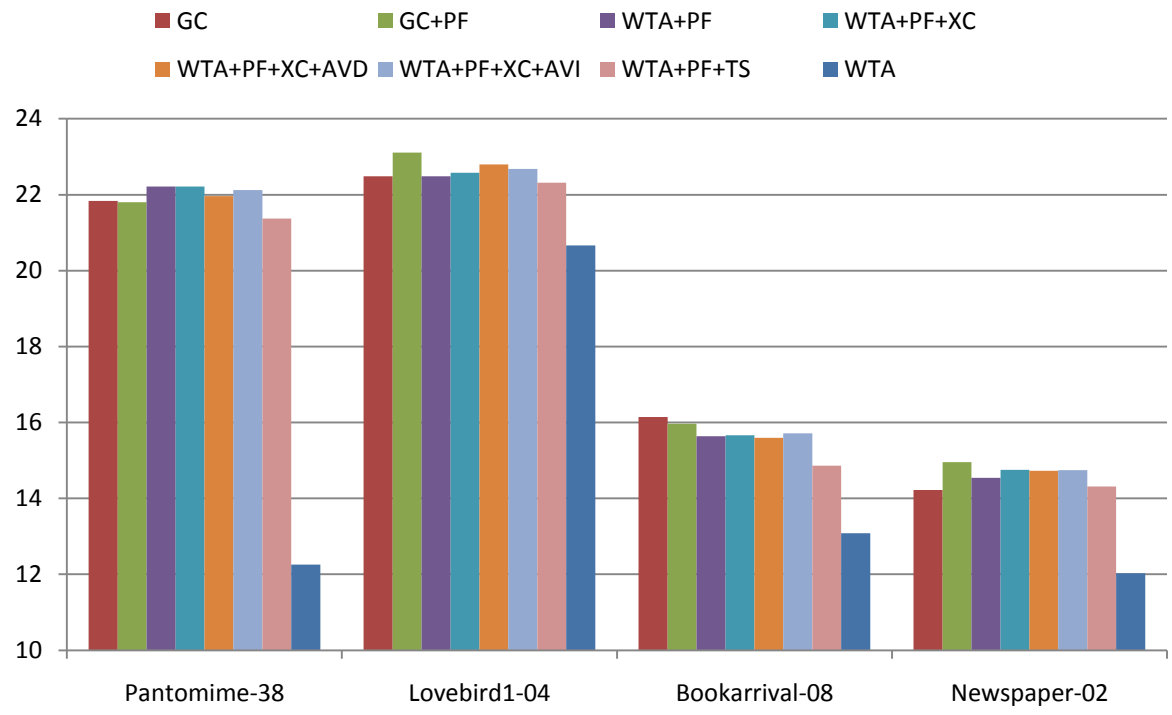


Figure 56. PSNR results for different combinations of techniques computed from forward (center-to-right) prediction with occlusion handling.

Some examples of how the produced results look like for the *Newspaper* and *Pantomime* sequences are shown in Figure 57 and Figure 60. In these figures, the difficulties that WTA optimization suffers from in lowly textured regions can clearly be seen. Plane fitting can in many cases significantly reduce this problem.

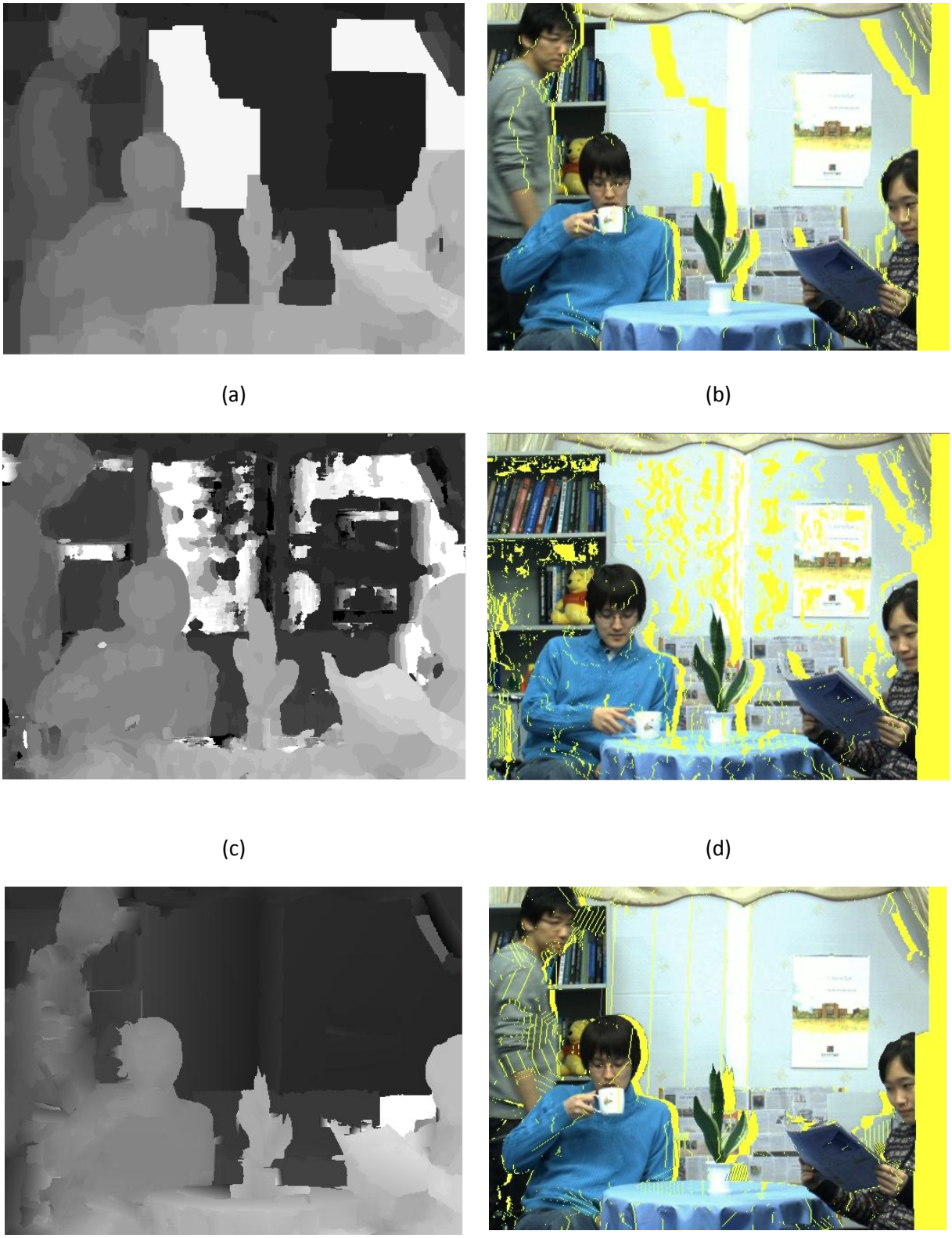
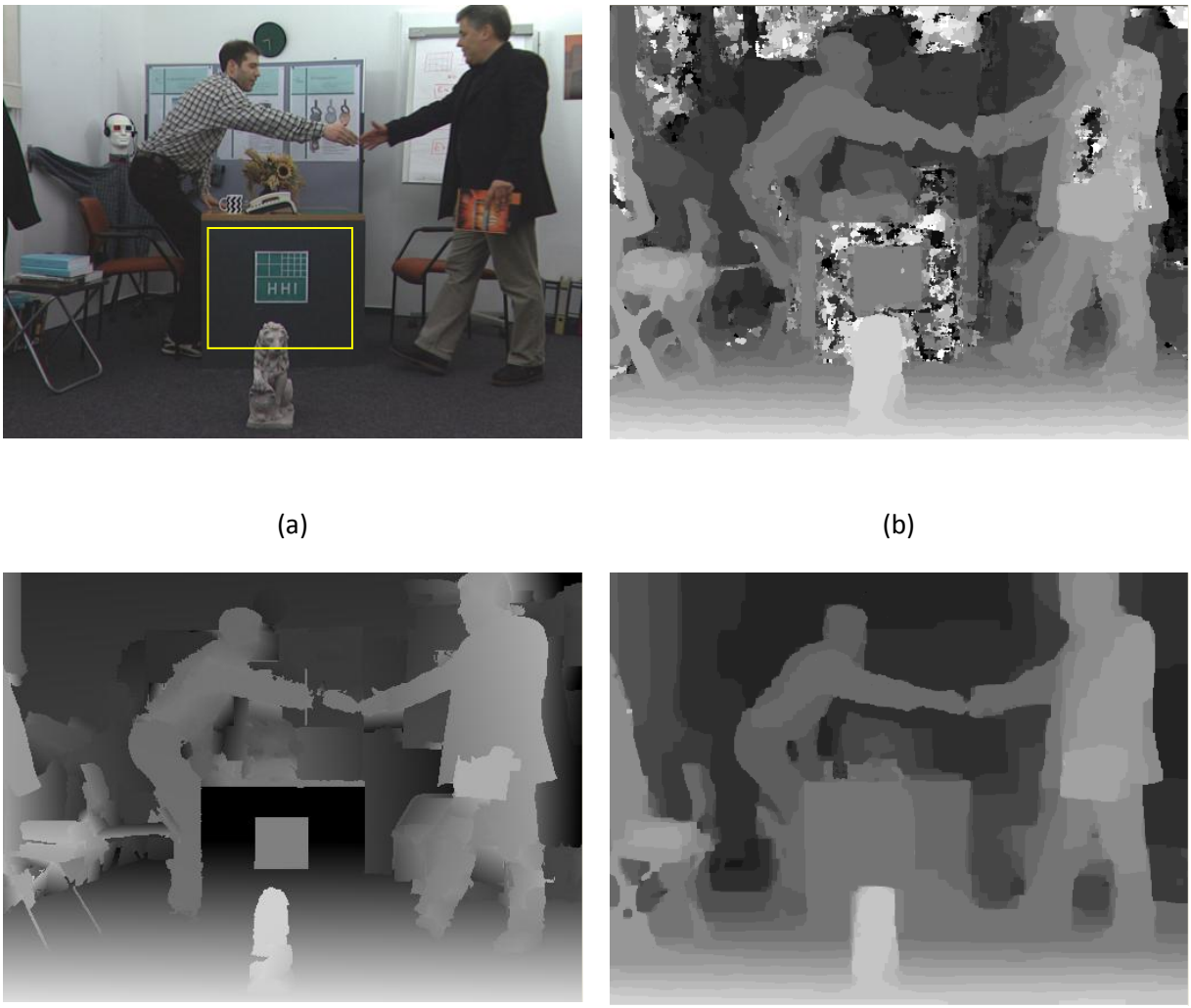


Figure 57. Newspaper sequence, disparity maps and forward predicted center-to-right views with disoccluded areas painted yellow: a-b) Results from GC optimization c-d) Results from WTA e-f) Results from WTA+PF

Figure 58. Pantomime sequence, disparity maps and forward predicted center-to-right views with disoccluded areas painted yellow: a-b) Results from GC optimization c-d) Results from WTA e-f) Results from WTA+PF

7.6.5 The strength of global optimization compared to local approaches

To show the main advantage of performing global optimization using an energy function formulated on pixel level, an example will be given from the Bookarrival sequence. This sequence contains large untextured regions that cannot be handled well with local WTA optimization. However, these regions have neighboring well textured regions, and for this kind of imagery the main benefit of GC energy minimization is easy to show.



(a) (b)

(c) (d)

Figure 59. Bookarival sequence (frame #34): a) The reference camera view with a yellow rectangle drawn around a problematic region b) Disparity estimate created from WTA optimization c) Disparity after a plane fitting (WTA+PF) d) Disparity estimate from GC optimization

Note the side of the desk in Figure 59 (a), marked with a yellow rectangle. To match pixels of this region is very problematic for a local method, as shown in Figure 59 (b). In principle, the lack of texture leads to the data cost being (almost) the same for all disparity values in this region. Thus, the WTA disparity estimate is very noisy.

A plane fitting can improve this WTA estimate to some extent, as seen in Figure 59 (c). However, the RANSAC algorithm occasionally leads to badly fitted planes, which is what has happened here as the desk's side is supposed to have a disparity value close to the disparity of the men.

The power of GC energy minimization is seen in Figure 59 (d). As there is a highly textured green area with white checks in the middle of the desk's side, the wrong disparity will have a high data cost here. This is also the case for areas close to the outer edges of the desk's side. Thanks to the energy function's smoothness term, these reliable regions lead to a propagation of correct disparity into the problematic areas.

7.6.6 Conclusion

When looking at PSNR values, it seems as if a local approach based on WTA optimization and plane fitting can match the performance of GC energy minimization. This is certainly also the case for some sequences. But when large untextured areas show up, local techniques cannot be recommended.

As was found earlier in the project, plane fitting helps to improve the quality of noisy WTA disparity estimates.

The result from the other implemented techniques is not easy to judge as the PSNR results are quite similar for all of them, and differences are hard to see when looking at the center-to-right predicted views.

What can be said, though, is that the temporal stabilization approach where only moving areas are filled in with new disparity leads to less flickering. Averaging disparity over time for static regions also leads to a less flickering behavior in the predicted/synthesized view.

8 Discussion

8.1 Stereo algorithm implementation

The implemented stereo matching algorithms can provide results on par with results from state-of-art algorithms posted on the Middlebury homepage for several image pairs, such as Cones and Teddy.

There was a problem with the Tsukuba image pair, though. The problem is most likely explained by the plane fitting as it was present both for the AdaptingBP [36] influenced algorithm and for the graph cuts energy minimization algorithm.

For the Middlebury homepage stereo image pairs, fast local matching together with image segmentation and plane fitting could in most cases match the performance of global energy minimization using graph cuts optimization of a pixel level energy function.

8.2 Real world image sequences

For the real world image sequences, the rather complex algorithm implementation influenced by the top performing algorithm on the Middlebury Stereo homepage did not perform very well. However, when terminating after an initial winner-take-all optimization and plane fitting, decent results were acquired for real world image data.

The graph cuts implementation gave more robust results for real world imagery, and as a remark, the performance for this implementation should practically be identical to the performance of DERS as the implementation outline is the same.

To make use of temporal information, some ideas based on motion detection were implemented and tested. It had been interesting to only compute new disparity for parts of a scene where changes are detected, but the results were not successful because of shortcomings in the basic motion detection technique that was used.

Finally, it was proven quite hard to evaluate quality for real world image data. The simple PSNR measure that was used could only give an indication of the true output quality. Hence, subjective evaluation was necessary.

8.3 Computational complexity

Computational complexity is an issue within stereo matching. Some proposed algorithms come with information on computational time when run on benchmarking stereo image pairs on some specific computer. But since the implementations are hardly ever available to the public, algorithms cannot be compared against each other in practice.

It has also been hard to compare performance of components used within the project. This is mainly explained by that some functionality was written directly in the native language of MATLAB, while other functionality was written in C and converted to MEX format, which runs much faster. However, the possibility of reasoning about the computational complexity of algorithms remains.

Most of the algorithms investigated in the survey use global energy minimization of a function formulated on pixel level. Many also use a computationally demanding adaptive window approach

for cost aggregation and contain several additional refinement steps. It is thus likely that they are significantly slower than the algorithms implemented within this project.

DERS is reasonably faster than the survey algorithms as well. It uses α-expansion graph cuts optimization, which is considered to be both faster and more accurate than belief propagation according to a survey comparing optimization techniques [25]. There is potential for speeding up DERS, though, with one example being that the cost aggregation is straightforwardly implemented without using convolution.

8.4 General thoughts

Since the release of the taxonomy [1] of Scharstein and Szeliski, stereo researchers have started to assess quality by using available stereo image sets that are accompanied by ground truth disparity maps.

Some problems related to this is that two of the four benchmarking stereo image pairs are purely made up by either fronto-parallel or slanting planar surfaces. Also, the images are very small and practically noise free. This stands in stark contrast to real world imagery. Many algorithms that can deliver good results for the Middlebury stereo images will most certainly encounter severe difficulties for real world material.

Also, stereo algorithm output results are highly dependent on the used parameters. In other words, an algorithm with better tuned parameters can most likely beat a better technique where the parameters are less tuned.

8.5 Future work

There is a lot of work to be done in the field of stereo correspondence, both for still images and for image sequences.

Methods that are robust to noise and that can handle real world images well, while still maintaining reasonable computational time, need to be found. Hence, parallelization of stereo algorithms is an interesting area for future research. Many algorithm steps, such as adaptive window cost aggregation, could reasonably be parallelized.

For image sequences, the use of temporal information to reduce occlusion problems and to increase quality/speed would be interesting to look further into. Some proposed ideas were found, but it seems as if not much research has been made within this field. Also, several of the proposed ideas seemed to be extremely slow judging by their outlines.

Finally, quality assessment of results is another problematic area and it would be highly desirable to have real world image sequences with ground truth data. There are some synthetic image sequences that come with ground truth, but these are not really representative for real world scenes.

9 Bibliography

[1] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Proc. IEEE Workshop Stereo and Multi-Baseline Vision (SMBV 2001)*, pp. 131-140, 2001.

[2] M. Z. Brown, D. Burschka, and G. D. Hager, "Advances in computational stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 993-1008, 2003.

[3] Q Wei, Converting 2D to 3D: A Survey, 2005.

[4] D Scharstein and R Szeliski. (2002-) The Middlebury Stereo Vision Page. [Online]. http://vision.middlebury.edu/stereo/

[5] Fredric Huguet and Frederic Devernay, "A Variational Method for Scene Flow Estimation from Stereo Sequences," *Proc. Intl. Conf. on Computer Vision*, 2007.

[6] F. Liu and V. Philomin, "Disparity Estimation in Stereo Sequences using Scene Flow," , 2008.

[7] Michael Isard and John Maccormick, "Dense motion and disparity estimation via loopy belief propagation," 2005.

[8] Shushi Guan and Reinhard Klette, "Belief-propagation on edge images for stereo analysis of image sequences," *RobVis'08: Proceedings of the 2nd international conference on Robot vision*, pp. 291-302, 2008.

[9] R. Klette and S. Morales, "Prediction Error Evaluation of Various Stereo Matching Algorithms on Long Stereo Sequences," 2009.

[10] Richard Szeliski, "Prediction Error as a Quality Metric for Motion and Stereo," *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, p. 781, 1999.

[11] S. Smirnov, A. Gotchev, and et al, "3D video processing algorithms - Part I," , 2010.

[12] David A. Forsyth and Jean Ponce, *Computer Vision: A Modern Approach*, US ed ed.: Prentice Hall, 2002.

[13] Richard Hartley and Andrew Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed.: Cambridge University Press, 2004.

[14] Daniel Scharstein and Richard Szeliski, "High-Accuracy Stereo Depth Maps Using Structured Light," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 195-202, 2003.

[15] D. Marr and T. Poggio, "Cooperative Computation of Stereo Disparity," 1976.

[16] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 4, pp. 401-406,

1998.

[17] R. Szeliski, *Computer Vision: Algorithms and Applications*., 2010.

[18] Kuk-Jin Yoon and In So Kweon, "Adaptive support-weight approach for correspondence search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 650-656, 2006.

[19] A. Hosni, M. Bleyer, M. Gelautz, and C. Rhemann, "Local stereo matching using geodesic support weights," *Proc. 16th IEEE Int Image Processing (ICIP) Conf*, pp. 2093-2096, 2009.

[20] Q. Yang, C. Engels, and A. Akbarzadeh, "Near Real-time Stereo for Weakly-Textured Scenes," *BMVC*, pp. 80-87, 2008.

[21] Heiko Hirschmüller and Daniel Scharstein, "Evaluation of Cost Functions for Stereo Matching.," *CVPR*, 2007.

[22] Olga Veksler, "Efficient graph-based energy minimization methods in computer vision," 1999.

[23] Stuart Geman and Donald Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 721-741, 1984.

[24] Y. Boykov, O. Veksler, and R. Zabih, "Markov random fields with efficient approximations," *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, pp. 648-655, 1998.

[25] R. Szeliski et al., "A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-Based Priors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 6, pp. 1068-1080, 2008.

[26] Yuri Boykov, Olga Veksler, and Ramin Zabih, "Fast Approximate Energy Minimization via Graph Cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222-1239, 2001.

[27] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399-404, 1956.

[28] Andrew V. Goldberg, "Recent Developments in Maximum Flow Algorithms," in Proceedings of Scandinavian Workshop on Algorithm Theory (SWAT), 1998.

[29] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss, "Understanding belief propagation and its generalizations," pp. 239-269, 2003.

[30] Pedro Felzenszwalb and Daniel Huttenlocher, "Efficient Belief Propagation for Early Vision," *International Journal of Computer Vision*, vol. 70, no. 1, pp. 41-54, 2006.

[31] Vladimir Kolmogorov, "Convergent Tree-Reweighted Message Passing for Energy Minimization,"

IEEE Trans. Pattern Anal. Mach. Intell., vol. 28, no. 10, pp. 1568-1583, 2006.

[32] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603-619, 2002.

[33] Kernel density estimation. [Online]. http://en.wikipedia.org/wiki/Kernel_density_estimation

[34] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32-40, 1975.

[35] Marco Zuliani, "RANSAC for Dummies," , 2009.

[36] Andreas Klaus, Mario Sormann, and Konrad Karner, "Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure," *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, pp. 15-18, 2006.

[37] Zeng-Fu Wang and Zhi-Gang Zheng, "A region based stereo matching algorithm using cooperative optimization," *CVPR*, 2008.

[38] Martin A. Fischler and Robert C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381-395, 1981.

[39] P. H. S. and A. Zisserman, "MLESAC: a new robust estimator with application to estimating image geometry," *Comput. Vis. Image Underst.*, vol. 78, no. 1, pp. 138-156, 2000.

[40] Mark Gerrits and Philippe Bekaert, "Local Stereo Matching with Segmentation-based Outlier Rejection," *CRV '06: Proceedings of the The 3rd Canadian Conference on Computer and Robot Vision*, p. 66, 2006.

[41] Qingxiong Yang, Liang Wang, Ruigang Yang, Henrik Stewenius, and David Nister, "Stereo Matching with Color-Weighted Correlation, Hierarchical Belief Propagation, and Occlusion Handling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 3, pp. 492-504, 2009.

[42] L. Xu and J. Jia, "Stereo matching: an outlier confidence approach," , 2008.

[43] Q. Yang, R. Yang, J. Davis, and D. Nistér, "Spatial-depth super resolution for range images," *CVPR07*, 2007.

[44] Shai Bagon. (2010) Shai Bagon's MATLAB code homepage. [Online]. http://www.wisdom.weizmann.ac.il/~bagon/matlab.html

[45] EDISON project. (2010) Homepage of the Edge Detection and Image SegmentatiON project. [Online]. http://coewww.rutgers.edu/riul/research/code/EDISON/index.html

[46] Y. Boykov, O. Veksler, and R. Zabih, "Efficient Approximate Energy Minimization via Graph Cuts,"

, 2001.

[47] V. Kolmogorov and R. Zabin, "What energy functions can be minimized via graph cuts?," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 2, pp. 147-159, 2004.

[48] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124-1137, 2004.

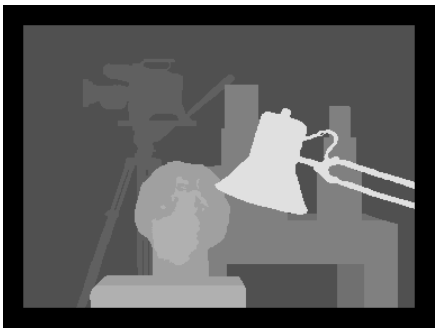
[49] Oliver Woodford, Philip Torr, Ian Reid, and Andrew Fitzgibbon, "Global Stereo Reconstruction under Second-Order Smoothness Priors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2115-2128, 2009.

[50] Oliver Woodford. (2010) Oliver Woodford's software homepage. [Online]. http://www.robots.ox.ac.uk/~ojw/software.htm

[51] M. Bleyer and M. Gelautz, "Graph-based Surface Reconstruction from Stereo-pairs using Image Segmentation," *SPIE*, vol. 5665, pp. 288-299, 2005.

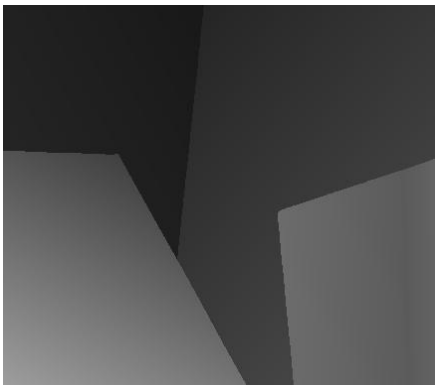
Appendix A. Middlebury stereo image pairs

Tsukuba reference view



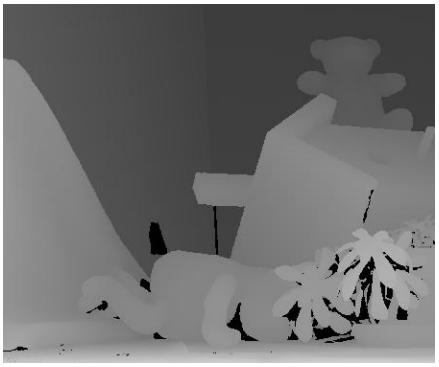
Tsukuba ground truth

Venus reference image



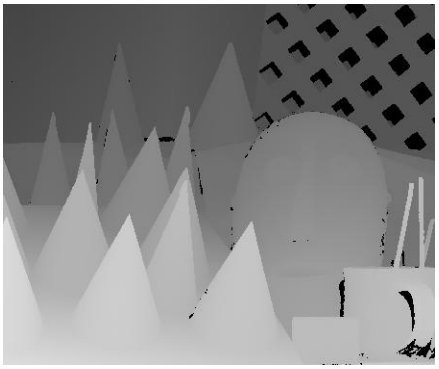
Venus ground truth

Teddy reference image



Teddy ground truth

Cones reference image



Cones ground truth

Appendix B. Stereo algorithm – experimental results

B.1 Winner-Take-All optimization

To examine the effect of different window sizes, compare AD with BT and to see the effect of using a gradient based cost function, exhaustive testing was performed. The parameter ranges were:

- Window sizes: $3x3$, $5x5$, $7x7$, $9x9$, $11x11$, $13x13$, $15x15$, $19x19$
- Dissimilarity measure of Birchfield-Tomasi: On/Off
- Gradient based matching cost: On/Off
- Gradient weights: $0 - 1.0$ in steps of 0.1

Tsukuba image set

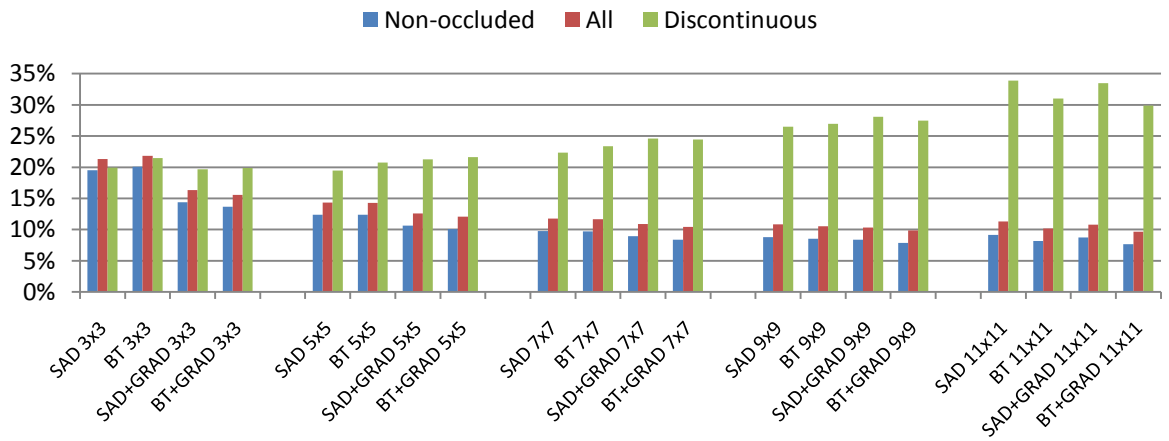


Figure 60. Percentage of badly estimated disparities in different regions of the Tsukuba image pair when performing WTA-optimization using different window sizes.

Venus image set

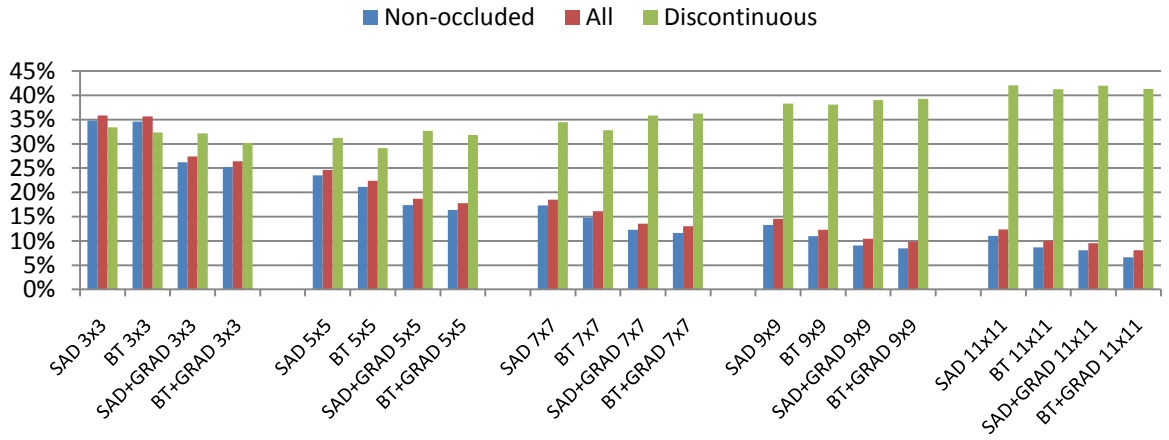


Figure 61. Percentage of badly estimated disparities in different regions of the Venus image pair when performing WTA-optimization using different window sizes.

Teddy image set

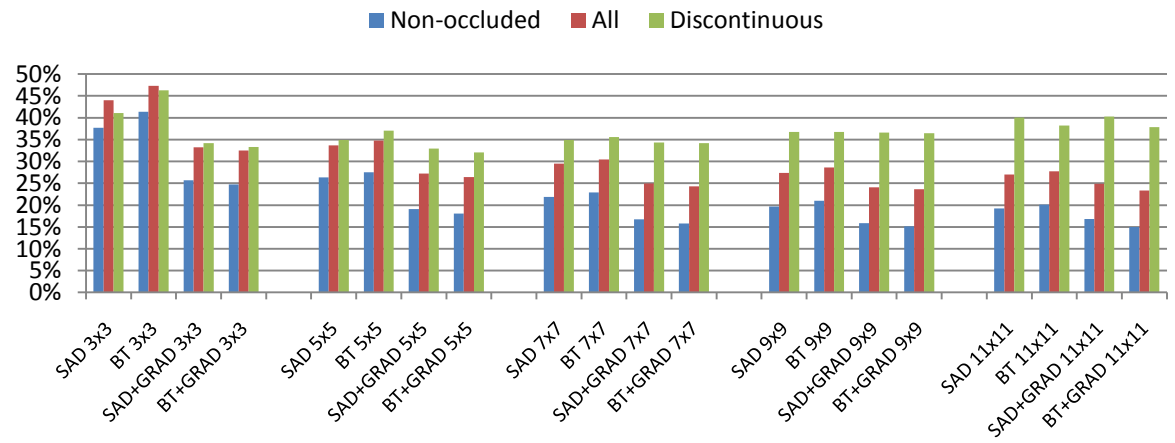


Figure 62. Percentage of badly estimated disparities in different regions of the Teddy image pair when performing WTA-optimization using different window sizes.

Cones image set

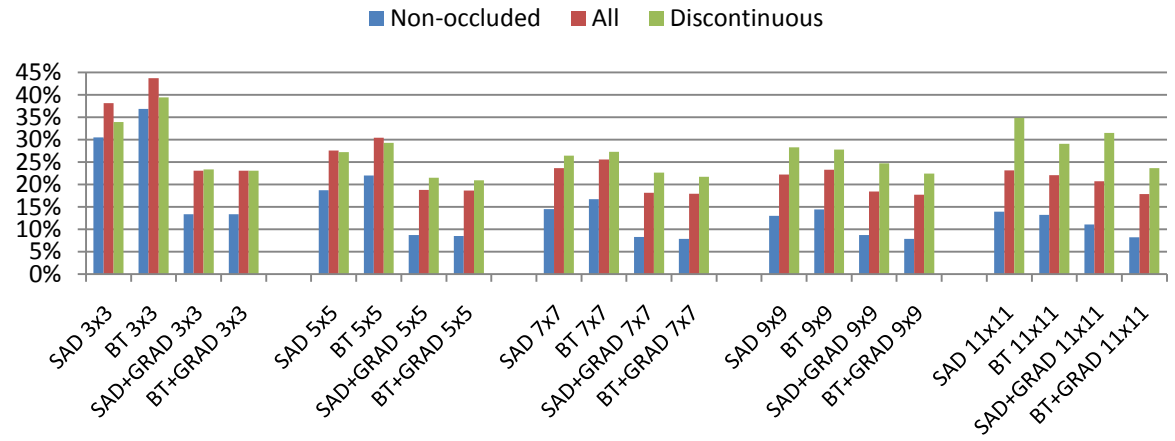


Figure 63. Percentage of badly estimated disparities in different regions of the Cones image pair when performing WTA-optimization using different window sizes.

B.2 Cost truncation

In this experiment, the implemented cost truncation is tested. Based on the previous results of the *local disparity estimation* experiment, BT was enabled and a gradient based cost function was used.

The parameters varied as follows:

- Window sizes: $3x3$, $9x9$
- Gradient weights: 0.7, 0.8, 0.9
- Automatic truncation threshold: On/Off
- Automatic truncation threshold percentage, when on: 70, 80, 90

B.2.1 Manual cost truncation

Cost truncation thresholds were set for two window sizes, 3x3 and 9x9 pixels. For these window sizes, the best available thresholds were found manually by looking at the amount of bad pixels. When the number of bad pixels starts to increase, this is a sign indicating that the threshold is set too low.

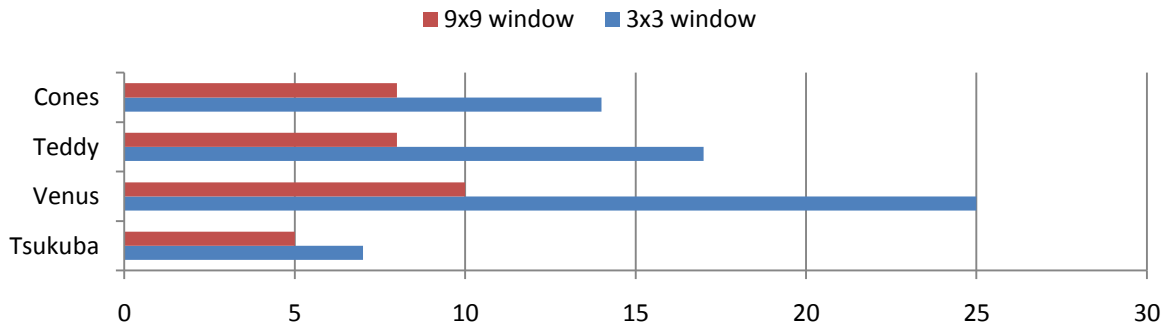


Figure 64. Manually determined cost truncation threshold values. The values were set so that a further decrease led to an increasing percentage of bad pixels.

The errors after WTA optimization for truncated costs were also compared against results for non-truncated costs and for results coming from automatically truncated costs, see next section.

B.2.2 Automatic cost truncation

Tsukuba image set

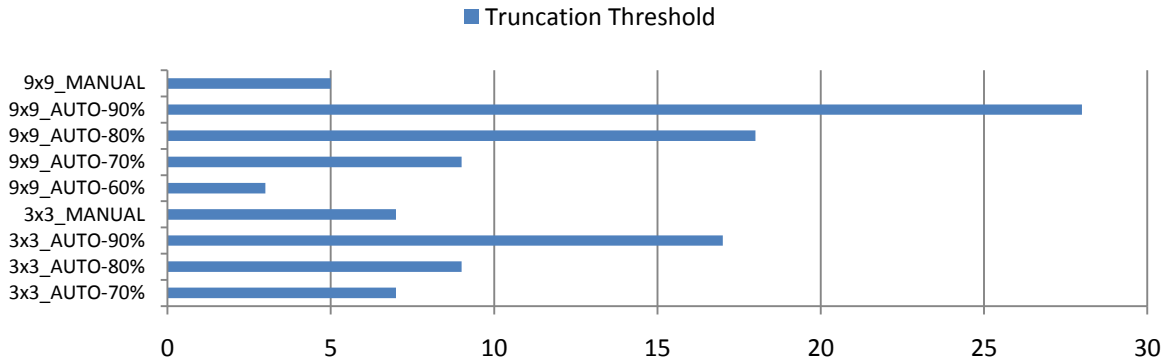


Figure 65. Results from automatically determining cost truncation values.

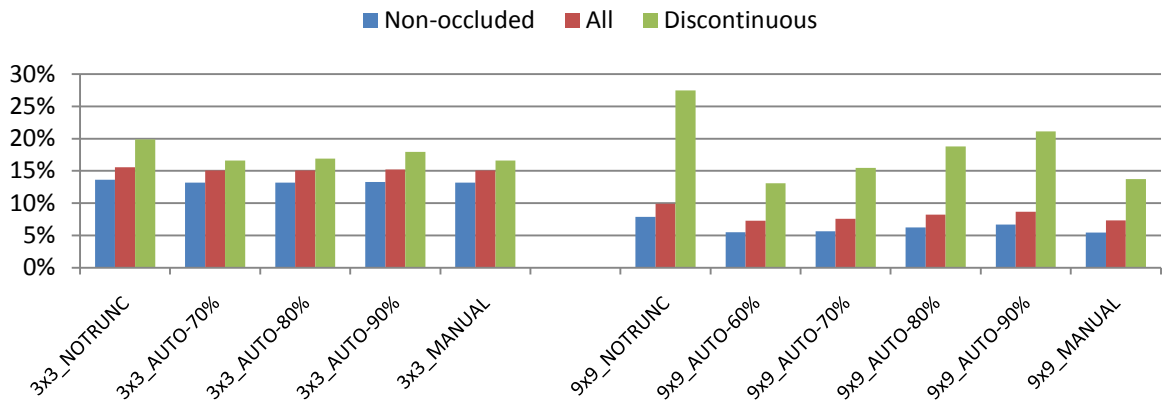


Figure 66. Percentage of bad pixels for a WTA estimated disparity when automatically determining cost truncation values.

Venus image set

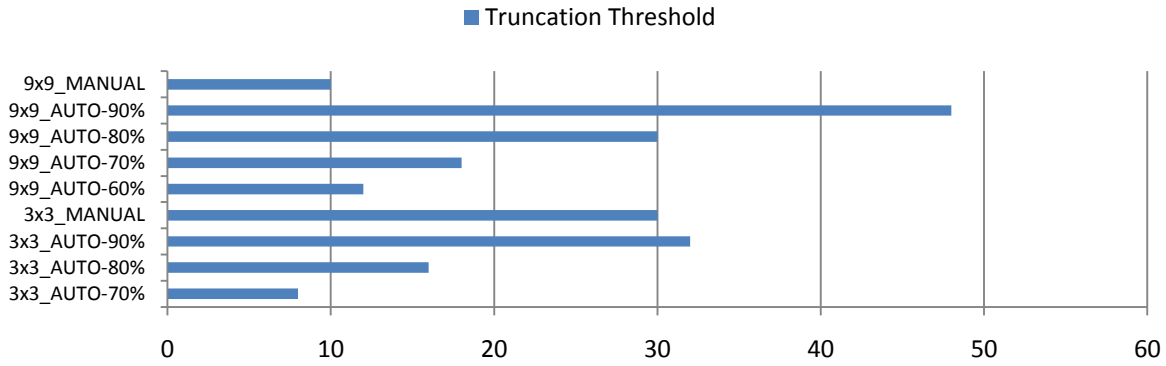


Figure 67. Results from automatically determining cost truncation values.

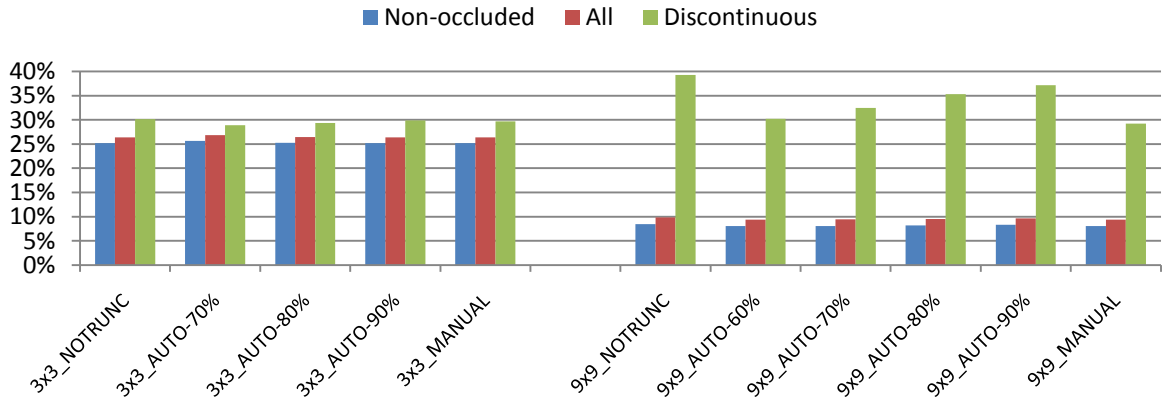


Figure 68. Percentage of bad pixels for a WTA estimated disparity when automatically determining cost truncation values.

Teddy image set

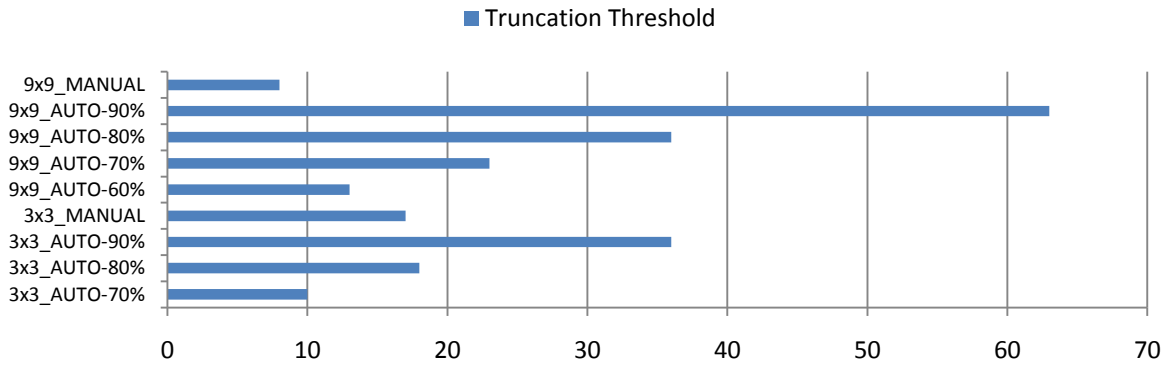


Figure 69. Results from automatically determining cost truncation values.

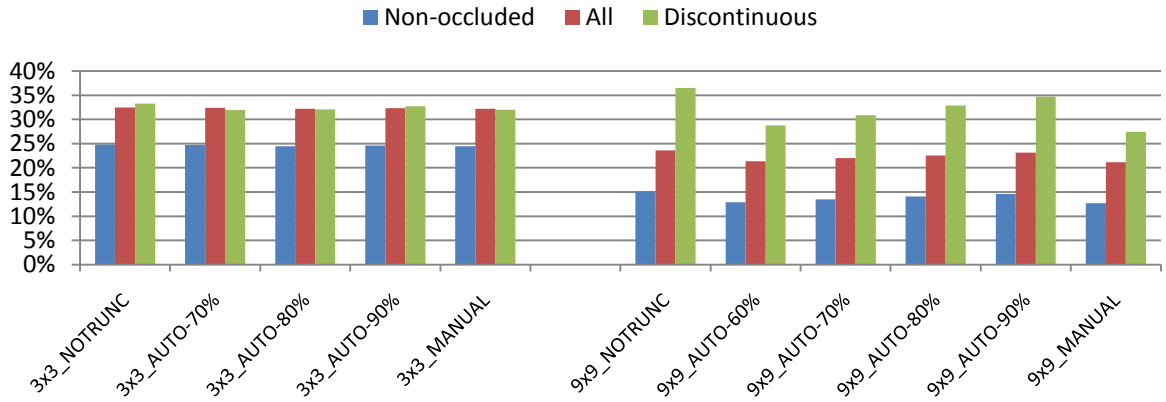


Figure 70. Percentage of bad pixels for a WTA estimated disparity when automatically determining cost truncation values.

Cones image set

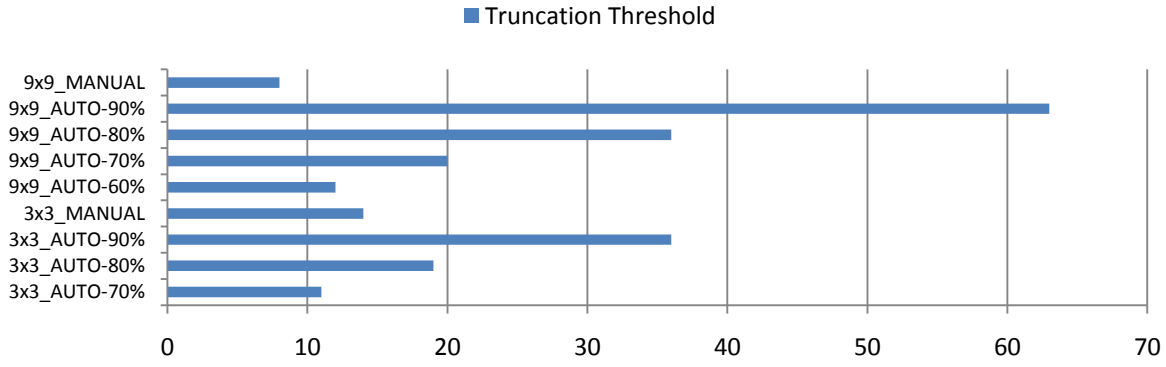


Figure 71. Results from automatically determining cost truncation values.

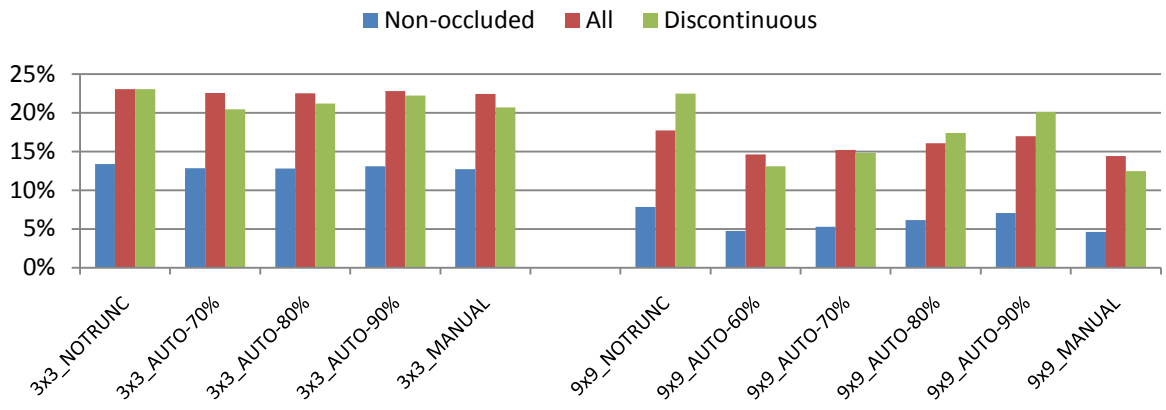


Figure 72. Percentage of bad pixels for a WTA estimated disparity when automatically determining cost truncation values.

B.3 Plane fitting

Based on the results of AdaptingBP [**36**], a small window size of $3x3$ pixels was used in the WTA optimization before plane fitting in these experiments. This small window size generates quite noisy disparity estimates, and the goal of performing plane fitting is to reduce the impact of the noise by robustly fitting planes in image segments using only disparity values that were validated in the cross checking.

B.3.1 Plane fitting using RANSAC/MSAC

In this experiment, the parameters varied as follows:

- RANSAC algorithm type: RANSAC, MSAC
- Error standard deviation: $0.1, 0.2, 0.3, 0.4, 0.5, 0.6$
- Probability that a data point is an inlier given that its distance to the fitted plane is below the threshold for the selected noise level, $P\left(\boldsymbol{d}_i \; is \; inlier \middle| e\left(\boldsymbol{d}_i, \boldsymbol{\mathcal{M}}(\boldsymbol{\theta})\right) < \delta\right)$: 0.99
- Probability of only selecting bad minimal sample sets, ϵ: 10^{-6}

The relation between specified standard deviation and the maximum allowed distance from the plane surface for an inlier data point is given in the table below for the case when $P\left(\boldsymbol{d}_i \; is \; inlier \middle| e\left(\boldsymbol{d}_i, \boldsymbol{\mathcal{M}}(\boldsymbol{\theta})\right) < \delta\right) = 0.99$.

σ	0.1	0.2	0.3	0.4	0.5	0.6
δ	0.3368	0.6737	1.0105	1.3474	1.6842	2.0211

Table 10. Relation between specified noise level and distance threshold. Data points that are at a distance of less than δ from a fitted plane are marked as inliers.

Other available parameters in the RANSAC Toolbox were set to their default values. Due to the random sample selection nature of RANSAC, five tests were conducted for each parameter setup, and the average errors were saved (as the error percentage fluctuates from one run to another, there would otherwise be a risk of drawing the wrong conclusions and not selecting the most appropriate parameters).

Results for the Tsukuba image pair

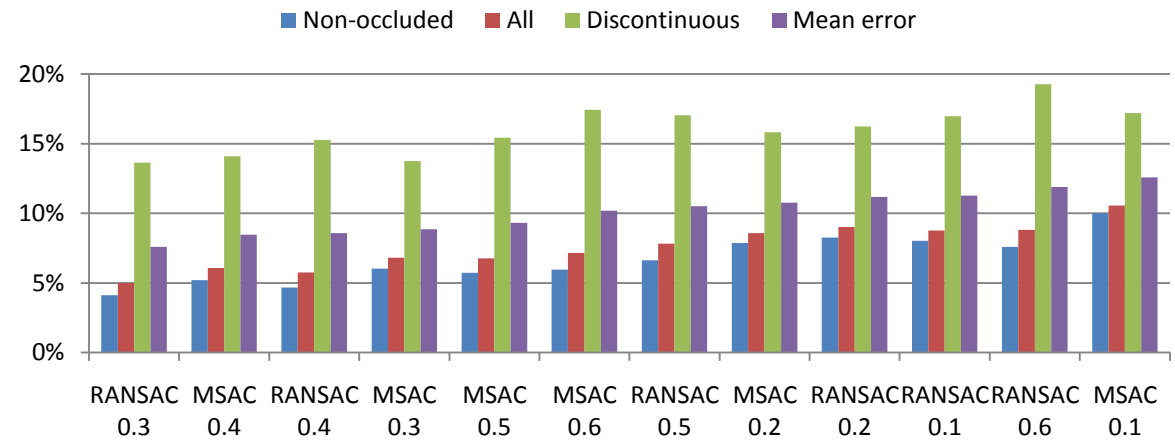


Figure 73. Plane fitting results for Tsukuba, sorted by percentage of 'all' bad pixels.

Results for the Venus image pair

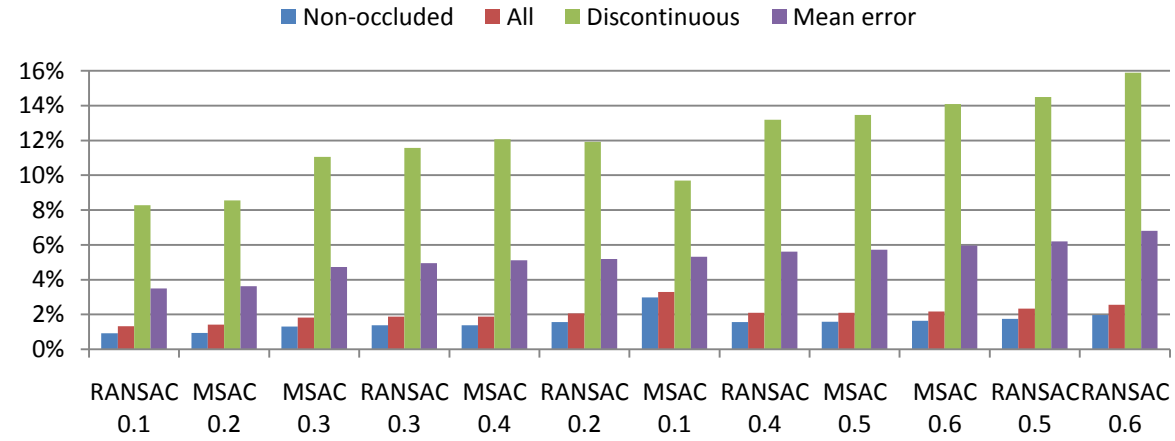


Figure 74. Plane fitting results for Venus, sorted by percentage of 'all' bad pixels.

Results for the Teddy image pair

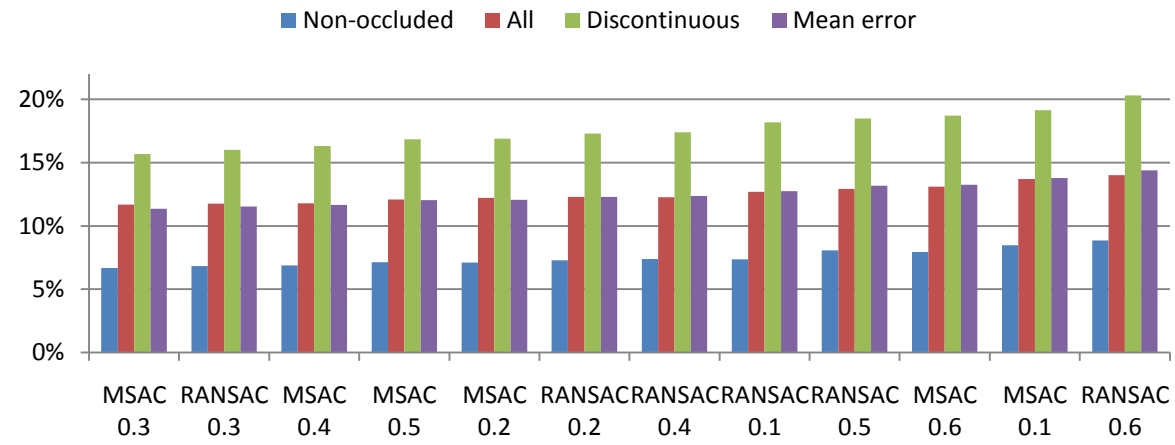


Figure 75. Plane fitting results for Teddy, sorted by percentage of 'all' bad pixels.

Results for the Cones image pair

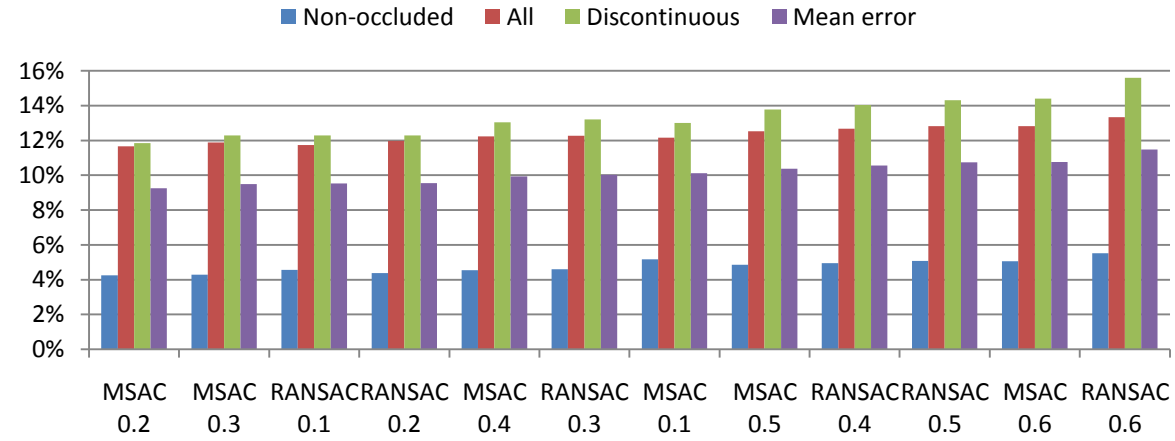


Figure 76. Plane fitting results for Cones, sorted by percentage of 'all' bad pixels.

Other window sizes

Experiments were also run with a window size of 7x7 to see the effect. Generally, the error goes up when using window sizes larger than 3x3 pixels. This is reasonably explained by smoothing.

Other window sizes

Experiments were also run with a window size of $7x7$ pixels. Generally, the error goes up when using window sizes larger than $3x3$ pixels. This is reasonably explained by the smoothing of the cost function that take place, which gives less accurate results at discontinuous regions.

Appendix C. Local matching implemented as suggested in CoopRegion

Image segmentation was performed with both bandwidths h_s and h_r set to 11. The minimum amount of pixels per segment M was set to 40.

A pixel matching cost where intensity differences and gradient differences were weighted equally was used. Also, the dissimilarity measure of BT was used instead of absolute differences.

Like in the paper [40], the window size was 51 pixels and the center segment pixels were weighted 100 times higher than pixels not belonging to the center segment. Moreover, *Geman-McLure* truncation of the cost was used with $\sigma = 3$ prior to aggregation, meaning that the cost was truncated according to $C_{TRUNCATED}(x, y, d) = \frac{C(x,y,d)^2}{\sigma^2 + C(x,y,d)^2}$.

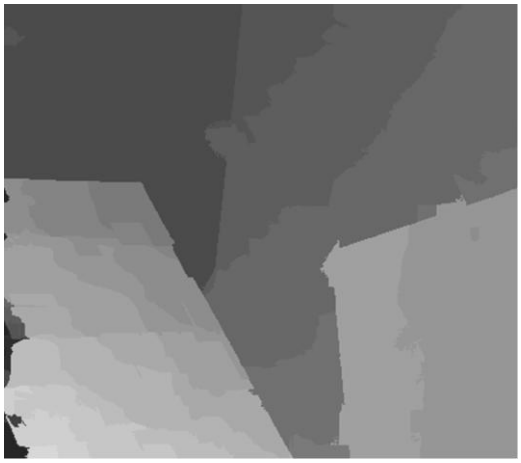
C.1 Result for Middlebury stereo images

See Section 4.2 for a definition of the bad pixel percentages $\mathcal{N}$, $\mathcal{A}$ and $\mathcal{D}$.

Tsukuba

$\mathcal{N}$	$\mathcal{A}$	$\mathcal{D}$
2.30%	2.52%	9.92%

Venus



$\mathcal{N}$	$\mathcal{A}$	$\mathcal{D}$
1.02%	1.46%	6.82%

Cones

$\mathcal{N}$	$\mathcal{A}$	$\mathcal{D}$
7.38%	*14.53%*	*15.27%*

Teddy

$\mathcal{N}$	$\mathcal{A}$	$\mathcal{D}$
11.64%	*17.27%*	*25.18%*

C.2 Pantomime

The result for a frame of the *Pantomime* image sequence (see Section 0) is given in Figure 77 below.

Figure 77. The result for Pantomime from using segment based local matching.