Fakultät für Physik und Astronomie

Ruprecht-Karls-Universität Heidelberg

Masterarbeit

Im Studiengang Physik

vorgelegt von

Jacob Nieswand

geboren in Konstanz

2018

Optimierung von Lichtfeldmessungen mithilfe des Strukturtensors

Die Masterarbeit wurde von Jacob Nieswand

ausgeführt am

Institut für wissenschaftliches Rechnen (IWR)

unter der Betreuung von

Herrn Priv.-Doz. Christoph Garbe

Department of Physics and Astronomy

University of Heidelberg

Master thesis

in Physics

submitted by

Jacob Nieswand

born in Konstanz

2018

Optimizations of light field measurements using the structure tensor

This Master thesis has been carried out by Jacob Nieswand

at the

Institut für wissenschaftliches Rechnen (IWR)

under the supervision of

Herrn Priv.-Doz. Christoph Garbe

Optimierung von Lichtfeldmessungen mithilfe des Strukturtensors:

Diese Arbeit befasst sich mit der Tiefenrekonstruktion einer Szene mittels 2D-Aufnahmen aus verschiedenen Blickwinkeln (Lichtfeld). Aus der Struktur der epipolaren Schnitte durch das Lichtfeld (EPIs) wird mithilfe des Strukturtensors die Tiefe errechnet. In dieser Arbeit werden verschiedene Modifikationen bestehender Algorithmen getestet, weiterhin werden Depth-of-defocus-Techniken getestet und verglichen. Zusätzlich wird ein Algorithmus basierend auf dem in der Stereo-Vision verbreiteten Semi-global Matching Algorithmus (SGM) als Nachbearbeitungsschritt implementiert.

Optimizations of light field measurements using the structure tensor:

This work deals with 3D depth reconstruction of a scene using 2D images captured from different angles (light field imaging). From the structure of the epipolar plane images (EPIs) the depth is calculated using the structure tensor. In this thesis, various modifications of existing algorithms are tested, further depth-of-defocus techniques will be tested and compared. In addition, an algorithm based on the semi-global matching algorithm (SGM), commonly used in stereo vision, is implemented as a post-processing step.

Contents

1 Introduction **8**

2 Theory **10**

2.1 Light field parametrization 10
2.1.1 The Lumigraph 10
2.1.2 Epipolar plane images 11
2.2 Depth from structure tensor 12
2.2.1 Implementation 13
2.2.2 The occlusion problem of the structure tensor 15
2.3 Modifications on the structure tensor 16
2.3.1 Kernel sizes 16
2.3.2 Colour-awareness with bilateral filtering 18
2.3.3 Thresholding the gradients 19
2.3.4 Occlusion-awareness using segmentation of the EPI 19
2.3.5 An alternative to the coherence as the confidence measure . . 22
2.3.6 Alternative merging of x- and y- orientation 24
2.4 Depth from focus 25
2.5 Semi-global matching 27
2.5.1 Semi-global matching for stereo vision 27
2.5.2 Semi-global matching for light fields 28
2.5.3 Occlusion awareness in SGM for light fields 30
2.5.4 SGM as post-processing smoothing 31
2.5.5 Iterative post-processing 32
2.5.6 A new penalty function 33
2.5.7 Semi-global matching implementation 34
2.6 Metrics for the evaluation of depth maps 35
2.6.1 Dense and sparse depth maps 35
2.6.2 Mean squared error 35
2.6.3 BadPixel score 36
2.6.4 Bumpiness 36
2.6.5 Discontinuity and planar scores 36
2.6.6 Mean relative error 36

3 Evaluation **37**

3.1 Depth from focus 37
3.1.1 Using depth-from-refocus as a pre-estimate for the ST 40

3.2 Modifications on the structure tensor 42
3.2.1 Sandglass kernel 44
3.2.2 Bilateral filter 44
3.2.3 Thresholding gradients 48
3.2.4 Occlusion awareness segmentation 49
3.2.5 Alternative coherence 52
3.2.6 Alternative merging of x- and y-direction 54
3.3 Semi-global matching 56
3.3.1 In-process SGM 56
3.3.2 Improved coherence measure by using SGM 58
3.3.3 SGM as pure post-processing 59
3.3.4 Iterative SGM post-processing 62
3.3.5 Application on real data 65
3.4 Comparison of results 68

4 Resumee and outlook **70**

A Derivation of the structure tensor **73**

B Alternative Coherence: Visualizing the quality of a mask **74**

C SGM parameter testing **76**

D Real Data **80**

E CNN disparity estimation implementation **81**

F Lists **84**
F.1 List of Figures 84
F.2 List of Tables 86

G Bibliography **87**

1 Introduction

In the field of computer vision, one big challenge is the 3D reconstruction based on 2D images. When reconstructing the depth based on two images capturing the scene from two points in space, we speak of *stereo vision*. If a specific feature point in one image can be detected in the other image as well, the distance of the feature point to the cameras can be calculated via triangulation.
The necessity for depth imaging is reasoned by the sheer amount of different applications, reaching from industrial quality management to the creation of CGI effects as well as applications in the gaming industry. 3D reconstruction recently has become irreplaceable in the car industry related to the development of autonomous driving systems. Another industrial application is the robotic vision for improving automation setups, such as vision-controlled pick-and-place systems.

Obtaining 3D Information via a camera system is mainly done by using either stereo camera systems, structured light systems, TOF(Time-of-flight)-cameras or interferometry cameras. While the last 3 mentioned systems are capable of reconstructing 3D-scenes of diffuse surfaces quite well, they do not make use of the surface RGB values.
Recently *light field imaging* as an expansion of stereo imaging has become a main field of research in computer vision. By capturing the scene from multiple viewpoints, a whole bunch of data can be processed to not only reconstruct the depth more robustly but also enable making statements about the surface characteristics. The real-time processing of a sufficient amount of 2D data still remains a challenge. Further, reconstructing the 3D image from 2D images comes with challenging tasks:

- Occlusions: As we cannot see the full scene from each viewpoint, some points of the surface we are looking at are either partly or fully occluded.

- Single-coloured surfaces: Parts of the scene with low textural structure make viewpoint-to-viewpoint-matching barely possible.

- Glossy surfaces: When developing a 3D reconstruction solution based on the assumption of *Lambertian* surfaces, we assume a surface point to be in the same colour no matter from which position/angle we look at it. However, this is not the case for reflective/glossy surfaces.

In this work, we try to tackle those problems using a depth reconstruction algorithm based on the work of Wanner [2014] while maintaining reasonable calculation times. Advanced research on light field depth reconstruction has been published by Diebold

[2016], expanding the approach to heterogeneous light fields. Schönpflug [2017] conducts research at inpainting low-feature areas in light field depth maps. In the work of Freist [2018], modifications on the algorithm of Wanner as well as a graph-based post-processing optimization method are proposed for obtaining improved depth maps.

2 Theory

2.1 Light field parametrization

The earliest introduction to light fields in literature can be found in Adelson et al. [1991], where they parametrize the field of light as a so-called 'plenoptic function'. If we assume that every point in space emits a light ray in a given direction which is characterized by an intensity value P, the whole information in the light field is given as a 6-dimensional function

$$P(V_x, V_y, V_z, \theta, \phi, \lambda), \tag{2.1}$$

where θ and ϕ are the solid angles describing the direction of any light field, λ describes the wavelength dependence. $V_{\{x,y,z\}}$ describe the room coordinates. If we use the pixels of a pinhole camera image as the coordinate system of our choice, the plenoptic function would be parametrized as

$$P(x, y, V_x, V_y, V_z, \lambda). \tag{2.2}$$

A more generalized model of the plenoptic function could also include a time dependence, leading to a 7-dimensional ray space. In general, the plenoptic function serves as the global function that gets mapped into a low-dimensional space in some form by any camera device, e.g. a pinhole camera mapping the whole ray space down to a 2-dimensional image.

A more detailed introduction to light fields can also be found in [27].

2.1.1 The Lumigraph

In the form of equation 2.2 the plenoptic function is a 7-dimensional function, which is difficult to record and to handle. As described by Wu et al. [2017] this plenoptic function is usually simplified in 3 steps: First, we neglect the time dependence assuming a static scene. Further, we neglect the wavelength λ. Instead, we define the mapped value of the plenoptic function as a vector containing the colour channels. Additionally, the plenoptic function obtains redundant information because light rays, lying on one line in space, propagate in the same direction, as introduced by Bolles et al. [1987]. The 4D-representation is also known as the *Lumigraph*. Those 4 dimensions can be split to 2 angular dimensions describing the direction of each ray and 2 dimensions for the location of the ray: Since every ray would pass an image plane of infinite size once (if the propagation vector is not parallel), two coordinates

for localizing the ray are sufficient. Commonly the *two-plane-parametrisation* is used to describe the light field. Every ray is characterized by the intersection of two arbitrary parallel planes Π and Ω. Mathematically spoken the light field L is a function

$$L : \Omega \times \Pi \to R \qquad x, y, s, t \to L(x, y, s, t), \tag{2.3}$$

where x, y are the coordinates in the first plane Ω and s, t are the coordinates in the second plane Π. If we move a camera in a plane and take pictures of a scene orthogonal to the camera plane, the image itself is described by the image coordinates x, y while the position of the camera is denoted as s, t. Both planes are parallel to each other; that way a light field can be measured in a straightforward manner.

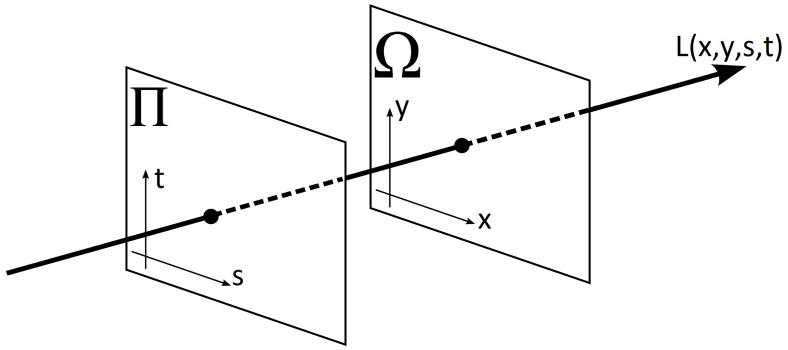

Figure 2.1: In a 4-dimensional two-plane parametrisation a light ray is characterized by the intersection with two parallel planes. We refer to the plane Π as the camera plane and the plane Ω as the image plane. From [32]

2.1.2 Epipolar plane images

We measure the light field in order to obtain as much information about the scene we are looking at as possible. In order to obtain the 3-dimensional structure one needs to map the light field in a certain manner: We take a look at one 2-dimensional slice through the 4-dimensional space while keeping 2 coordinates constant: one horizontal coordinate x^* in the image plane and one vertical coordinate t^* in the camera plane (or vice versa). The slice Σ_{x^*,t^*} is a 2-dimensional image called *epipolar plane image* (EPI). In figure 2.2 the extraction of an EPI from camera array data is visualized. As seen in figure 2.3 it consists of lines with different slopes, each point on one line with the same slope belongs to the same point in the scene viewed from different angles. From the slope Δ we obtain the distance from the camera plane with the equation

$$\text{distance} = \frac{f \cdot b}{\Delta}, \tag{2.4}$$

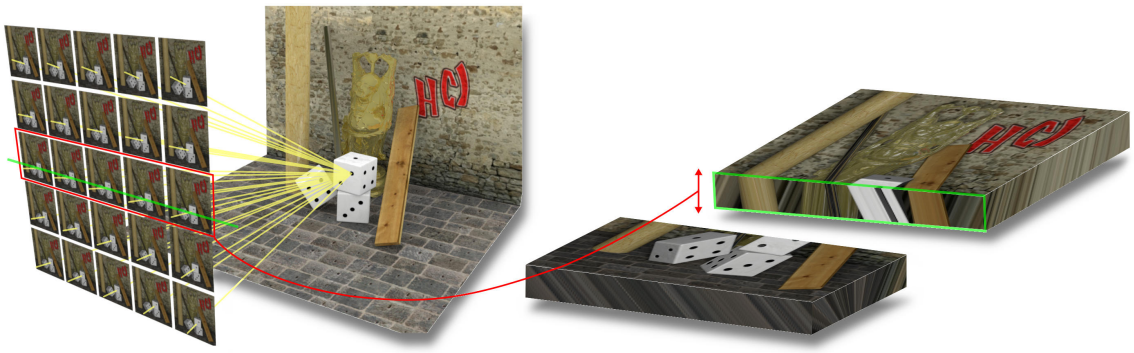

Figure 2.2: Visualization of an epipolar plane image extraction: A camera array takes images of the same scene from slightly different angles (left array). For a fixed image coordinate y^* (green) and a fixed camera coordinate t^* (red) the pixels are extracted and stacked up resulting in an EPI Σ_{y^*,t^*} (green box on the right). From [3]

where f is the focal length of the camera and b is the baseline between the camera positions. One may notice that this equation looks equal to distance estimation in stereo vision, where we replace the slope Δ by the disparity shift of a feature in 2 views. In fact, a stereo light field capture would result in an EPI with only two pixel rows, the slope of a line consisting of only two points is then defined as the disparity shift between the views.

Figure 2.3: An epipolar plane image (EPI) that consists of 9 rows (9 equidistant views or sample points in the camera plane). Points with the same colour correspond to the same scene point. Since the viewpoints are slightly shifted, the scene point is also shifted in each view by the disparity d. The pixels from the centre view position are marked in red.

2.2 Depth from structure tensor

Measuring depth from light field data can be done using various approaches. Wu et al. [2017] divide light field depth estimation approaches in three categories:

1. Sub-aperture image matching-based methods
2. EPI-based methods
3. Learning-based methods

In the following, we focus on one specific EPI-based method on which is investigated as part of this work. However, the reader is encouraged to take a look at the cited publication 'Light Field Image Processing: An Overview' by Wu et al. [2017] for a state-of-the-art overview of different light field approaches.
Wanner [2014] makes use of the oriented structure of the EPI using image-processing techniques. This idea has first been introduced by Bigun [1987] in 1987.

The 2-dimensional structure tensor J on a function $g : \Omega \to R, \Omega \subset R^2$ is defined as

$$J = \begin{pmatrix} G * \frac{\partial g}{\partial x}\frac{\partial g}{\partial x} & G * \frac{\partial g}{\partial x}\frac{\partial g}{\partial s} \\ G * \frac{\partial g}{\partial s}\frac{\partial g}{\partial x} & G * \frac{\partial g}{\partial s}\frac{\partial g}{\partial s} \end{pmatrix}, \tag{2.5}$$

where G is a Gaussian window function. The derivation can be found in the appendix. The first eigenvector of this tensor J indicates the preferred orientation in the local neighbourhood (defined by the window function). The second eigenvector is orthogonal to the first.

Jähne [2013] proposes a coherence value as a measurement for the anisotropy of the local environment:

$$C = \frac{\sqrt{(J_{11} - J_{22})^2 + 4J_{12}^2}}{J_{11} + J_{22}} \tag{2.6}$$

which obeys the value 1 in case of complete anisotropy, a value of 0 would indicate total isotropy.
In case of an EPI, the structure tensor provides an estimate for the slope of an EPI line, as defined by Bigun [1987]:

$$\Delta = \tan\left(\frac{1}{2}\arctan\left(\frac{J_{22} - J_{11}}{2J_{12}}\right)\right) \tag{2.7}$$

we refer to Δ as the disparity d in the following. Via equation 2.4 one obtains the depth in meters.

2.2.1 Implementation

The following implementation steps have been proposed by Wanner [2014]. The code uses *crosshair* light field data that can be obtained e.g. by a camera as depicted in figure 2.4. Instead of a full 2D camera array, the scope of the light field in the camera coordinates is significantly reduced. In the following, the horizontal camera row and the vertical camera column are viewed separately as 1-dimensional light-field cameras.

From the input images one extracts the EPIs, which have the dimensions

$$\begin{aligned} \text{vertical view} &\rightarrowtail \text{Nr of rows in image coordinates} \times \text{Nr of cameras} \\ \text{horizontal view} &\rightarrowtail \text{Nr of columns in image coordinates} \times \text{Nr of cameras} \end{aligned}$$

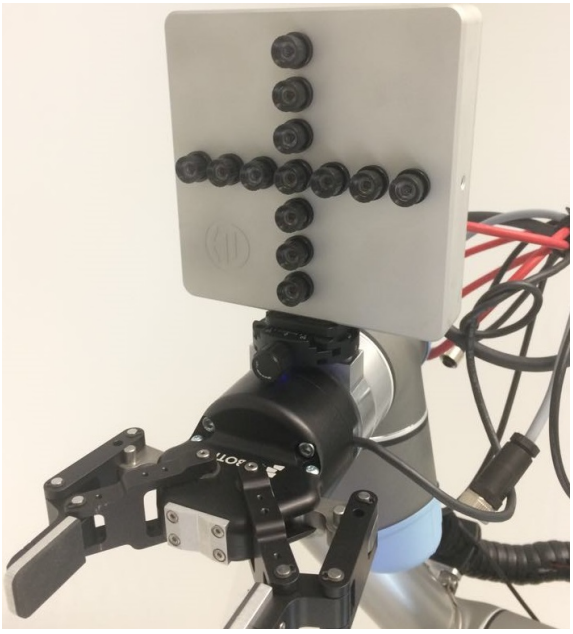

Figure 2.4: Instead of a complete camera array a crosshair camera reduces the number of cameras to one row and one column for faster processing. This *Lumi+* - Scanner from HDVision Systems is mounted on a robot arm for industrial applications.

For each EPI we calculate the structure tensor independently. This is done by first pre-smoothing the EPI with a 3×3 Gaussian kernel to avoid noise. Note that before calculating the gradient, the EPI is converted to grey-scale format.
The gradient is calculated via the Scharr-filter, which has the form

$$\text{Scharr}_x = \begin{pmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{pmatrix} \tag{2.8}$$

$$\text{Scharr}_y = \begin{pmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{pmatrix}. \tag{2.9}$$

Wanner [2014] and Diebold [2016] both tested out different gradient filters and came to the conclusion that the Scharr-filter performs best. With the gradients the structure tensor and the disparity is obtained via equation 2.5 and 2.7.
Further, Wanner found out that the structure tensor results are most accurate when the disparity is close to zero. Therefore the EPI is artificially refocussed by a simple linear shifting of the rows such that the slope of any line in the EPI would be increased by the same amount. The disparity shift can simply be subtracted from the disparity later. The refocussing of the EPI is illustrated in 2.5. For each EPI the structure tensor is calculated at each disparity shift so that the necessary disparity range is fully covered.

The value with the highest coherence measure (equation 2.6) is selected from all disparity shifts. Hence one obtains two depth estimations for each image coordinate in the centre view: one from the vertical EPI, one from the horizontal EPI. The merging of both direction follows

$$d(x,y) = \begin{cases} d_{\text{horizontal}}(x,y) & C_{\text{horizontal}}(x,y) > C_{\text{vertical}}(x,y) \\ d_{\text{vertical}}(x,y) & C_{\text{horizontal}}(x,y) < C_{\text{vertical}}(x,y) \end{cases}. \tag{2.10}$$

This implementation is referred to as the structure tensor (ST) pipeline in the following.

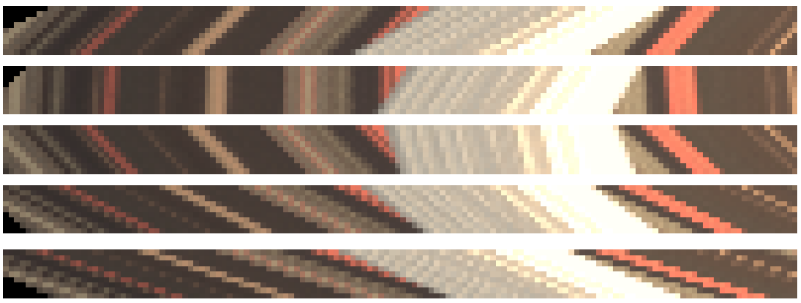

Figure 2.5: The EPI is refocussed by integer disparity steps. If the slope at a scene point is zero (vertical line) the EPI is perfectly focussed on that point. An integration of all views would still result in a sharp image at the corresponding depth.

2.2.2 The occlusion problem of the structure tensor

Having a look at the results of the benchmark test of Honauer et al. [2016], one realizes that most light field depth estimation algorithms suffer from large errors near depth discontinuities. Since the angular pixels close to the edge of a depth discontinuity are at least partly occluded, a loss of accuracy near discontinuities is expected. The ST almost always produces a systematic error near discontinuities, leading to a 'magnification' of the object closer to the camera in the depth map, see figure 2.7. We refer to this as 'edge fattening'. The reason for this error has its origin in the smoothing of the EPI as part of the algorithm. At the occlusion, the edge between the fore- and background follows the same orientation as the foreground does.
Furthermore, the occlusion mostly comes with a colour change, resulting in a high gradient which dominates the local environment orientation: Edge fattening is the consequence. In figure 2.6 one can recognize two effects leading to edge fattening: Firstly the EPI is smoothed with a 3×3 Gaussian kernel which already will enlarge the foreground structure (b). This becomes clear when we illustrate the norm of the gradients (c), since the strong (white) gradients all measure the foreground (green) structure. A clear shift to the left is visible from (b) to (c). However, this bias error

is small compared to the second error that is indicated in (d): The red dot lies clearly in the background structure. Calculating the structure tensor components from the local environment (green square), one would obtain the foreground structure at the red dot, since the strong gradients in the square dominate. In (e) the total bias in labelling the two structures is illustrated.

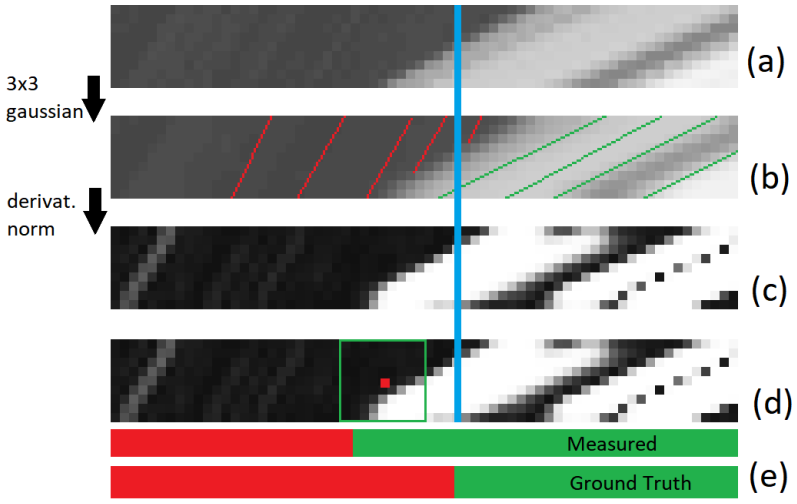

Figure 2.6: (a) The structure of an occlusion border in a grey-value EPI. (b) Before calculating the gradient, the EPI is smoothed with a 3×3 Gaussian kernel. One sees the smoothed EPI with coloured lines indicating the Ground Truth orientation. (c) shows the norm of the gradient calculated via the Scharr-filter. White represents a high gradient, black represents a low gradient. In (d) the local environment around the red dot (x_r, y_r) as an example point is marked to show which gradient values go into the structure tensor components $J(x_r, y_r)$. (e) marks the orientation labelling measured and ground truth. The blue line marks the transition in the centre view that corresponds with the ground truth labelling.

2.3 Modifications on the structure tensor

In the following, different methods will be explained which modify the structure tensor algorithm proposed by Wanner [2014]. In section 3 implementations of those modifications will be evaluated and discussed.

2.3.1 Kernel sizes

The structure tensor algorithm calculates the depth based on the preferred orientation of the EPI in a local neighbourhood. If one chooses a larger neighbourhood, the resulting depth map is likely to be smoother, while the accuracy at edges in the

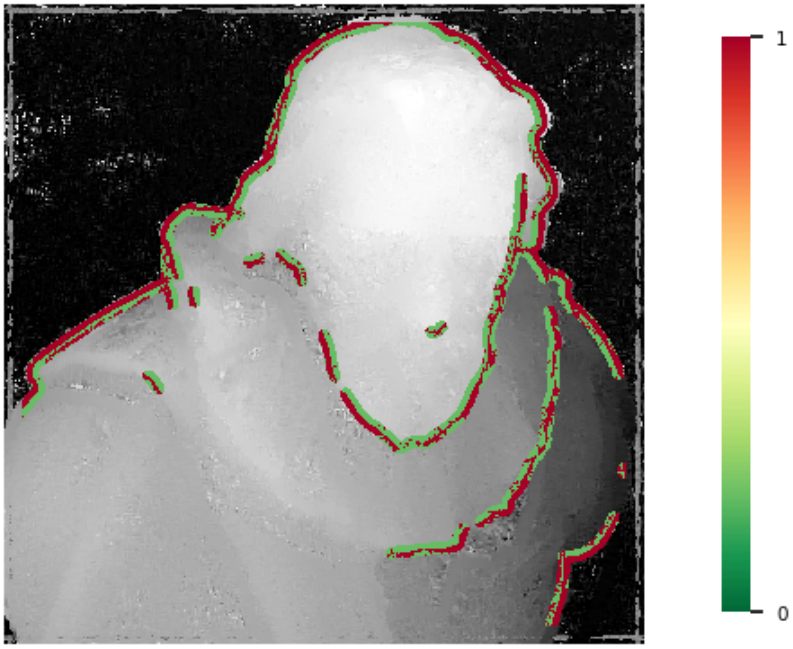

Figure 2.7: Evaluation of the deviation from ground truth at the depth discontinuity for scene *cotton*. In red/green the mask for calculating the deviation is depicted. If the disparity deviation is greater than 0.07 pix, it is coloured red.

scene rapidly decreases. Even though one often wants to smoothen the depth map in a post-processing step to reduce the noise and cancel out outliers, the structure tensor algorithm by Wanner is forced to smoothen the depth map in the process by estimating the depth from the local neighbourhood in the EPI. In the work of Wanner [2014] the effect of the kernel size has been examined extensively and he calculated the best parameters for different scenes by grid search. Note that if the kernel is smaller than the height of the EPI, the outer cameras are neglected and do not enter in the calculation of the depth. To maintain the information of all cameras and vary the kernel size at the same time, Diebold [2016] proposes asymmetrical kernel sizes to eventually reduce edge-fattening effects. However, asymmetrical kernels still need to find a trade-off between denoising and avoiding edge fattening and do not significantly increase the quality of the depth map result.
Another approach is to choose a custom kernel form which only weights the local gradients that could possibly be part of the 'correct' line in the EPI. Such a kernel is illustrated in figure 2.8, it has the form of a sandglass. In y-direction, it still follows a Gaussian distribution.

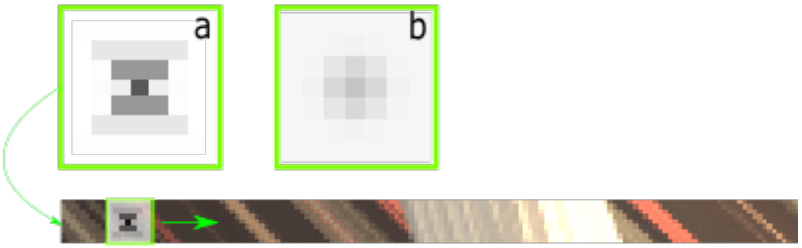

Figure 2.8: (a) shows a custom-shaped kernel in form of a sandglass. (b) shows a normal Gaussian kernel. The kernels are used for the calculation of the ST components along the EPI.

2.3.2 Colour-awareness with bilateral filtering

In its original form the structure tensor pipeline weights the square of all gradients in a local environment by the distance to the reference point. Since all local gradients in the EPI that belong to the same colour are very likely to belong to the same orientation, one has to find all points with the same colour which then lie on the same line in the EPI in case of a Lambertian surface. The preferred orientation of the gradients at those points will then correspond to their slope with high confidence. However, most objects are not perfectly Lambertian such that not all points on the correct line have to be of the exact same colour. Only filtering the exact colour will lead to noisy results. Instead, one could implement a filter weighting not only the distance in a Gaussian manner but also the distance from the EPI kernel origin in the colour RGB space; such a filter is called 'bilateral filter' (Tomasi and Manduchi [1998]). Pixels with completely different colour will be excluded from the neighbourhood when calculating the structure tensor. In equation 2.5 the window function G is then defined as

$$G_P = \frac{1}{W_p} \sum_{q \in N} g_{\sigma_d}(||p - q||) g_{\sigma_c}(||I_{p,\mathrm{EPI}} - I_{q,\mathrm{EPI}}||), \tag{2.11}$$

where

p is the position of the pixel,

q is the position of the pixel in a neighbourhood N,

g_{σ_d} is a Gaussian function weighting the distance between p and q using σ_d as standard deviation,

g_{σ_c} is a Gaussian function weighting the colour distance in RGB space between the RGB values of the original EPI at p and q using σ_c as standard deviation,

W_P is a normalization factor so that the sum over all the neighbourhood always is 1.

It is important to notice that the colour distance is extracted from the EPI, though we are filtering the components of the second derivatives, see equation 2.5. In fact, the EPI only serves as a guide for the filtering.

2.3.3 Thresholding the gradients

In section 2.2.2 it is explained why the structure tensor struggles to serve with a good estimation at depth transitions. It is referred to two errors; one coming from the first smoothing, the other one as a result from the second Gaussian smoothing. To avoid the second error, one could in principle normalize all local gradients such that in figure 2.6 (d) the foreground gradients in a local environment obey the same weight in the structure tensor calculation as the background gradients around the red dot.
The gradient components of the EPI would be given by

$$\tilde{\vec{\nabla}}(x,y) = \frac{1}{\sqrt{\nabla_x(x,y)^2 + \nabla_y(x,y)^2}} \vec{\nabla}_x(x,y) \tag{2.12}$$

Nevertheless, on some occasions one wants the stronger gradients to play a bigger role in the preferred orientation since small gradients are more likely to be noisy. One way to damp the gradients at occlusions while not strengthening the small noisy gradients in an EPI is to implement a *truncation threshold*. The gradient is then given by

$$\tilde{\vec{\nabla}}(x,y) = \begin{cases} \frac{\text{threshold}}{\text{norm}} \vec{\nabla}(x,y) & \text{if} \quad \text{norm} > \text{threshold} \\ \tilde{\vec{\nabla}}(x,y) & \text{else} \end{cases} \tag{2.13}$$

$$\text{with} \quad \text{norm} = \sqrt{\nabla_x(x,y)^2 + \nabla_y(x,y)^2}. \tag{2.14}$$

$$\tag{2.15}$$

The threshold should be chosen the lowest possible in a manner that gradients higher than the threshold are barely affected by noise. In section 3.2.3 different thresholds are tested, if not mentioned otherwise a threshold of 0.1 is used assuming that the underlying EPI is a grey-value image with range 0-1.

2.3.4 Occlusion-awareness using segmentation of the EPI

In this section, a method is proposed that attempts to handle both occlusion errors mentioned in section 2.2.2.
Bilateral filtering (section 2.3.2) and thresholding the gradients (section 2.3.3) may improve the depth map estimation, however they come with trade-offs in order to handle the occlusion problems. An alternative way to maintain sharp transitions is to find those transitions in the EPI and calculate them separately. By setting the

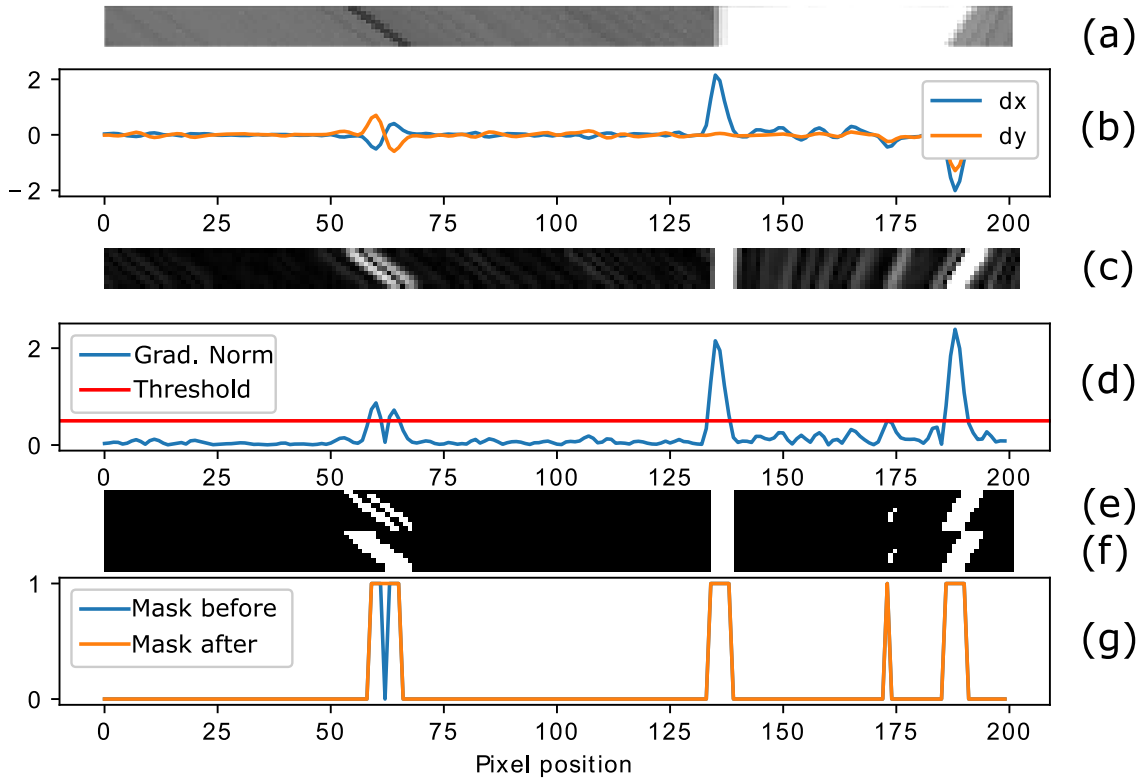

Figure 2.9: (a) Input EPI to be segmented. (b) shows the local derivatives along the centre view line. (c) shows the vector norm of the derivative. (d) shows the centre view norm with the threshold that is applied to mask transitions. (e) shows the resulting mask. (f) shows the improved mask using morphological image processing. (g) shows the centre view line from (e,f).

gradient at the transition in the EPI to 0, background and foreground structure are calculated without being biased by the gradient of the transition itself. Therefore the EPI is segmented binarily into occlusion transitions and continuous areas, whereby the norm vector is thresholded to create a binary mask on the EPI.
Possibly one could think of a better criterion for segmenting occlusion/non-occlusion areas than the norm vector size. However, using this binary mask there is no need to calculate the disparity values beforehand and it can be implemented in the workflow directly. In principle the algorithm works as follows:

1. Segment the transitions and the rest of the EPI with a binary mask based on the gradient norm.

2. Calculate the structure tensor components on the masked gradient of the EPI. All masked gradients are zero and do not affect the local environment structure.

3. Calculate the structure tensor components again, now on the inverted masked gradient of the EPI.

4. Join the disparity values from both masks.

In figure 2.9 the segmentation process is illustrated step-by-step. The gradient of the grey-value EPI (a) is calculated in x- and y-direction (b), from which one calculates and thresholds the norm (c, d). The threshold itself is chosen to be 0.5 if not mentioned otherwise. It is essential to modify the resulting mask from the binary thresholding (e) using morphological image-processing operators as explained in detail in [2] . We make use of morphological closing (structured in-filling of image region boundary pixels), which is explained in figure 2.10. Having a look at the

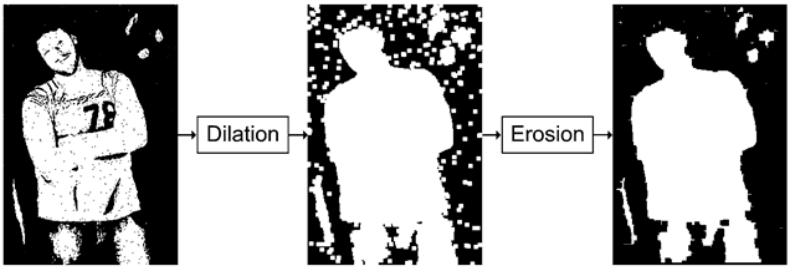

Figure 2.10: Morphological closing is a combination of *dilation* and *erosion*. Dilation applies a custom-sized binary kernel on a binary input file and turns all **0** to **1** as long as at least one **1** is found in the kernel. Erosion on the other hand turns all **1** to **0** in the kernel. Image from [4]

different scenarios in the given EPI that can happen when segmenting, the necessity for morphological closing becomes clear, see figure 2.9. The EPI has two visible disparity transitions, one at pixel [133] and one at pixel [185]. At [133] (a) it is seen that the transition affects multiple neighbouring pixels even though the EPI isn't smoothed yet. Our aim is to mask those transitions. However at Pixel [63] and[174] the threshold is also crossed, since big gradients do not necessarily have to lie on a transition. Still, the false-positive masked transitions are less uncomfortable to handle than a false negative result. It is clearly visible at [63], that zero crossings in the gradient lead to holes in the mask which need to be closed by morphological closing. Since the local environment of each pixel is only given by near pixels in the same segment (masked/unmasked), small holes lead to erroneous results at those pixels. After morphological closing, a last post-processing step has to be made to handle the smearing due to the 3×3-Gaussian pre-smoothing In figure 2.10 (d) one sees that the transition peak is Gaussian-formed. If one simply cuts out the segment at the threshold transition, the background segment gradient is still dominated by the gradients directly next to the segment transitions. This becomes visible in figure 2.11, where the inverted mask of the EPI gradient is shown. Thus, morphological dilation with a 3×3 kernel is applied on the inverted filter, resulting in a complete separation of any peak crossing the threshold.

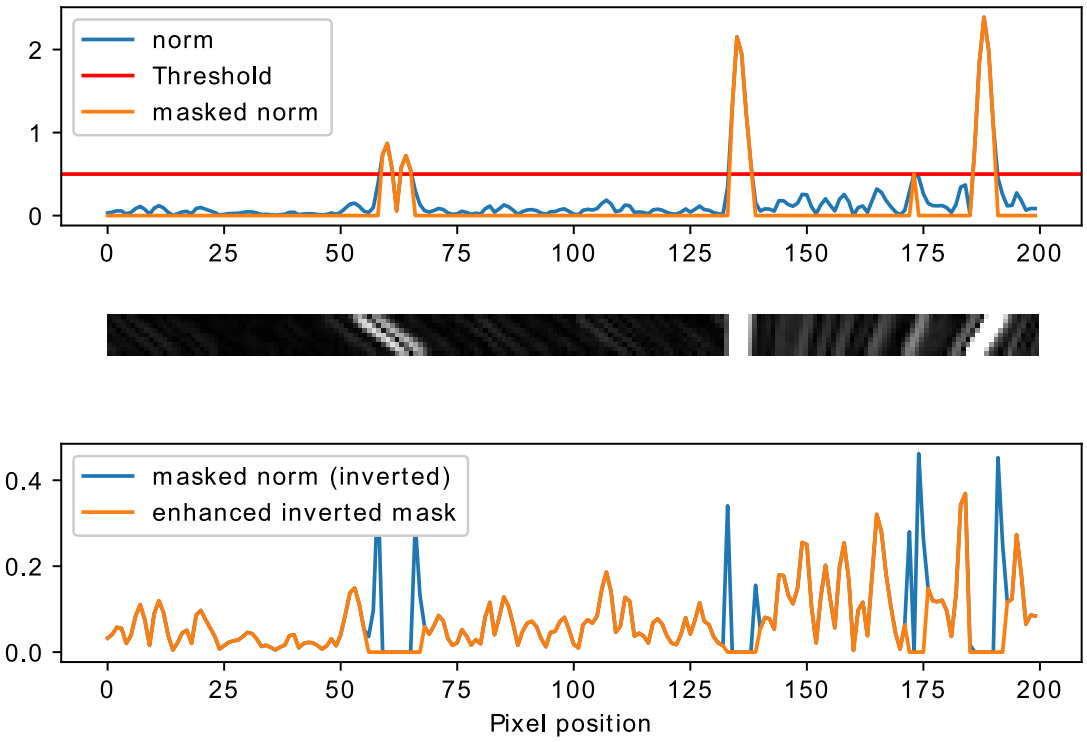

Figure 2.11: In the upper diagram the thresholded gradient norm of the EPI (middle) is seen. The norm (masked with the inverted mask) is depicted in the lower figure (blue), while the inverted mask with a dilation filter is depicted in orange.

2.3.5 An alternative to the coherence as the confidence measure

The coherence (2.6) describes how strong the anisotropy of a structure in an image is. In the structure tensor pipeline (section 2.4) this value is used as a confidence measure for the corresponding depth value. In this section, we try to define a more profound definition for the confidence based on the distribution of the gradients in a small environment. From the structure tensor components we obtain the disparity

$$d = \tan\left(\frac{1}{2}\arctan\left(\frac{J_{22} - J_{11}}{2J_{12}}\right)\right), \tag{2.16}$$

whose error Δd is then given by first order error propagation:

$$\Delta d = \sqrt{\left(\frac{\partial d}{J_{11}}\Delta J_{11}\right)^2 + \left(\frac{\partial d}{J_{22}}\Delta J_{22}\right)^2 + \left(\frac{\partial d}{J_{12}}\Delta J_{12}\right)^2} \tag{2.17}$$

When substituting

$$d = \tan\left(\frac{1}{2}\arctan(x)\right), \quad \text{with} \quad x = \frac{J_{22} - J_{11}}{2J_{12}} \tag{2.18}$$

we obtain

$$\frac{\partial d}{\partial x} = 0.5 \cdot \frac{d^2 + 1}{x^2 + 1}. \tag{2.19}$$

With

$$\frac{\partial x}{\partial J_{11}} = \frac{x}{J_{11} - J_{22}}, \quad \frac{\partial x}{\partial J_{22}} = \frac{-x}{J_{11} - J_{22}}, \quad \frac{\partial x}{\partial J_{12}} = \frac{x}{J_{12}} \tag{2.20}$$

inserting in 2.17:

$$\Delta d = \sqrt{\left(\frac{0.5 \cdot (d^2 + 1)}{x^2 + 1} \cdot x\right)^2 \cdot \left(\left|\frac{\Delta J_{12}}{J_{12}}\right|^2 + \left|\frac{\Delta J_{11}}{J_{11} - J_{22}}\right|^2 + \left|\frac{\Delta J_{22}}{J_{11} - J_{22}}\right|^2\right)} \tag{2.21}$$

The errors ΔJ_{11}, ΔJ_{12} and ΔJ_{22} can be obtained from the statistical deviations of the gradient:

$$\Delta J_{11} = G * |2\nabla_x|\Delta\nabla_x \tag{2.22}$$

$$\Delta J_{22} = G * |2\nabla_y|\Delta\nabla_y \tag{2.23}$$

$$\Delta J_{12} = G * \sqrt{\nabla_x^2\Delta\nabla_y^2 + \nabla_y^2\Delta\nabla_x^2} \tag{2.24}$$

Those equations are obtained from equation 2.5 using first order error propagation. We define $\Delta\nabla_x, \Delta\nabla_y$ as the statistical standard deviation of ∇_x, ∇_y in the neighbourhood and calculate it via

$$\Delta\nabla_x = \sqrt{E[\nabla_x^2] - E[\nabla_x]^2} \tag{2.25}$$

$$\Delta\nabla_y = \sqrt{E[\nabla_y^2] - E[\nabla_y]^2} \tag{2.26}$$

with the expectation value defined as the Gaussian weighted mean:

$$E[I] = G * I \tag{2.27}$$

However, equation 2.25 only makes sense under the assumption that the underlying distribution of the magnitude whose standard deviation will be computed is close to a normal distribution. This is in general not the case for the gradient components since deviations in the magnitude of the gradient should not affect on the disparity error estimation. Given that the gradients are normalized, a Gaussian distribution of the gradient around the mean normalized gradient is expected:

$$\Delta\nabla_{x,\text{normalized}} = \sqrt{E[\nabla_{x,\text{normalized}}^2] - E[\nabla_{x,\text{normalized}}]^2} \tag{2.28}$$

$$\Delta\nabla_{y,\text{normalized}} = \sqrt{E[\nabla_{y,\text{normalized}}^2] - E[\nabla_{y,\text{normalized}}]^2} \tag{2.29}$$

The interrelation between $\Delta\nabla_{y,\text{normalized}}$ and $\Delta\nabla_x$ is given by

$$\nabla_{x,\text{normalized}} = \frac{1}{\sqrt{\nabla_x^2 + \nabla_y^2}}\nabla_x \tag{2.30}$$

$$\Delta\nabla_{x,\text{normalized}} \approx \frac{1}{\sqrt{\nabla_x^2 + \nabla_y^2}}\Delta\nabla_x, \tag{2.31}$$

where we neglect the fact that the normalization itself is also erroneous. This simplifies and speeds up the calculation. Equation 2.30 leads to the expression

$$\Delta\nabla_x = \sqrt{\nabla_x^2 + \nabla_y^2} \cdot \sqrt{E[\nabla_{x,\text{normalized}}^2] - E[\nabla_{x,\text{normalized}}]^2} \tag{2.32}$$

$$\Delta\nabla_y = \sqrt{\nabla_x^2 + \nabla_y^2} \cdot \sqrt{E[\nabla_{y,\text{normalized}}^2] - E[\nabla_{y,\text{normalized}}]^2}, \tag{2.33}$$

which can be calculated from the EPI. If we insert the above equation 2.32 into 2.22 into 2.21, we obtain a deviation measure Δd for the disparity based on the gradient deviation in the EPI. At occlusion edges,we expect a bimodal distribution of the gradient, such that the deviation $\Delta\nabla_{x/y,\text{normalized}}$ should be large. Thus the new error measure should handle occlusion edges as well.

2.3.6 Alternative merging of x- and y- orientation

In the old ST pipeline, the coherence value as defined in equation 2.6 not only determines which depth value should be chosen from refocussing the EPI, it is also used to decide whether the local depth estimation is taken from the x-directed EPI or the y-direction. In the article of Sheng et al. [2018] a full light field is evaluated such that multiple EPIs are available for each point. To handle occlusions, the EPI with less intensity variance is chosen to calculate the depth. As we are working with a crosshair light field only providing two EPIs for each scene point, this is not a possible approach in our case. Choosing the one with higher coherence seems logical, however at occlusions we are mostly provided with a clear structure in the EPI leading to a high coherence value. As a result, the EPI including edges is chosen in the close neighbourhood of a discontinuity edge, leading to 'edge fattening'.
In his work, Wanner [2014] also proposes a TV-L minimization scheme to merge the disparity values resulting from x-directional and y-directional EPIs. Despite achieving good results, the improvement did not justify the calculation time of this optimization scheme. He thus sticks to choose the value with higher coherence.
Instead, we propose to choose the disparity value which relates to a larger depth and only chooses the closer disparity value, if the coherence difference is big enough. Therefore we introduce a threshold:

$$d(x,y) = \begin{cases} \min(d_x(x,y), d_y(x,y)) & \text{if } |c_x(x,y) - c_y(x,y)| < \text{threshold} \\ d_x(x,y) & \text{else, if } c_x(x,y) > c_y(x,y) \\ d_y(x,y) & \text{else} \end{cases} \tag{2.34}$$

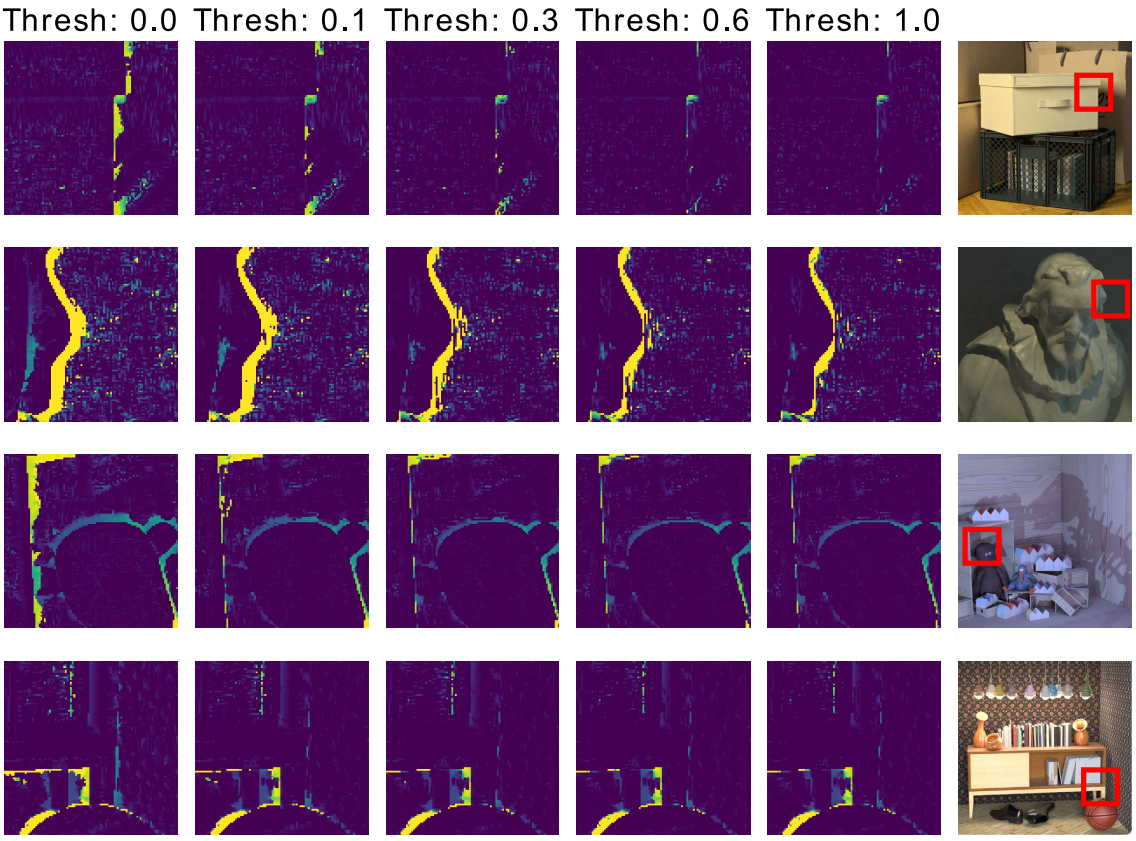

Figure 2.12: From left to right: Parameter dependent alternative merging. Up to down: scenes *boxes, cotton, dino, sideboard*. The deviation from ground truth is marked yellow, zero deviation is dark-purple.

Figure 2.12 shows the effect of varying the threshold introduced in equation 2.34 on the depth map by showing the deviation to the ground truth at occlusion discontinuities for different parameter values. The occlusion edges shrink significantly with a higher threshold. However, the effect is scene-dependent.

2.4 Depth from focus

One advantage of using light fields for depth measure is its ability to get a two-dimensional mapping of the scene with focus at any depth. Integrating the views of the light field camera array has the same effect as the integration of a focussed lens camera, as the lens is simply integrating slightly different viewpoints of the same scene point when focussed on the correct depth.
Obtaining the refocussed integrated image is a synthetic process that only requires shifting the view coordinates artificially. Given a full four-dimensional light field $L(u, v, x, y)$ we can refocus the light field as described in [22] :

$$L'(u, v, x, y) = L(u(1 - d'), v(1 - d'), x, y), \tag{2.35}$$

where d' describes the relative pixel shift. The disparity is directly related to the absolute depth of the focus if the relevant camera parameters are known. Given the baseline b in meters and the focal length f in pixels, the depth Z is given as

$$Z = \frac{f \cdot b}{d}. \tag{2.36}$$

We obtain

$$\bar{L}(x,y) = \frac{1}{N_{u,v}} \int\int L'(u,v,x,y) du dv = \frac{1}{N_{u,v}} \sum_u \sum_v L'(u,v,x,y) \tag{2.37}$$

Once we can focus at any range, one can adopt *depth-from-focus*-techniques as described in Watanabe and Nayar [1998] for depth measure. If the scene point at a given image coordinate (x, y) in the centre view is in focus, the contrast in the integrated image $\bar{L}(x, y)$ is high, thus a contrast measure at each pixel combined with stepwise refocussing yields a depth map.
For measuring the contrast, one has different options: The most straightforward approach is calculating the first derivative of the grey-value image. At high contrast structure, the local intensity changes are expected to be high. Alternatively one could measure the second derivative Laplacian that eventually results in higher robustness. The implementation and tests of those techniques for the benchmark dataset can be found in section 3.1.
Using a pinhole camera array allows us to go further and find a response value that shows higher consistency. Taking the absolute difference between the centre view of the camera array and the refocussed image yields promising results as shown in Tao et al. [2017]. Under the assumption of Lambertian surfaces, the RGB value of any scene point should be the same under all angles. Thus when refocussed on the correct depth, summing over all angles should result in a value that ideally is the same as in the centre view alone. This is referred to as *photo consistency*; for more information read Tao et al. [2017]. The response value at a given depth is obtained from

$$D'(x,y) = \frac{1}{|W_D|} \sum_{x',y' \in W_D} \left| \bar{L}(x',y') - P(x',y') \right|, \tag{2.38}$$

where $P(x, y)$ is the centre view. For more robustness, it is averaged over a small window. Note that calculating the norm results in a 1-channel-image while the input images are RGB-images.
Tao et al. propose another measure that they refer to as *angular correspondence*. It follows the same principle, but instead of integrating the refocussed light field followed by comparing it to the centre view, they directly take the difference of each viewpoint to the centre view and sum up those differences:

$$D'(x,y) = \frac{1}{N_{u,v}} \sum_u \sum_v \left| L'(u,v,x,y) - P(x,y) \right|. \tag{2.39}$$

We tested those methods against the common contrast measures mentioned above, the results are found in section 3.1.

2.5 Semi-global matching

2.5.1 Semi-global matching for stereo vision

In contrast to light field depth estimation techniques stereo systems more often suffer from mismatching pixels between the left and right images. Many attempts have been made to smoothen bad pixels, resulting in blurred edges or long calculation times. One promising attempt to improve matching results was published by Hirschmuller [2005]: we speak of semi-global matching (SGM). In general, matching

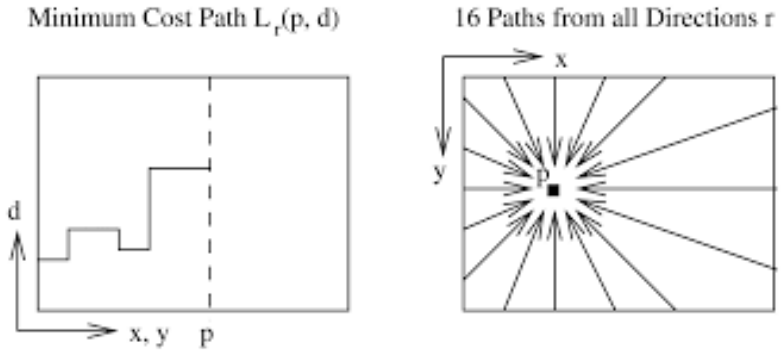

Figure 2.13: Left: Minimum cost path along an exemplary scanline. Right: each point gets passed by multiple paths from different directions. Image from [15]

of two stereo images means shifting the disparity over the predefined disparity range and comparing both images (pixel- or block-wise) until we have a cost value at each image point for each discrete disparity. We assign to each pixel $\vec{p}$ the disparity value $D_{\vec{p}}$ which is related to the lowest cost $C(\vec{p}, D_{\vec{p}})$. This matching does not have to be unique, resulting in erroneous pixel disparities. To overcome this one wants to minimize a global cost function of the form

$$E(D) = \sum_{\vec{p}} \left(C(\vec{p}, D_{\vec{p}}) + \sum_{q \in N_p} \begin{cases} P1 & \text{if } |D_{\vec{p}} - D_{\vec{q}}| = 1 \\ P2 & \text{if } |D_{\vec{p}} - D_{\vec{q}}| \geq 1 \\ 0 & \text{else} \end{cases} \right). \tag{2.40}$$

The fist term sums all matching costs over the whole image, while the second term forces continuity by comparing the disparity of all neighbour pixels N_q to the disparity D_p; if a small discontinuity is detected ($D_{\vec{p}} - D_{\vec{q}} = 1$), a small penalty is added to the global cost function. Since a small discontinuity can be found essentially at any tilted plane, only a small error is added. A bigger disparity difference indices a clear discontinuity in the disparity map. Note that the penalty $P2$ can be divided by the gradient $|\nabla|$ of the original image to allow a disparity discontinuity when we find edges in the image; at these points we expect the disparity to be discontinuous.

We obtain

$$P2'(x,y) = \frac{P2}{|\nabla|(x,y)}. \tag{2.41}$$

However, minimizing the global cost function involves computational cumbersome algorithms as it is an NP-complete Problem (Hirschmüller [2011]). Semi-global matching however chooses another approach by minimizing the global cost function along one-dimensional lines – this can indeed be calculated in polynomial time. The new smoothed cost function $S(\vec{p}, D_{\vec{p}})$ at pixel $\vec{p}$ is then given as the sum of all 1D minimum cost paths that are ending in $\vec{p}$. The minimal cost L'_r along the path r is defined recursively as

$$L'_r(\vec{p}, D) = C(\vec{p}, D) + \min \begin{cases} L'_r(\vec{p}_{\text{before}}, D) \\ L'_r(\vec{p}_{\text{before}}, D+1) + P1 \\ L'_r(\vec{p}_{\text{before}}, D-1) + P1 \\ \min_i L'_r(\vec{p}_{\text{before}}, i) + P2 \end{cases} \tag{2.42}$$

By always adding the minimum path cost of the previous pixel on the scanline we are looking at, we solve equation 2.40 in one dimension. Note that the rolling sum can reach quite high numbers that are unpleasant to handle on the computer; a normalization is implemented by subtracting $\min_D L'_r(p_{\vec{\text{before}}}, D)$ from all pixel cost values $L'_r(\vec{p}, D)$. The position of the minimum cost function at pixel $\vec{p}$ is unaffected by that normalization. In figure 2.13 an exemplary scanline is shown on the left. The point p is reached by 16 paths, whose cost functions $L'_r(\vec{p}, D)$ are cumulated (figure 2.13 right).

2.5.2 Semi-global matching for light fields

Even though Hirschmüller describes SGM as a complete algorithm to obtain a disparity map from a stereo image input, we further refer to SGM as the true novelty of his work: The implementation of an approximation to the global solution of the cost function (equation 2.40). Independent from the method one uses to calculate a disparity map, one needs a cost function defined in disparity space for each pixel to make use of SGM. Similar to the Stereo matching depth estimation, the structure tensor depth estimation pipeline for light field data sets produces a disparity map and a coherence value at each disparity shift. This implies that the SGM algorithm can be adapted to improve the results of the structure tensor pipeline. However, there are some significant differences between these two methods:

1. The structure tensor algorithm is tuned to a much smaller disparity range. While Hirschmuller [2005] scans a disparity range of 32 pixels, the benchmark data sets for light fields mostly include close-up views of objects, with a disparity range between 2 and 10 pixels. In figure 2.14 one can see the different values that are allocated in memory for each pixel of the image.

2. The sub-pixel accuracy using the structure tensor is a lot higher than the stereo matching sub-pixel accuracy. A simple adaptation of the algorithm to the structure tensor pipeline would require to give up the best feature that is provided by the ST, its sub-pixel accuracy.

Figure 2.14: SGM data that needs to be scored at each pixel in case of stereo (upper) and light field (lower). The cost values related to the 'correct' disparity are marked red.

To handle those problems, we do not throw away the sub-pixel accuracy. Instead, we use the float-value disparities to decide whether we penalize a disparity discontinuity or not. As one can see in figure 2.14, we have to process additional information, since the exact disparity value is not implicitly given by the index of the allocated cost value (in contrast to the original algorithm). Switching to a continuous space requires a new definition of the global cost function we defined in equation 2.40.

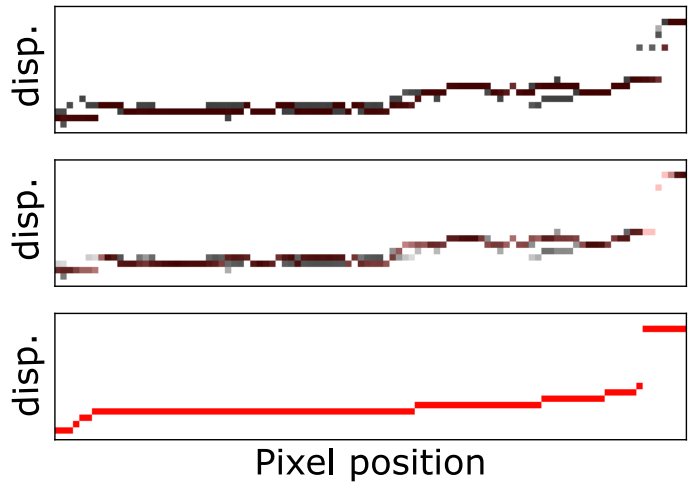

Figure 2.15: Upper: Measured disparity path (red) with underlying candidate values (grey). The intensity of the grey value reflects the cost value. Middle: disparity Paht after running SGM. Lower: Ground truth path

In the following, we refer to s as the disparity shift in the ST algorithm. Note

that we replaced D by d to clarify that the disparity is no longer discrete:

$$E(d) = \sum_{\vec{p}} \left(C(\vec{p}, d_{\vec{p}}) + \sum_{q \in N_p} \begin{cases} P1 \cdot |d_{\vec{p}} - d_{\vec{q}}| & \text{if } |d_{\vec{p}} - d_{\vec{q}}| \leq 1 \\ P2 & \text{if } |d_{\vec{p}} - d_{\vec{q}}| > 1 \end{cases} \right). \tag{2.43}$$

The recursive 1-d form to solve the global constraint on a scanline then changes to:

$$L'_r(\vec{p}, s) = C(\vec{p}, s) + \min_i \begin{cases} L'_r(\vec{p}_{\text{before}}, s_i) + P1 \cdot |d_{\vec{p},s} - d_{\vec{p}_{\text{before}},s_i}| & \text{if } |d_{\vec{p},s} - d_{\vec{p}_{\text{before}},s_i}| \leq 1 \\ L'_r(\vec{p}_{\text{before}}, s_i) + P2 & \text{if } |d_{\vec{p},s} - d_{\vec{p}_{\text{before}},s_i}| > 1 \end{cases} \tag{2.44}$$

The biggest difference lies in the fact that the small factor that is smoothing the image linearly increases with the distance. This change is necessary under the assumption that the disparity space is continuous. In other words we cluster disparity differences between two neighbouring points as either part of one surface ($|d_{\vec{p},s} - d_{\vec{p}_{\text{before}},s_i}| \leq 1$) that gets smoothed by the linearly increasing penalty, or assume a real disparity discontinuity that is penalized regardless of the size of the jump - the second error remains constant as it is in stereo matching. Note that $P2$ is modified as before (equation 2.41) by

$$P2'(x, y) = \frac{P2}{|\nabla|(x, y)}. \tag{2.45}$$

with $|\nabla|(x, y)$ being the centre view colour difference

$$|\nabla|(x, y) = \sqrt{\left(C_b(\vec{p}) - C_b(\vec{p}_{\text{before}})\right)^2 + \left(C_g(\vec{p}) - C_g(\vec{p}_{\text{before}})\right)^2 + \left(C_r(\vec{p}) - C_r(\vec{p}_{\text{before}})\right)^2} \tag{2.46}$$

$C_{\{\text{b,g,r}\}}$ are the blue, red and green colour channels of the centre view, respectively. The penalty for a disparity discontinuity is lowered if the colour intensity difference at this discontinuity is high. In figure 2.15 an exemplary scanline path through the scene *pens* is shown before and after using SGM. The position of the disparity candidates along the scanline are shown. While in the upper row we see a lot of ambiguities, in the lower row (after using SGM) a clear continuous path comes to light. Figure 2.16 shows that each path for itself leads to tearing effects in the given path direction. Only after merging all cost values from the different paths we obtain a smooth depth map while maintaining sharp edges.

2.5.3 Occlusion awareness in SGM for light fields

As discussed in section 3.2.4 the foreground structure is often dominating near disparity discontinuities. However, the structure tensor still finds the background structure near the boundary within a different SGM shift. With both disparity values

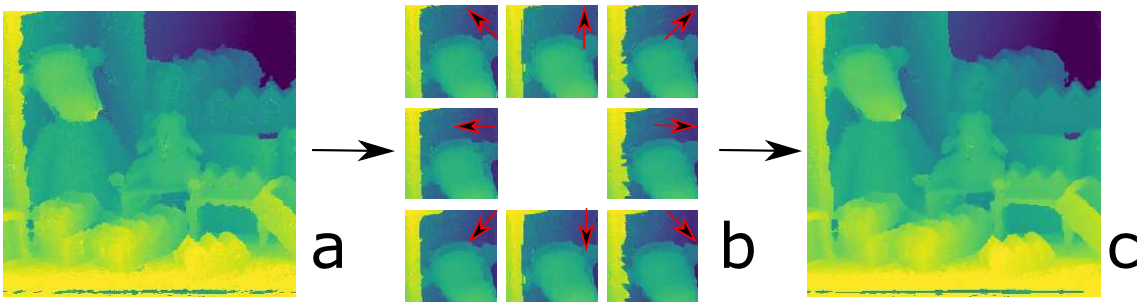

Figure 2.16: a) Depth map of scene *dino* using the old ST pipeline. b) Depth maps after using SGM only in one direction, respectively. c) Depth map after running SGM, using 8 paths.

stored at pixel p with different costs, the foreground is chosen more often.
The simple heuristic approach is to change the global minimization function 2.43 such that a positive disparity jump is less punished than a negative one. In fact, the function changes to

$$E(d) = \sum_{\vec{p}} \left(C(\vec{p}, d_{\vec{p}}) + \sum_{q \in N_p} \begin{cases} P1 \cdot |d_{\vec{p}} - d_{\vec{q}}| & \text{if } |d_{\vec{p}} - d_{\vec{q}}| \leq 1 \\ P2 & \text{if } d_{\vec{p}} - d_{\vec{q}} > 1 \\ P3 & \text{if } d_{\vec{p}} - d_{\vec{q}} < -1 \end{cases} \right). \tag{2.47}$$

2.5.4 SGM as post-processing smoothing

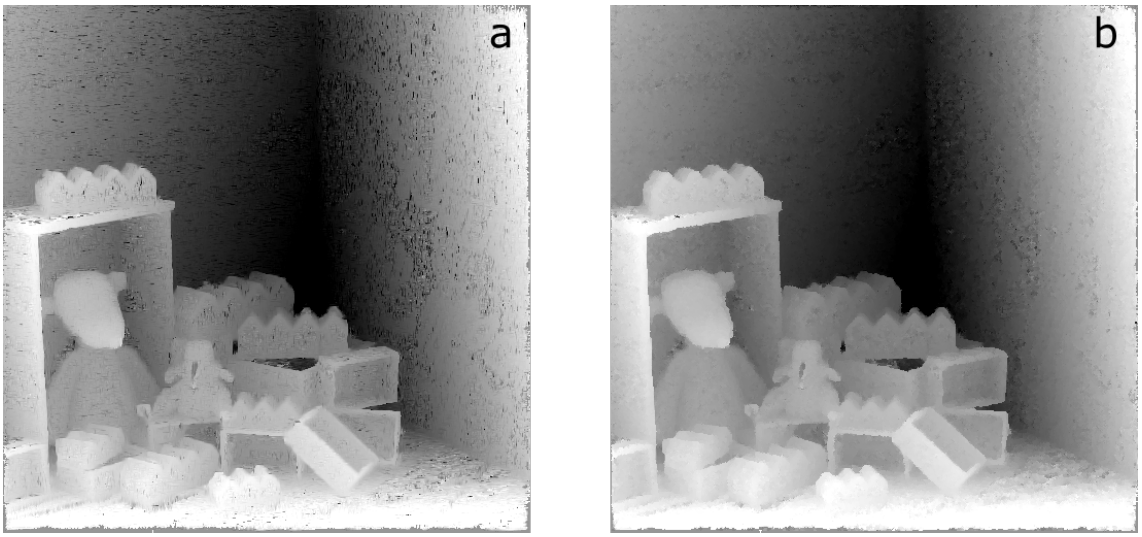

Figure 2.17: a) Erroneous disparity map before smoothing. b) Same disparity map after smoothing.

Minimizing the global energy function by solving it along different paths is possible as long as for each point a disparity candidate list with cost values is given. In

general, it is possible to extract those candidate values in a post-processing step by evaluating the close neighbourhood of a point in the disparity map. In the scope of this work, we propose to choose for each pixel 5 neighbouring pixels that are randomly chosen from the close neighbourhood after constructing the disparity map. In this sense, the SGM is used as a pure post-processing step that can be used to improve any depth map, regardless of the underlying algorithm for depth reconstruction. The effect of smoothing the disparity map while maintaining edges is depicted in figure 2.17 as an example.

2.5.5 Iterative post-processing

To reach the best result possible the SGM post-processing is applied multiple times. This iterative processing is inspired by the work of Freist [2018] that uses graph cutting to optimize the depth map. Instead of choosing random pixels from the close neighbourhood, the candidate values are found by adding/subtracting a Δ from the original pixel depth value d_i, such that the candidate values are given by the triple $(d_i; d_i - \Delta; d_i + \Delta)$. The candidate with the lowest cost is chosen and new candidates are formed for the next iteration. To reach convergence, Δ shrinks by a factor of two each iteration as long as the chosen candidate was the central one. Otherwise, Δ stays the same size. In figure 2.18 a) an exemplary disparity value is shown in dependence of the iteration step. The candidate window gets smaller each step, the stopping criterion is given by the overall iteration number. However,

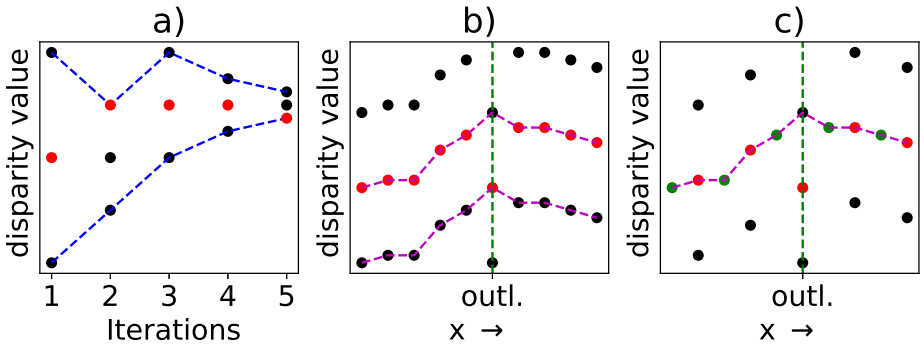

Figure 2.18: a) One exemplary pixel value in function of iteration steps. At each iteration, 3 candidate values for each pixel are chosen, the one with the lowest score is marked red. The candidate value window is marked with blue lines. b) One exemplary scanline through the depth map. Red marks the input values, the black points are the added candidate values. The green line marks the position of an outlier. The magenta-coloured lines show the paths with the lowest cost. c) same as the diagram b) with the difference of using a chess pattern initialisation.

setting the same Δ at each pixel leads to problems along any SGM scanline, as sketched in figure 2.18 b). Since all candidates are initialized with the same Δ one

finds 3 identical surfaces through the scanline that are shifted by Δ. Since they are almost identical, the cost of the disparity values lying either on one or the other surface is close to being identical. Hence the algorithm produces large errors along the scanline when choosing the wrong 'layer'. To overcome this problem one could apply a chess-pattern-like initialisation of the pixels. When proposing candidates to the SGM algorithm, every second pixel in the 2D-depth map is held constant to one value while the other pixels obtain 3 candidate values. After iterating, the second half of the pixels will be held constant. As depicted in figure 2.18 c) only one surface line will have a low cost value. Doing this iteration, though, doubles the amount of time the code consumes.

2.5.6 A new penalty function

The iterative pipeline does not assume that any of its candidates at one pixel position have to be close to the ground truth disparity, e.g. having an outlier d_i at pixel $i(x, y)$ does not necessarily mean that either $d_i - \Delta$ or $d_i + \Delta$ are close to the ground truth. With punishing those candidates as described in equation 2.43, all 3 candidates would obtain the same cost function value. However, one wants the candidate that is closest to its neighbours to be the one with the lowest cost. Thus, the cost function should be monotonically increasing as function of the distance to the previous disparity value d_{i-1}. A simple linearly increasing cost function would lead to overly smoothed edges. The proposed global cost function that is to be minimized then looks like

$$E(d) = \sum_{\vec{p}} \left(C(\vec{p}, D_{\vec{p}}) + \sum_{q \in N_p} P2 \cdot \left(1 - \exp\left(\frac{-|d_{\vec{p}} - d_{\vec{q}}|}{P1} \right) \right) \right). \tag{2.48}$$

Note that the initial cost can be neglected since we assume the same initial cost for all candidates such that we obtain the form

$$E(d) = \sum_{\vec{p}} \sum_{q \in N_p} P2 \cdot \left(1 - \exp\left(\frac{-|d_{\vec{p}} - d_{\vec{q}}|}{P1} \right) \right) \tag{2.49}$$

with the 1D recursive form

$$L'_r(\vec{p}, s) = \min_i \left(L'_r(\vec{p}_{\text{before}}, s_i) + P2 \cdot \left(1 - \exp\left(\frac{-|d_{\vec{p},s} - d_{\vec{p}_{\text{before}}, s_i}|}{P1} \right) \right) \right). \tag{2.50}$$

The equation 2.50 is plotted in figure 2.19 on the right side. In comparison to the old penalty function it has one less parameter, it is continuous and it always favours smaller disparity differences in order to get iterative processing to work. The error $P2$ is modified the same way as before, see equation 2.45. Large image gradients are more likely to lie on a disparity discontinuity, therefore the error is reduced. In figure 2.20 the effect of different penalty functions is depicted. Figure 2.20 a) shows

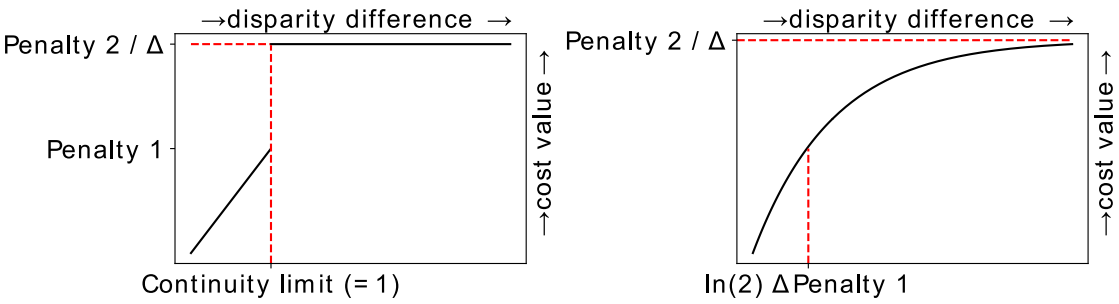

Figure 2.19: Sketch of different proposed Penalty functions. Left: Function as described in equation 2.44. Right function as described in 2.50.

the result of the scene *dino* after smoothing with the exponential penalty function 2.50, while figure 2.20 b) shows the disparity map after iterative SGM processing with a linear penalty function. One sees that the linear function leads to smoother surfaces, however the smoothness constraint also oversmoothens the edges in the image. In a) the edges are maintained sharp. The figure 2.20 c) shows the ground truth for comparison.

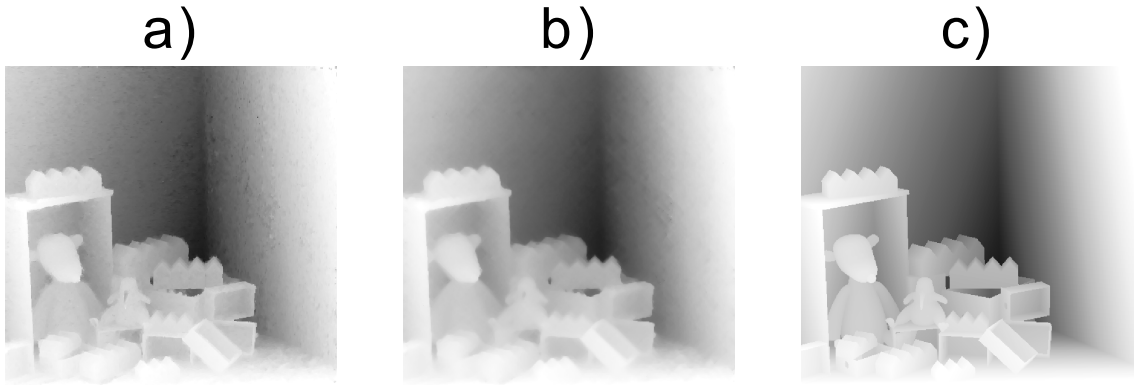

Figure 2.20: a) Depth map after iterative SGM smoothing using an exponential penalty function. b) Instead, a linear penalty function is used. c) Ground truth

2.5.7 Semi-global matching implementation

The implementation for using semi-global matching in our work is in line with the open source implementation published by Moratto [2013]. This code, however, uses only 8 instead of 16 paths along the image. Hirschmuller [2005] claims to reach computation times that are around 20 times faster than the implementation of Moratto with comparable hardware set-up (source: [21]), neither do we achieve fast computation times. Further, Ernst and Hirschmüller [2008] implement the SGM algorithm

on a GPU leading to an enormous increase of efficiency. Since all paths can be calculated in parallel, a GPU implementation can be implemented in a straightforward manner. The programming of the GPU implementation exceeds the boundaries of this work and will be part of future research and development. Instead, we focus on the feasibility of implementing SGM for light fields.

2.6 Metrics for the evaluation of depth maps

In the following, technical dataset details and general evaluation measures are presented. The datasets are provided by Honauer et al. [2016].

2.6.1 Dense and sparse depth maps

The synthetic light field datasets are designed with Blender [2014] and provide photos from the scene in a crosshair pattern, similar to the camera configuration in figure 2.4. In contrast to stereo camera setups, light field imaging enables a densely sampled reconstruction of the depth at each pixel. However, a full point cloud of a scanned object which is ideal for object detection applications will not be reconstructed due to occlusions. To evaluate the quality of the depth map, one could either try to optimize the overall error on the whole depth map (MSE) or count the number of pixels that are close to the Ground Truth with given tolerance Level (BadPixel score). From thereon one either tries to provide a sparse depth map masking out all the pixel where the confidence is low or the disparity map is provided on the whole pixel space. While sparse disparity maps are the choice for exact measurements in industrial applications, the second method facilitates comparisons of different methods. For this reason, the thesis mainly focusses on the metric score on the dense disparity estimation maps using different algorithms. Only for the application on real data one has to face large regions with low confidence, thus masking becomes relevant again.

2.6.2 Mean squared error

The Mean squared error, typically multiplied by a factor of 100, is defined as

$$\text{MSE} = \frac{\sum_x (d(x) - gt(x))^2}{\text{Nr of Pixels}} \cdot 100 \tag{2.51}$$

and describes the overall deviation from ground truth. $d(x)$ is the measured disparity value at pixel x, $gt(x)$ is the ground truth value at pixel x.

2.6.3 BadPixel score

The number of pixels at which the disparity value $d(x)$ deviates more than the tolerance t from the ground truth $gt(x)$ is given by

$$\text{BadPixel}(t) = \frac{\{|d(x) - gt(x)| > t\}}{\text{Nr of Pixels}}. \tag{2.52}$$

2.6.4 Bumpiness

The bumpiness score is quantified by the Frobenius norm of the Hessian matrix H_f of a function f which is defined here as $f = d - gt$. Smooth surfaces lead to a better bumpiness score regardless of possible missallignments and offsets:

$$\text{Bumpiness} = \frac{\min(0.05, ||H_f(x)||_F)}{\text{Nr of Pixels}} \cdot 100 \tag{2.53}$$

2.6.5 Discontinuity and planar scores

In some scenes, masks are applied to evaluate the score at different spots in the scene. This way it can be focussed quantitatively on challenging spots such as planar surfaces or discontinuities. Given a mask M the BadPixel score is then given as

$$\text{BadPixel}(t) = \frac{\{x \in M : |d(x) - gt(x)| > t\}}{\text{Nr of Pixels } x \in M}. \tag{2.54}$$

2.6.6 Mean relative error

In some cases we refer to the mean relative error (MRE) instead of the mean squared error. The difference lies in the normalisation:

$$MRE = \frac{\sum_x (d(x) - gt(x))^2}{(\text{max. disp - min. disp}) * \text{Nr of Pixels}} \tag{2.55}$$

3 Evaluation

In section 3.1, different depth-from-focus-techniques are tested and evaluated with regard to improving the ST depth estimation by pre-estimating the depth. Section 3.2 deals with the evaluation of different modifications on the ST pipeline. In the section 3.3, the feasibility of implementing SGM to light fields is evaluated. Lastly, all results are compared and discussed in section 3.4.

3.1 Depth from focus

The depth measure using epipolar plane analysis requires iterative calculation of the structure tensor for each EPI at each disparity. A way to overcome this is to generate a pre-estimate of the depth before actually calculating the correct depth. This could also help to prevent possible errors due to periodic scene characteristics which can lead to mismatch errors when calculating the structure tensor. Therefore the depth pre-estimate should fulfil the following criteria:

1. It should be *dense*, meaning that the number of pixels with low confidence should be the lowest possible.
2. It should result in a *fast* measure, ideally faster then it would take to do the full iterative structure tensor algorithm.
3. It does not have to be sub-pixel accurate, since it only serves as a pre-estimate.

The methods that are tested are described in section 2.4. We test four different ways to obtain a depth map using depth from focus:

Photo consistency This measure takes advantage of the fact that the difference between the refocussed two-dimensional image and the centre view is close to zero when refocussed to the correct depth. Response value:

$$D'(x,y) = \frac{1}{|W_D|} \sum_{x',y' \in W_D} \left| \bar{L}(x',y') - P(x',y') \right|, \tag{3.1}$$

Angular correspondence In contrast to the *photo consistency* measure, it first calculates the absolute difference between each camera array view and the centre view followed by the summation of those deviations. The response value is given as in equation (2.39)

$$D'(x,y) = \frac{1}{N_{u,v}} \sum_u \sum_v |L'(u,v,x,y) - P(x,y)| \tag{3.2}$$

First derivative The first derivative is calculated for contrast measure by applying the Sobel filter onto the refocussed image I:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \cdot I \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \cdot I \tag{3.3}$$

The directional gradients are simply added up to the response value

$$D'(x,y) = |G_x(x,y)| + |G_y(x,y)| \tag{3.4}$$

Laplace Here we calculate the second derivative Laplacian by applying the Sobel operator twice:

$$D'(x,y) = \text{Laplace}(I)(x,y) = \frac{\partial^2 I}{\partial x^2}(x,y) + \frac{\partial^2 I}{\partial y^2}(x,y) \tag{3.5}$$

Figure 3.1: At different pixel positions we take a look at how the pixel response value behaves for the four different depth-from-focus techniques: Photo consistency (orange), angular correspondence (blue), first derivative (red) and the Laplace method (green). A high value means high confidence (low cost).

In the following, we are going to compare the methods mentioned above qualitatively and quantitatively. In figure 3.1 one can see the pixel response value refocussed at different disparities for all the methods at example points in the test-scene *complextestscene*. Points close to edges as well as points on clear surface with less structure on it are chosen. One can see that the pixel response of the angular correspondence and the photo consistency method for those points show a more consistent behaviour, meaning that only one clear maximum can be seen. The derivative method as well as the Laplace method both seem to have trouble especially on pixel coordinates close to edges. This can be seen most clearly at the edge of the melon, where the first derivative shows two maxima, while the angular correspondence and photo consistency measure find a clean maximum indicating at which depth the point can be found. However, on surfaces with less structure as for example on the potato, the photo consistency measure struggles to find a clear maximum – still, it has the maximum at the right position. Having a look at the actual disparity maps

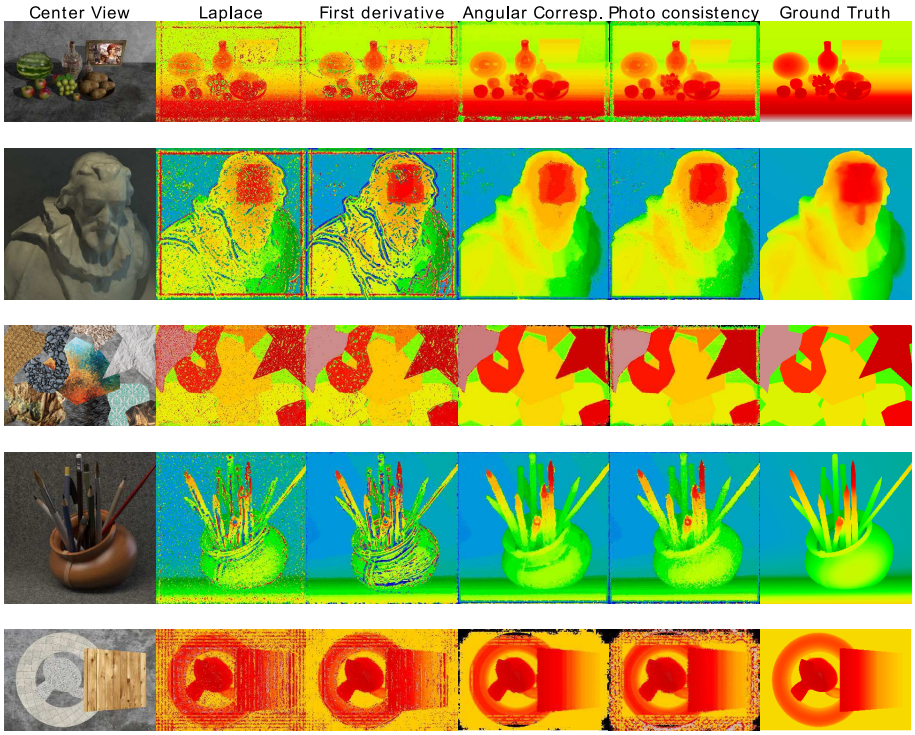

Figure 3.2: The depth maps of the scenes (from up to down) *complextestscene, cotton, occlusiontestscene, pens, testscene, tiltplane* are depicted for different depth-from-focus techniques. The disparity step size is 1/20.

produced by the different techniques (figure 3.2) we can already capture that the angular correspondence and photo consistency method produce the more consistent, smooth disparity maps. A quantitative evaluation confirms this impression: The diagram in figure 3.3 shows the mean relative error for all 4 cues evaluated on the scenes depicted in figure 3.2. It becomes obvious that for all scenes either the photo

consistency or the angular correspondence cue obtain the lowest mean relative error.

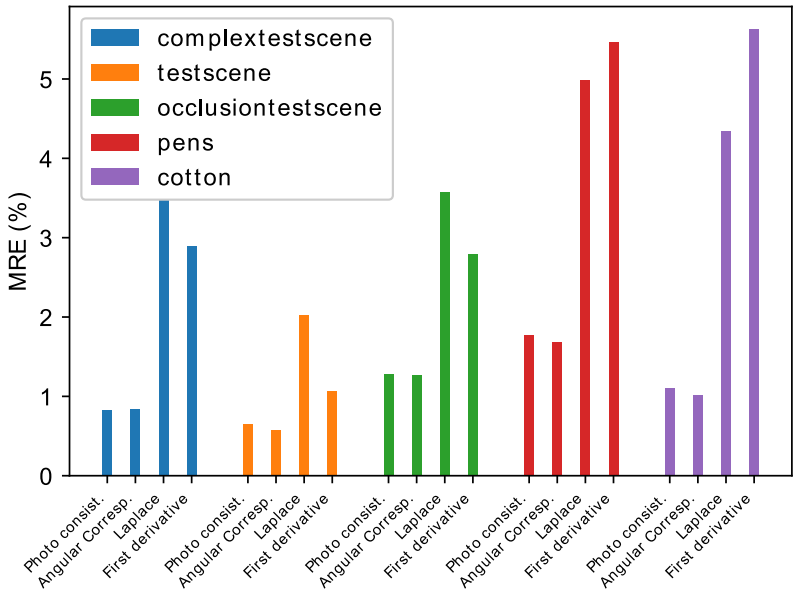

Figure 3.3: The mean squared error is depicted for depth-from-focus techniques. The step size is 1/20.

3.1.1 Using depth-from-refocus as a pre-estimate for the ST

One long-term aim of this work is to make the structure tensor pipeline more robust. We achieve this by using the depth-from-refocus method as a pre-estimate, therefore reducing the disparity window at each pixel. The light field is now refocussed by integer disparity steps, resulting in disparity maps that obtain equidistant layers. In figure 3.6 on the right the scene *complextestscene* is shown with different refocussing stepsizes, where a) shows refocussing on integer step size. All disparity maps with integer disparity step size are shown in figure 3.4, right next to the ground truth integer depth map. From the depth maps one can already guess that the angular correspondence and the photo consistency cue produce the denser depth estimates. This again is confirmed if we take a look at the MRE in figure 3.5: The angular correspondence and photo consistency measures result in the lowest MRE, though the angular correspondence cue is about 1.5 times more time-consuming than the photo consistency measure under the same conditions. The process time for all cues is depicted in figure 3.6 on the left. We stick to the photo consistency measure, as it produces the best results in terms of speed and precision.
The next step is to use the pre-estimate at each pixel to skip the iterative refocussing in the structure tensor pipeline. However, since not all points on an EPI are of the same depth, the EPI is segmented in parts of the same pre-estimation depth. Those parts are refocussed separately to the estimated depth, results are illustrated in figure

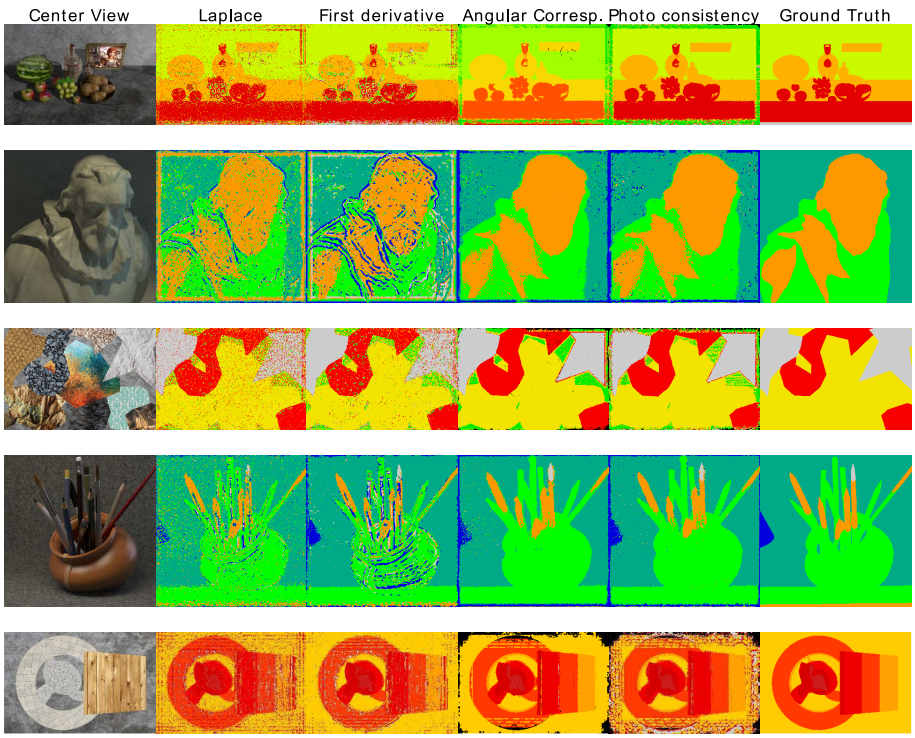

Figure 3.4: The depth maps of the scenes (from up to down) *complextestscene, cotton, occlusiontestscene, pens, testscene* is depicted for different depth-from-focus techniques. On the right the ground truth is depicted. The disparity step size is 1.

3.7. Note that the edges of the depth images have been removed when calculating the MRE. Unfortunately, the photo consistency pre-estimate pipeline yields worse results than the ST pipeline itself. The reason is that artefacts from the pre-estimate persist after the structure tensor pipeline has been applied to the EPI.
Also, the pre-estimation is even more time-consuming than the structure tensor itself. Because of the splitting of the EPI, the runtime is scene dependent. We find significant runtime differences between the scenes, the results can be found in table 3.1.1. If one skips the pre-estimation step and takes the ground truth data as a 'pre-estimate', the pipeline is significantly faster than iteratively refocussing on each depth. The pre-estimation step itself takes about 2 seconds. The ST depth estimation with a depth-from-focus pre-estimation step takes longer than using the ground truth as a pre-estimation since the EPI gets segmented into more split EPIs when using the cue pre-estimate. A global smoothing scheme on the pre-estimate can possibly improve the runtime by minimizing the amount of split EPIs to be calculated, however this would result in even more artefact effects. We leave that aspect for future work.
In summary, we did not achieve an improvement regarding the MRE or the process time, when using the photo consistency measure as a pre-estimate for the structure tensor.

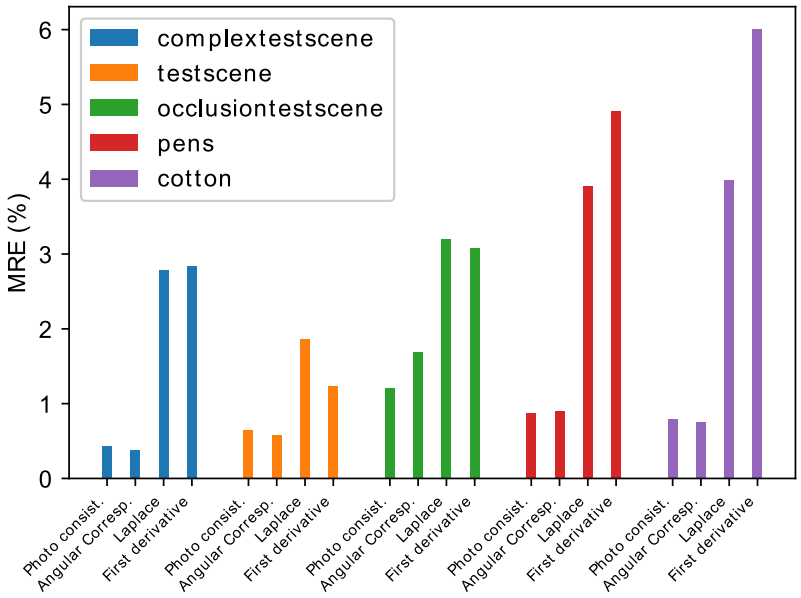

Figure 3.5: The mean squared error is depicted for depth-from-focus techniques. The step size is 1.

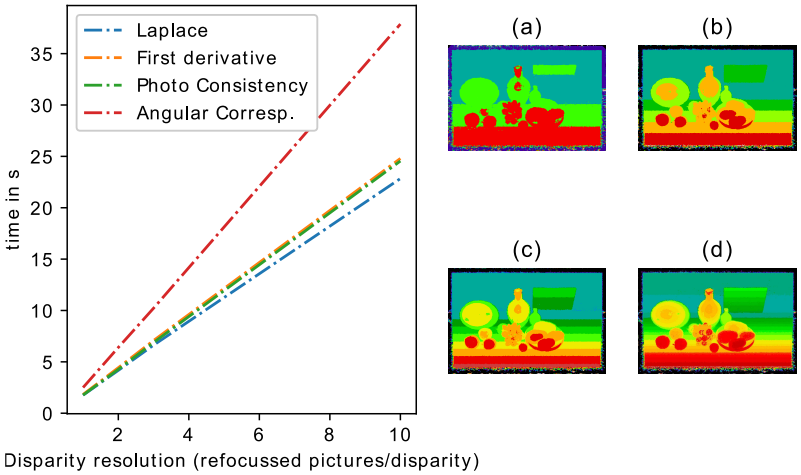

Figure 3.6: On the left, the speed of different depth-from-focus techniques is depicted in function of the resolution chosen. (a)-(d) show the disparity map of the scene *complextestscene* for the resolutions 1, $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{10}$ respectively.

3.2 Modifications on the structure tensor

As depth from focus techniques didn't work out well, we examine different approaches by directly modifying the ST pipeline, with the aim of solving the occlusion problem while maintaining a dense disparity map.
In the following section, we concentrate on benchmark data from the dataset of

Scene	Structure tensor pipeline	Pre-estimate+ ST	GT as pre-estimate
complextestscene	12.89 s	13.62 s	8.83 s
pens	5.1 s	5.2 s	4.7 s
cotton	5.09 s	6.78 s	3.97 s
tiltplane	22.25 s	17.46	9.3 s
occlusiontestscene	10.9 s	12.1 s	6.7 s

Table 3.1: Time for different pipelines. Left: We use no pre-estimate. Middle: A pre-estimation based on the photo consistency cue is used. Right: The ground truth as a pre-estimate is used.

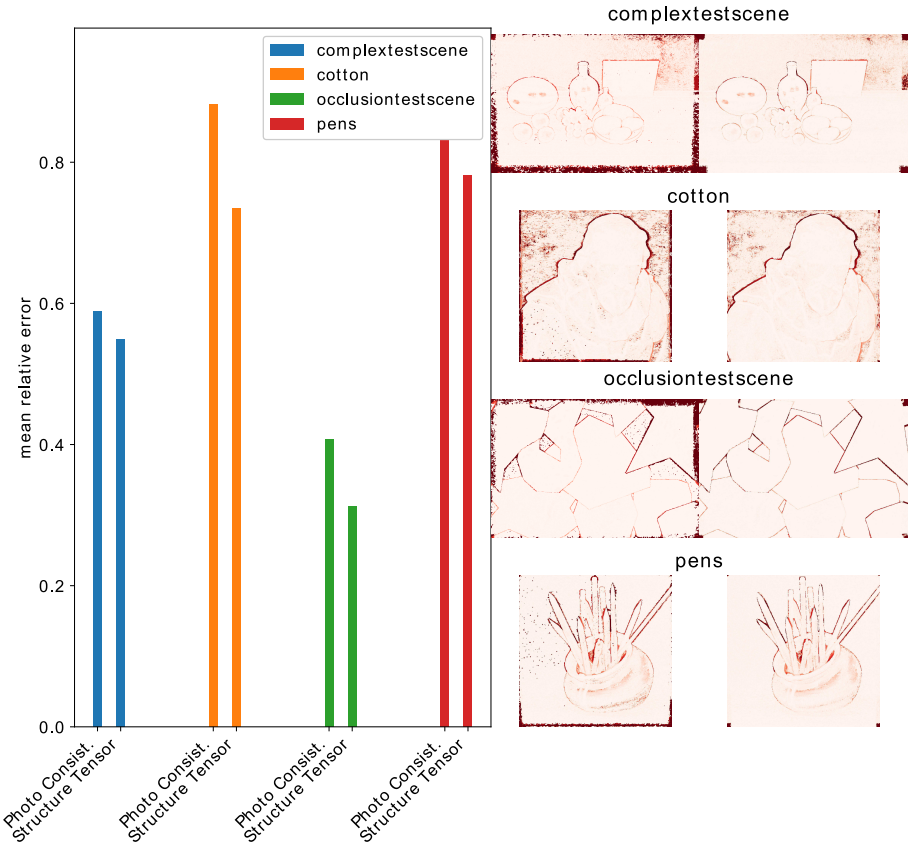

Figure 3.7: Comparison between the ST pipeline with and without a photo consistency pre-estimate. On the left, the MRE is depicted for different scenes. The right images show the absolute difference of the depth map to the ground truth, while the left column represents the results with the photo consistency pre-estimate.

Honauer et al. [2016] as it not only provides a big data set, it also comes with a toolkit for evaluating scenes with different metrics. Whenever it is referred to a metric score a lower score always means better score.

3.2.1 Sandglass kernel

In figure 3.8 the results we obtain from using a sandglass-shaped kernel instead of a normal Gaussian kernel are shown. Even though one could expect the custom-sized kernel to perform better at edges, the opposite is the case; the original Gaussian kernel outperforms the sandglass kernel in all scenes. The motivation to define the kernel in form of a sandglass was to localize structure clearly to the pixel it belongs to. However, it becomes clear that the sandglass kernel cannot handle the occlusion problem explained in section 2.2.2. On the surface regions less smoothing between neighbouring pixels lead to a higher noise resulting in a higher mean squared error.

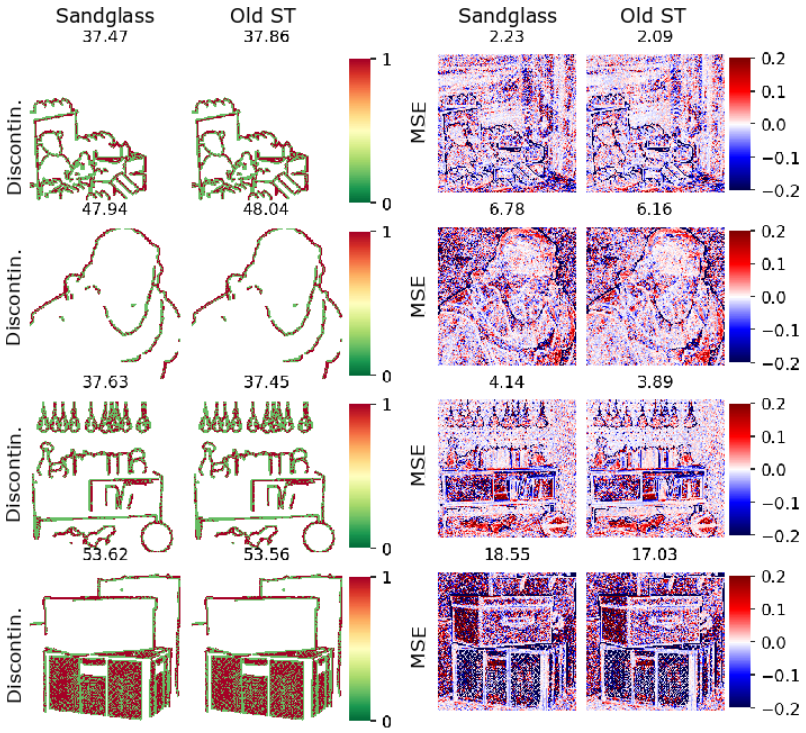

Figure 3.8: Comparison between the normal structure tensor and the sandglass- like kernel for the scenes *dino, cotton, sideboard, boxes*. The left side compares the mean squared errors in the disparity map, the right side shows the error where discontinuities in the disparity map are masked explicitly.

3.2.2 Bilateral filter

This section illustrates the effects of using a bilateral filter in the structure tensor pipeline instead of a Gaussian filter. Figure 3.10 shows the MSE as well as the discontinuity score when using a bilateral filter with the original EPI as a guide. The scores are normed to the score of the ST pipeline. For large σ_{Colour} both converge

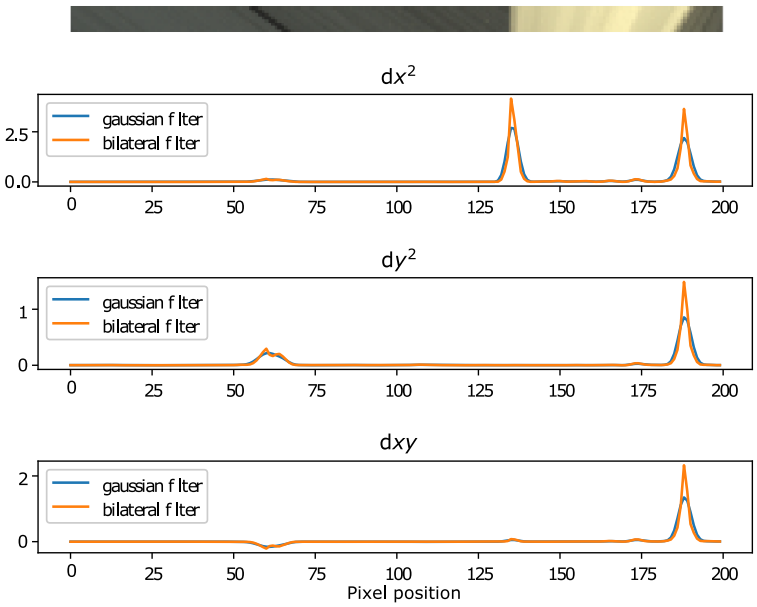

Figure 3.9: Bilateral filtering of an EPI. Upper: EPI. Lower diagram: Structure tensor components after applying Gaussian/bilateral filter respectively.

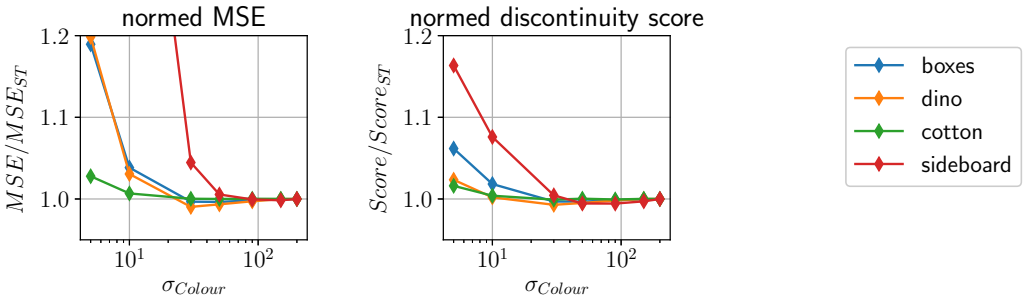

Figure 3.10: Bilateral filtering with original EPI as the guide: On the left, the mean squared error for different scenes in function of the σ_{Colour} of the bilateral filtering is shown, divided by the MSE with using a Gaussian filter. On the right the Discontinuity score based on Honauer et al. [2016] is shown, also normed to the score using the standard Gaussian filter.

to 1, because in this limit the weighting based on the colour space distance (see equation 2.11) is a constant function. Thus, one obtains the same result as with a normal Gaussian filter. Note that the σ_{Space} remains the same as in the case of a normal filtering ($\sigma_{\text{Space}} = 7$ pix). Unfortunately, the MSE metric score and the discontinuity score is worse than with using the Gaussian kernel for the complete parameter space. Especially surfaces with a lot of texture suffer from splitting the weights by their colour values. This becomes clear as the scene *sideboard* with the fine patterned wallpaper reaches the worst results. Any pixel that doesn't encounter

enough neighbouring pixels with an equal colour results in a structure tensor that only takes into account the very local gradient in the EPI, resulting in a noisy disparity estimation.
Since the aim was to especially handle disparity transitions by not mixing up two regions in the EPI with different colours, we take a look at the results for the discontinuity metric defined by Honauer et al. [2016]. Even if one can see a slight improvement at some transitions looking at the disparity maps, the discontinuity score is unaffected since the edge fattening is still visible. It becomes visible in figure 3.9 that the structure tensor components along the EPI do not improve significantly using a bilateral filter due to the pre-smoothing in the ST pipeline. One would expect sharp peaks in the structure tensor components near disparity discontinuities. Since the results do not satisfy the expectation and the guided bilateral filter also takes about 5 times longer than the normal structure tensor, we try an alternative approach.
Instead of using the original EPI as a guide, one could directly take the structure tensor component vector

$$\vec{S} = \begin{pmatrix} \frac{\partial g^2}{\partial x \partial x} \\ \frac{\partial g^2}{\partial s \partial s} \\ \frac{\partial g^2}{\partial x \partial s} \end{pmatrix} \tag{3.6}$$

as the guide for the bilateral filtering of itself. The idea behind it is closely related to the EPI segmentation as described in section 3.2.4. Big gradients in the EPI are more consistently referring to the same transition. By separating small and big gradients, the influence of the structure in front is less significant than it is without using the bilateral filtering. As one can see in figure 3.11, using the ST components as a guide can actually result in a better MSE. For each scene, the score is dependent

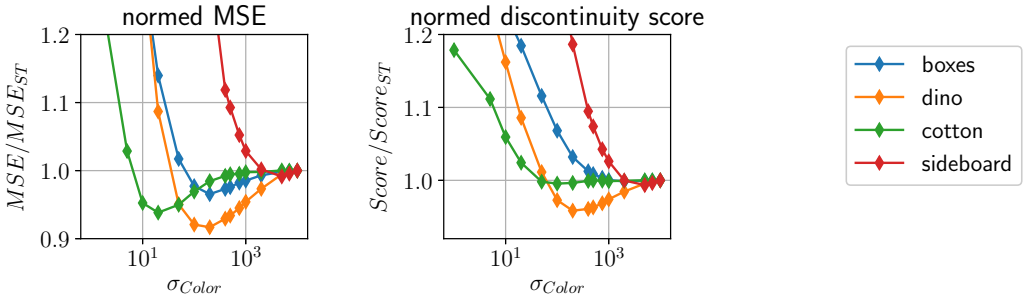

Figure 3.11: Bilateral filtering with structure tensor components themselves as the guide: On the left, the mean squared error for different scenes in function of the σ_{Colour} of the bilateral filtering is shown, divided by the MSE with using a Gaussian filter. On the right the discontinuity score based on Honauer et al. [2016] is shown, also normed to the score using the standard Gaussian filter.

on the parameter σ that is chosen. Also, the discontinuity score does not improve even though in figure 3.12 a clear improvement can be seen at occlusion borders. Actually, the edge fattening is by far bigger than the masks used to determine the

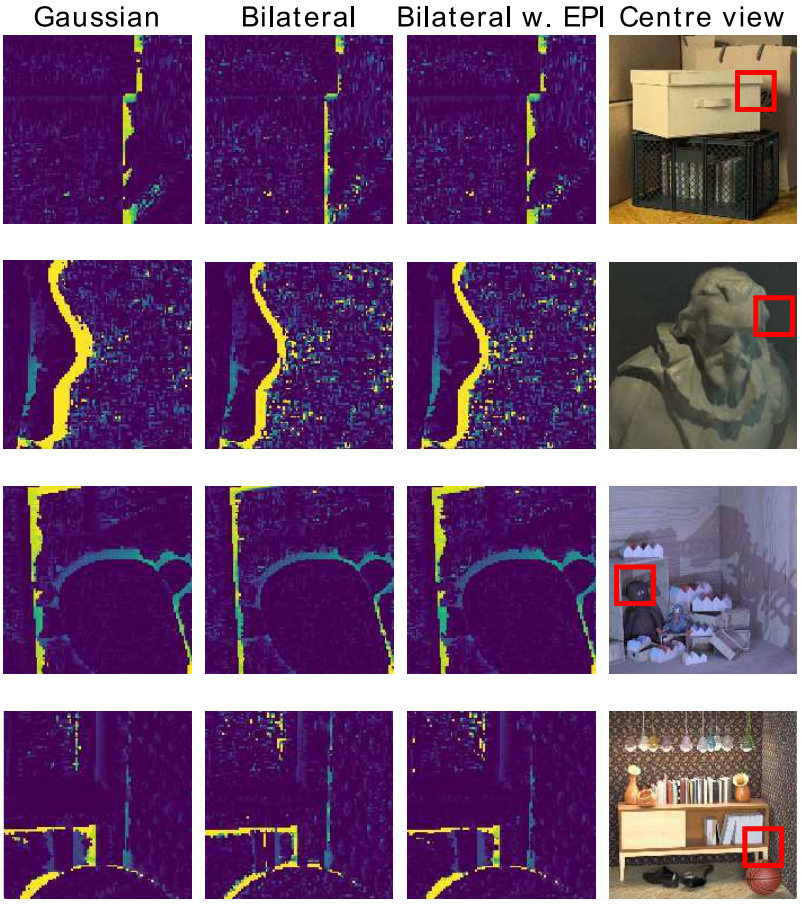

Figure 3.12: Results of Bilateral filtering for the scenes *boxes, cotton, dino, sideboard.*

discontinuity score in figure 3.11. In figure 3.12 The depth maps zoomed at some disparity transitions in the given scenes *boxes, cotton, dino* and *sideboard* are shown, calculated with the Gaussian and bilateral filter. One can observe that the bilateral filter (with the structure tensor component as a guide) results in sharper edges that are significantly less 'fattened' than the Gaussian equivalent. This becomes clear when looking at the head of the teddy bear in the scene *dino*. Even though the front object still occupies more space than it should (edge fattening), the edge has become smaller. This effect cannot be seen when using a bilateral filter with the EPI as a guide, however.
Additionally, it becomes clear when looking at the scene *boxes* in figure 3.12 that the depth estimation using the bilateral filter with the structure tensor component as a guide is less affected by the colour differences in the image. When using the Gaussian ST, the error along the edge is dependent on the background structure

as the transition to the lettering on the bag results in less error than above. The bilateral filter, however, does produce a consistent error that may be better to handle for post-processing.

3.2.3 Thresholding gradients

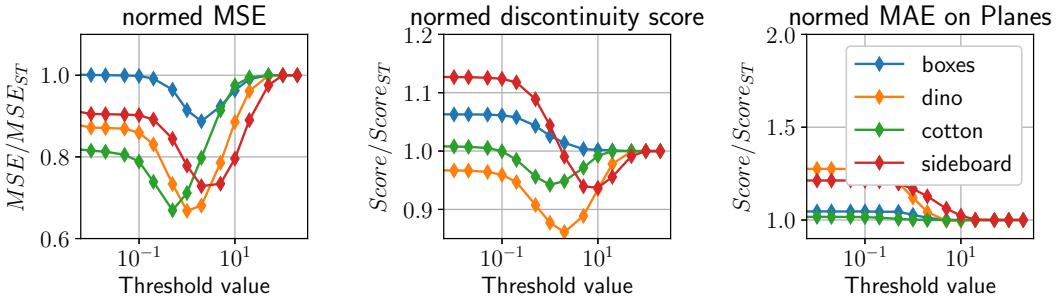

Figure 3.13: Results when thresholding the gradients in the EPI are shown. All results are divided by the score obtained from the normal structure tensor algorithm. On the left, the MSE is shown, in the middle the discontinuity score is shown, on the right we see the MSE on planar surfaces in the scene. All metric scores follow the metrics from Honauer et al. [2016].

When the gradient norm is thresholded as explained in section 2.3.3, the main aim is to change the weights of the gradients contributing to the structure tensor such that all gradients contribute the same. The smallest gradients underlay noise effects, therefore they are not upscaled. A truncation threshold is set so that all gradients in the EPI with a norm bigger than the threshold are normed to that threshold value. As one can see in figure 3.13, for big threshold values the result is equal to the old ST pipeline results as all gradients in the EPI are below the threshold, thus they are unaffected. Setting an extremely low threshold has the same effect as normalizing all gradients to the same strength. Especially on planar surfaces which typically have less texture and therefore low gradient strengths in the EPI, the error score is worse when normalizing the gradient, as one can see in figure 3.13 in the right plot. However, the MSE is mostly affected by the edge fattening error as described in section 2.2.2. Therefore, simple normalizing already improves the MSE, since the gradient strength of the transition weights less into the structure tensor. Even though a low threshold improves the behaviour at discontinuities, the larger noise results in a worse BadPixel score, thus also affecting the discontinuity score. For this reason, no significant improvement can be seen in figure 3.14 in the middle. Since the scene *boxes* is the noisiest one, the MSE is affected least by improving the disparity transitions. The MSE in the scene *cotton* on the other hand is mostly affected by the edge fattening error, thus its MSE is improved most by the

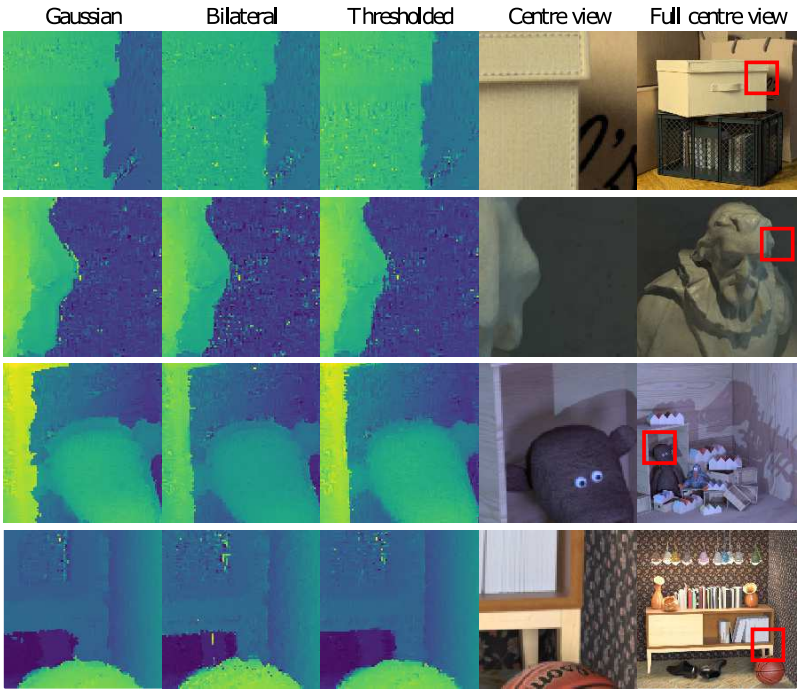

Figure 3.14: Disparity map zoom-ins for different ST modifications. From left to right: Gaussian filtering, bilateral filtering and Gaussian filtering when thresholding the gradients.

thresholding of the gradients. All scenes show an improved MSE for the complete parameter space. Note that the threshold is plotted in a logarithmic scale. In figure 3.14 the thresholded gradient ST result is compared to the bilateral and Gaussian ST result. It is obvious that with a Gaussian kernel, transitions are fattened and erroneous. It seems that the bilateral filter at least for the upper 3 scenes performs best, while the thresholded ST still produces the same errors as the Gaussian ST, as one can see in the *boxes*-scene (upper row). Nevertheless, the additional noise (see scene *sideboard*, last row) produced by the bilateral filtering finally results in a worse MSE score. In figure 3.15 the Gaussian outer kernel size is varied for the old ST and for the thresholded gradient ST. Obviously, the discontinuity score becomes worse for bigger gradients and the error score on planes improves due to bigger kernels being less affected by noise in both cases. In contrast to the old ST pipeline, the MSE stays constant for large kernel sizes.

3.2.4 Occlusion awareness segmentation

In the next step we are trying to detect occlusions in the scan directly via the gradient norm since a high gradient is a good indicator for an occlusion edge (explained in section 2.3.4). The result, though, is dependent on multiple parameters, two of them are the kernel size of the morphology closing kernel. In figure 3.16 (upper row) the metric scores are plotted in function of the y-directional size of the kernel. The

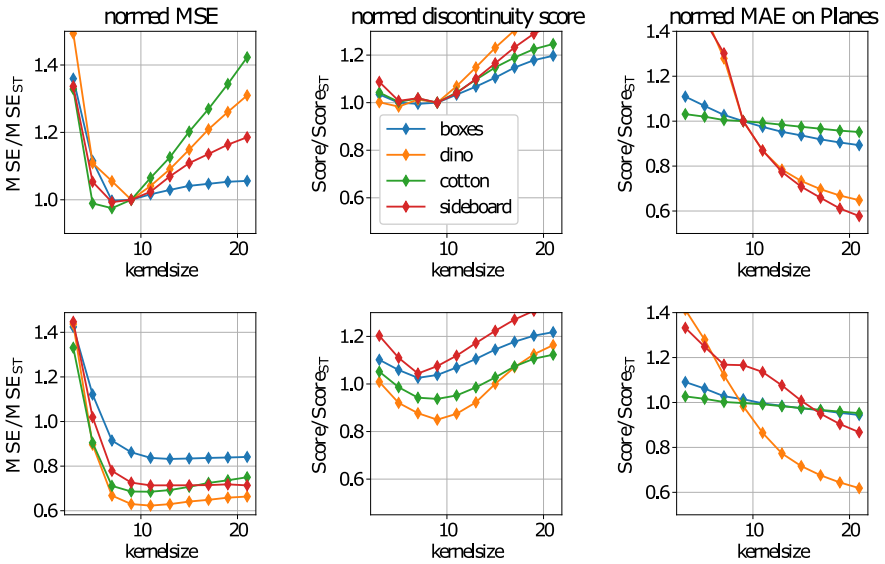

Figure 3.15: Top:Varying the kernel size, using the old ST pipeline. Bottom: Varying the kernel size with thresholding the gradients.

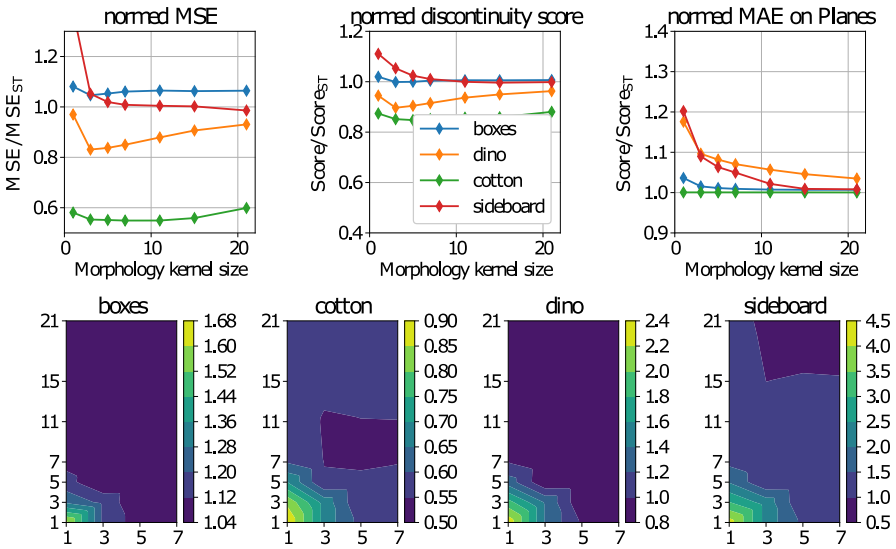

Figure 3.16: Upper row: Metric scores in dependence of the size in y-direction of the morphology kernel, x-direction size is 7.
Lower row: dependence of the MSE for the size of the kernel. the y-size is depicted on the x-axis.

x-morphology kernel is held constant to the size of 7. From figure 3.16 (lower row) we can extract, that for all scenes we obtain the best result when sticking to a 7-pix x-directional kernel size. As the EPI is only 9 pixels wide, we do not use larger kernels.
The larger the kernel size is, the larger the connected parts belonging to the same

segment in the EPI are. More segmentations have a negative effect on the mean average error on planar surfaces with lots of texture, which we can see in the scenes *dino* and *sideboard*. This is confirmed in the upper right plot showing the MAE on planar surfaces for bigger and smaller closing kernels. A larger morphology size kernel results in a better planar surface score, however in the limit for very large kernels no segmentation is happening, thus the relative score is converging to 1. Furthermore, the scenes *cotton* and *dino* are the ones whose mean squared error is mostly dependent on occlusions in the scene. In figure 3.17 the segmentation

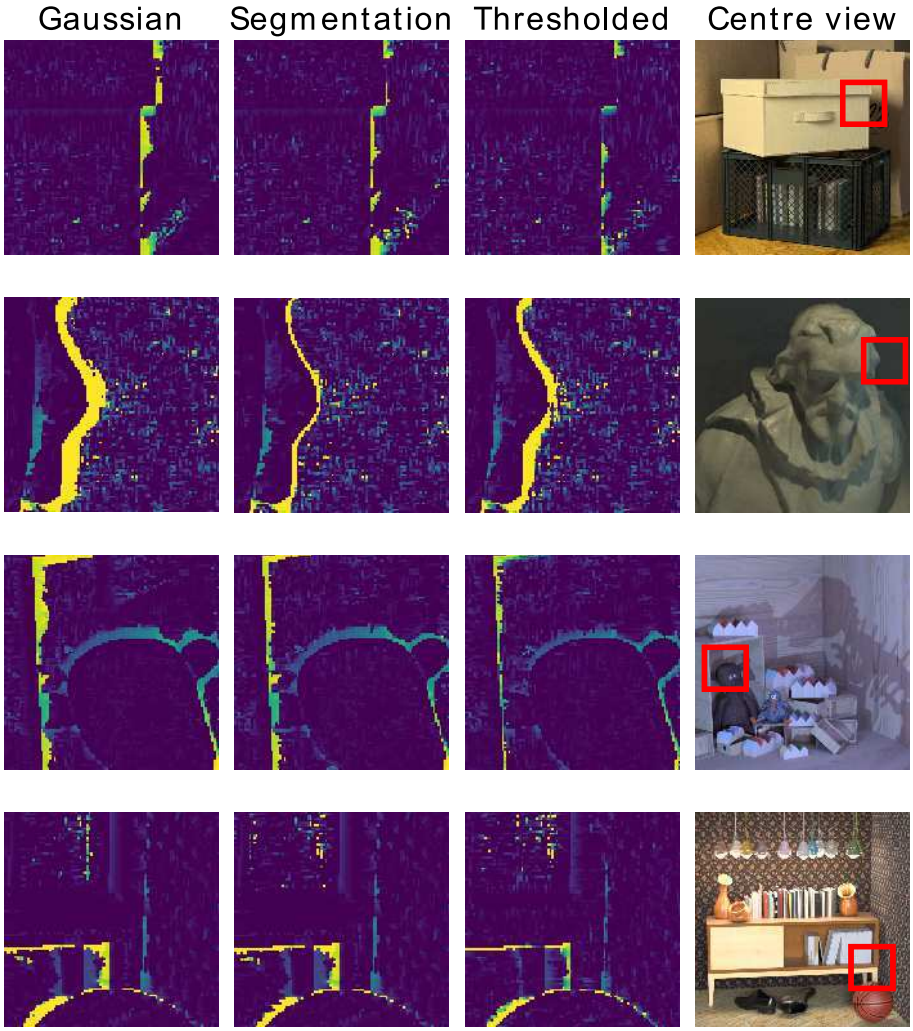

Figure 3.17: Disparity map zoom-ins for different structure tensor filtering. From left to right: Gaussian filtering, segmentation filtering and Gaussian filtering with thresholding the gradients.

algorithm is compared to the old pipeline as well as to the thresholded norm pipeline that is evaluated in the previous section. Again we have a look at the same scenes *boxes, cotton, dino* and *sideboard*. It becomes clear that both algorithms to some degree improve the depth reconstruction near occlusion edges. Still, the results

differ from each other. In the zoom-in on the scene *boxes* the thresholded pipeline seems to achieve slightly better results, however on the occlusion in the scene *cotton* the segmentation algorithm proves to produce better results. The segmentation especially works well with 'clean' edges, where the edge itself results in a clear peak in the norm of the plot (See figure 2.9 in section 2.3.4). In the other scenes, however, the segmentation does not seem to work well as the algorithm only achieves minor improvements.

3.2.5 Alternative coherence

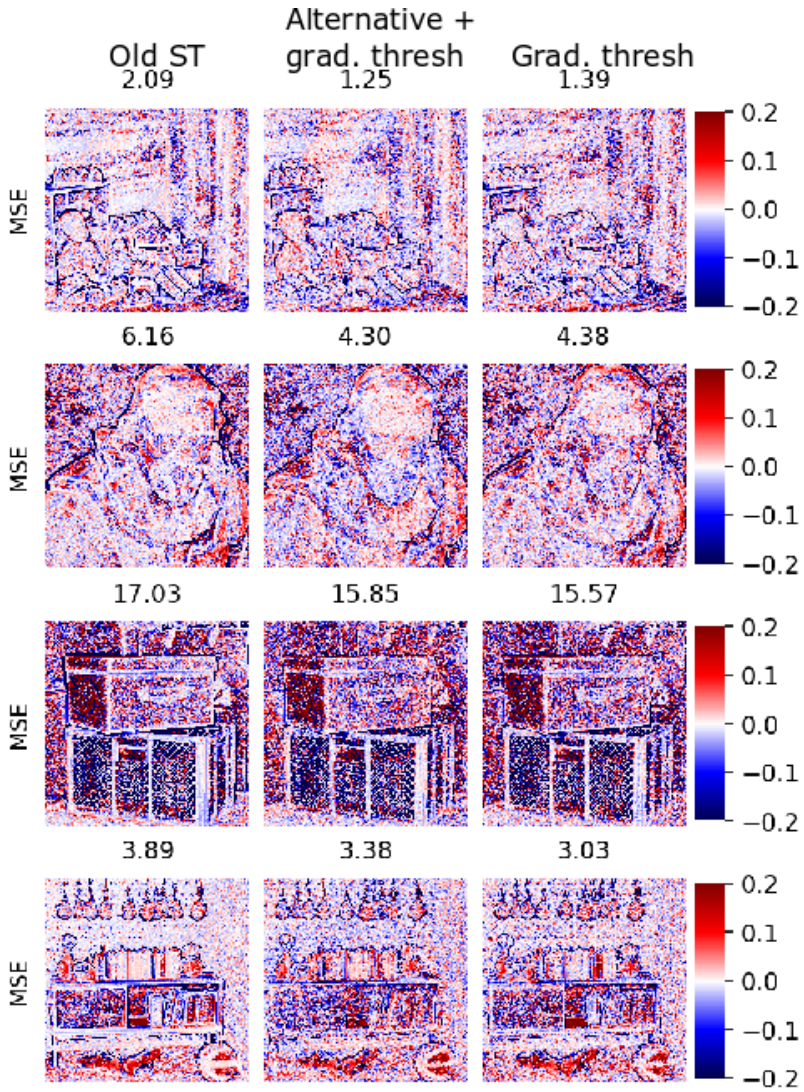

Figure 3.18: Left: The mean squared error using the old ST pipeline. Middle: Using the alternative error measure + thresholding the gradients. Right: Using the old coherence measure + thresholding the gradients.

The use of the alternative cost function as explained in section 2.3.5 only works in combination with thresholding the gradient (section 2.3.3). For this reason, we compare the error maps of the old pipeline, the threshold pipeline and the new error measure to each other in figure 3.18. From the given score above each sub-image, one can conclude that the new error measure does perform about equally as good as the thresholded pipeline. The alternative coherence measure was expected to result in a better disparity map because in theory, it should handle occlusion edges to some degree better than the old coherence measure does. Still, 3.18 shows that the errors remain roughly in the same spots. Both the old coherence and the new error measure

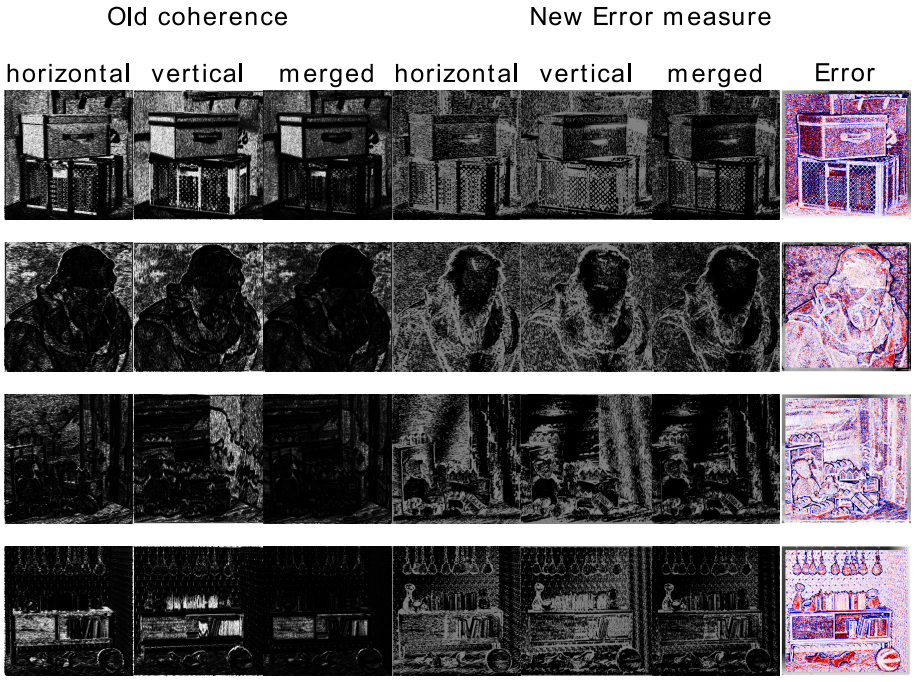

Figure 3.19: Columns on the left: The coherence value from the horizontal EPI, vertical EPI and the merged coherence map. Middle to the right columns: Alternative error measure based on error propagation. On the very right: Actual deviation from the ground truth.

calculate the respective error from the EPI in horizontal and vertical direction. Both views get merged based on the error measure of each view, resulting in a merged coherence/error map. This can be seen in figure 3.19, where for each scene the horizontal, vertical and merged coherence/error map is illustrated as well as the actual error that was produced by the pipeline. Since in figure 3.18 one recognizes that the disparity error maps for both pipelines don't differ much, only the error of the old pipeline is shown in figure 3.19. One can see that the merged error of the new error measure seems to have a bigger overlap with the actual error, making it a better tool for masking out bad pixels than the coherence value. Especially in the scene *cotton* the occlusion edge errors fit better to the error with the new error measure.

To quantify this result, we choose different error thresholds for both pipelines and set all disparity values whose related error is below the threshold to zero. The higher the amount of masked out pixels is, the higher we expect the MSE to be since the number of disparity values being 0 increases linearly (those disparity values also account for the total MSE). This linear relation between the percentage of disparity values set to zero and the total MSE is plotted in figure 3.20 in a logarithmic manner. Using equal

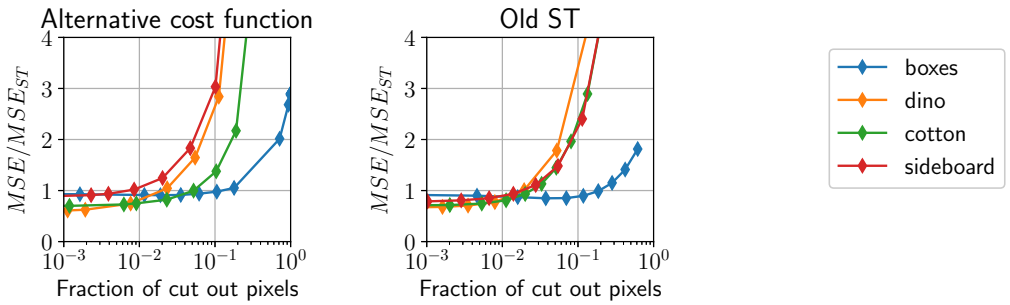

Figure 3.20: The relative MSE is plotted against the percentage of pixels that are set to 0 at different coherence values. On the left, the new error metric is used. On the right, the old pipeline is used.

axes this would result in curves with linear form. The logarithmic scale, however, allows us to evaluate more easily how many pixels have to be set to zero until the result gets significantly worse. As long as the masked out pixels are already highly inaccurate, the resulting MSE doesn't change much by setting them to zero. These plots confirm the observation that the error in the scene *cotton* is prognosticated better by the new error measure, since setting the upper 10 % of the highest error pixels to 0 results in a better score than doing the same with the old coherence measure. In the other scenes, the quality of the error measure is around equal in both cases. In the appendix we show a visualization of the quality of the depth maps using histograms (section B).

3.2.6 Alternative merging of x- and y-direction

When merging the x- and y- directional depth estimations giving preference to the depth value that is further away, one obtains better results on occlusion edges while the number of outliers rises. This trade-off is favourable, as long as the noisy outliers can be handled by post-processing algorithms. The trade-off is controlled via the threshold in equation 2.34 and is set to 0.7 in figure 3.21. In the close-up views of the deviation from the ground truth in all four scenes (column 3 and 4 in figure 3.21) one sees that occlusion edges are significantly improved, especially in scene *boxes* and *dino*. In scene *cotton* the edge remains, however it is smaller. The last scene does not show a significant improvement at occlusions. In all scenes the noise has increased. While on occlusions the coherence does not serve as a good measure

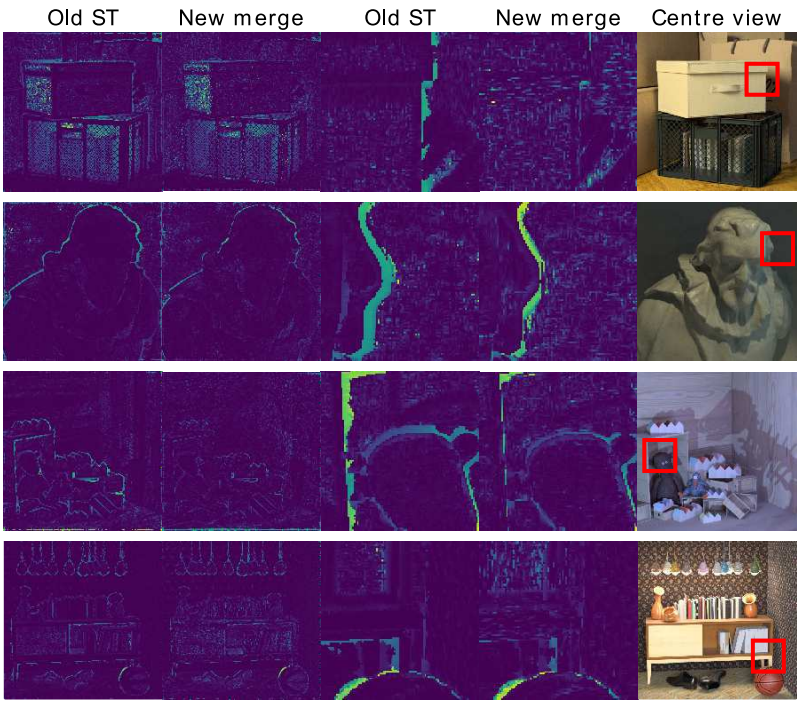

Figure 3.21: Left columns: Comparison between the old merging and the new merging for the whole depth map. The absolute deviation from the ground truth is shown. Middle columns: Close-up view on occlusion edges. Right column: Localisation of the close-up view in the scene. Parameter from equation 2.34 is set to 0.7.

to decide whether to take the x-directional or the y- directional depth estimation, on planar areas a higher coherence value is mostly the better one. As a result, one faces noise as it can be observed well in scene *boxes* and *sideboard*.
Nevertheless, the MSE is improved for all scenes if the threshold value is set between 0 and 0.5 as one can see from figure 3.22 on the left side. The threshold value 0

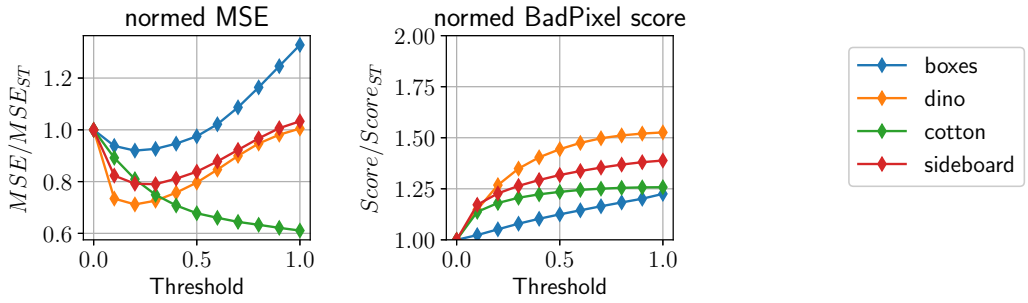

Figure 3.22: Alternative merging: Parameter dependence. Left: Mean squared error. Right: BadPixel score.

corresponds to the old pipeline, the threshold value 1 corresponds to always choosing the lower disparity value. Choosing big threshold values leads to increased noise, thus worsening the MSE score. Note that we can see from figure 3.22 on the right that the BadPixel score is increasing, followingly the amount of pixels that are highly accurate decreases for a high threshold value. However, the occlusion handling can compensate for the noise which is leading to the increased BadPixel score. Further, we can smoothen out slight errors more easily than bigger deviations at occlusion edges. The same plot can be found in figure 3.23 where the algorithm is combined

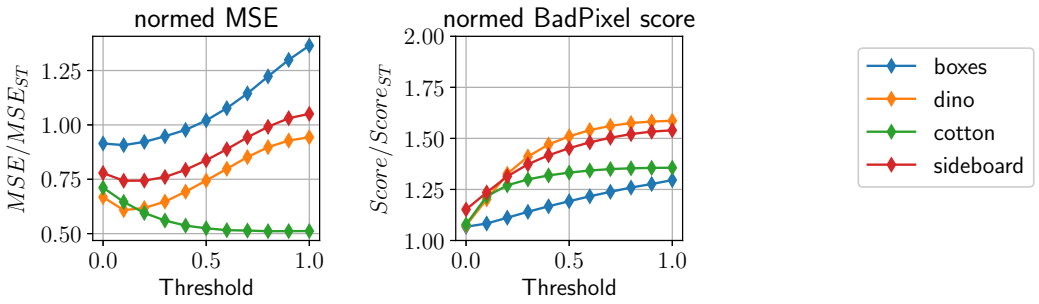

Figure 3.23: Alternative merging + thresholded gradients: Parameter dependence. Left: Mean squared error. Right: BadPixel score.

with thresholding the gradients as described in section 2.3.3. One can see that we can see a slight improvement in the MSE for the scenes *dino* and *sideboard*, while the improvement of the occlusions in scene *boxes* cannot compensate the increasing noise. The scene *cotton* with big occlusion edges obtains the best result for the highest threshold value. Even though it doesn't improve the MSE score it may be favourable to have this merging implemented if it can be handled by post-processing smoothing. This is evaluated using the SGM algorithm in section 3.3.3. Note that the dependent parameter is the one referenced in equation 2.34, the gradient threshold is set to 0.1 . For this reason, the MSE/MSE_{ST} score does not start at 1. Even though in figure 3.22 and figure 3.21 it seems that occlusions can be handled by alternative merging, the gradient threshold does the same, such that both combined do not necessarily improve the result as seen in figure 3.23 on the left. For a higher merging parameter, the BadPixel score is always increasing due to the higher noise coming from low coherence disparity values.

3.3 Semi-global matching

3.3.1 In-process SGM

First, we evaluate the SGM algorithm as described in section 2.5.3. The disparity candidates as well as the related cost values are obtained from the structure tensor estimate on the EPI at different disparity shifts. The SGM algorithm as implemented

here is dependent on 3 parameters as implemented from equation 2.47. We tried

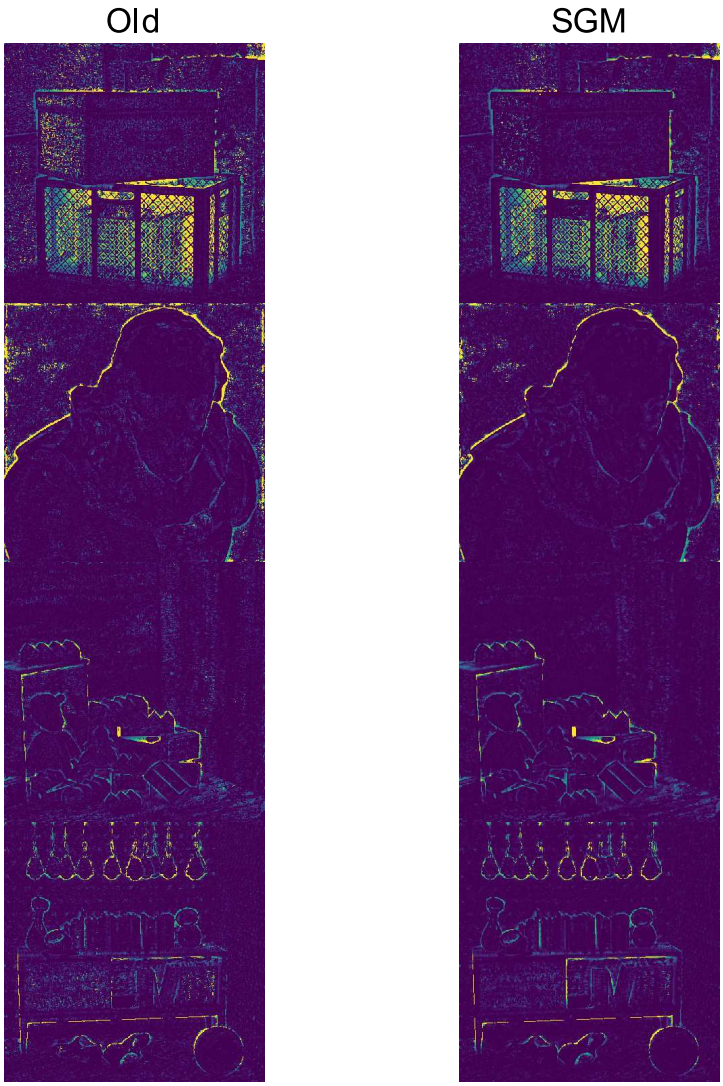

Figure 3.24: Disparity error maps using the old pipeline (left) and the SGM algorithm (right). Yellow is a big deviation, dark-purple is no deviation.

to perform a grid search to find the best parameters. The illustration of the grid search can be found in the appendix in figure C.2, C.3 and C.4 for different metrics. The difference between the maximum and minimum score in the grid search is too small to see any clear dependence on any of the parameters. Note that we sampled the parameter space logarithmically to cover a wider space. Obtaining independence of the parameters is positive for working with the algorithm, the reason for the independence lies in the fact that at most pixels the candidate disparity values between which the SGM algorithm decides are the same. If the structure tensor only finds one unambiguous disparity value, this value is chosen regardless

of the parameters. On planar surfaces this unambiguity is not given, thus the SGM algorithm is able to smoothen such surfaces. This can be seen in figure 3.24, where the old pipeline is compared to the result using the SGM algorithm. The depth map is improved at surfaces with noisy values, thus the SGM has a smoothing effect. At occlusion edges the SGM pipeline does not perform significantly better.

3.3.2 Improved coherence measure by using SGM

Even though the SGM pipeline does not handle occlusion edges well, it still does recognize disparity jumps as high costs in the cost function. In figure 3.25 the cost function is depicted in comparison to the actual error as well as in comparison to the old pipeline. Large disparity jumps are punished with a big penalty, thus they can be detected easily. In scene *cotton* and *sideboard* such disparity jumps are detected, while the other disparity jumps we see are less than one pixel shift and they are only punished by a small penalty. The cost function itself could be used to mask out possibly erroneous pixels. However, the usability of an error measure based on SGM is restricted due to the dependence on the penalty parameters mentioned in equation 2.47.

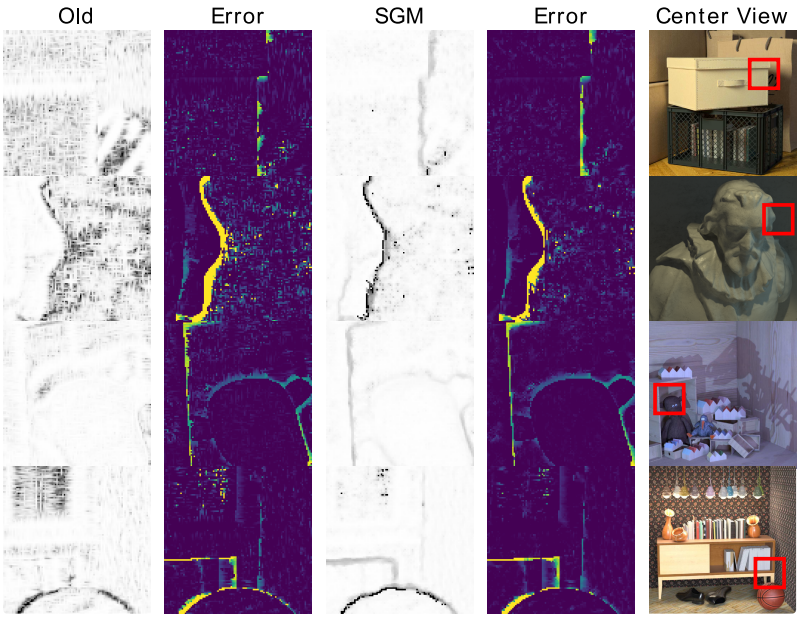

Figure 3.25: The assumed error (grey-valued) when calculating depth without using SGM (left) and with SGM (right). In Colour the actual deviation from ground truth for both pipelines is depicted. Yellow is a big deviation, dark-purple is no deviation.

3.3.3 SGM as pure post-processing

For now, we take as input the processed disparity map from the structure tensor and create the SGM candidates at each pixel from the local neighbourhood as described in section 2.5.4. Figure 3.26 shows how MSE, BadPixel score and processing time

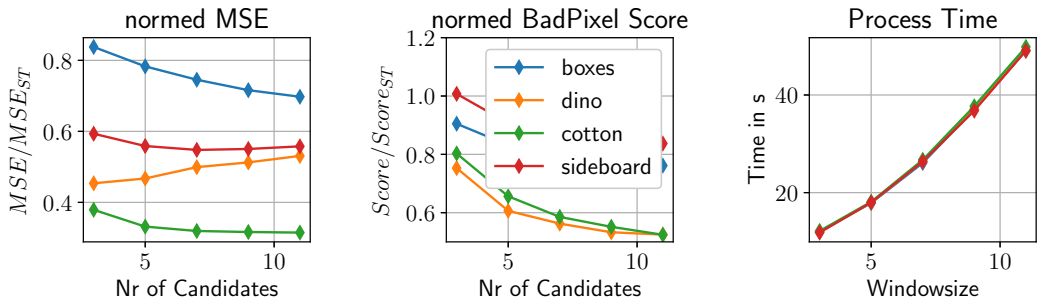

Figure 3.26: SGM as pure post-processing combined with alternative merging: The number of candidates that is chosen from the direct neighbourhood is varied. Left/Middle: The MSE/BadPixel score in dependence of the Nr of candidates. Right: the processing time of the algorithm dependent on the Nr of candidates. The window size is held constant at 5.

depend on the number of candidates from the local neighbourhood which is defined as a 5x5 window around the relevant pixel. The candidates are chosen randomly from the neighbourhood. Note that the input depth map has been created with a modified structure tensor pipeline including thresholded gradients and alternative merging. The gradient threshold is constantly set to 0.1, while the merging parameter is set to 0.3 .

One sees that the MSE not necessarily improves given more candidates. In the scene *cotton* and *sideboard* we reach a fast saturation of the MSE while in the scene *boxes* one would even need more candidates to reach convergence. This has to do with the large noisy regions in the scene that make it impossible to get an acceptable candidate from the neighbourhood. In scene *dino* the MSE even gets worse with a higher number of candidates. On the other hand, the BadPixel score is improved for all scenes. One can conclude that the post-processing SGM manages to improve the depth map at small regions of deviations (outliers), however larger erroneous regions cannot be corrected because of the limited amount of candidate values. Using an iterative approach we try to overcome this problem in the next section.

Since the SGM algorithm is implemented here as a pure post-processing step, one can analyse the effect of applying modifications on the structure tensor beforehand. Figure 3.27 shows the dependence on the merging parameter that is characterized by equation 2.34. No thresholding of the gradient is applied here. Larger parameters lead to better results at discontinuities, however as already shown in section 3.2.6 a higher merging parameter leads to an increased BadPixel score, though the normed score stays below the 1.0 mark. Again scene *boxes* reaches a high MSE due to the

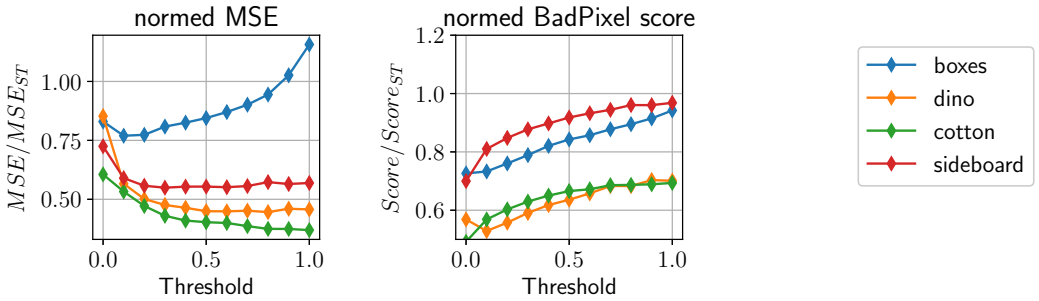

Figure 3.27: Alternative merging with SGM post-processing Left/Right: MSE/BadPixel score as a function of the merging threshold.

large monochrome areas leading to noisy depth maps. One solution to handling such large deviations at surfaces is to mask out contiguous regions of low coherence. This has been implemented for the application on real data, see section 3.3.5.
As we did before in section 3.2.6 we now take a look at the effect of implementing a gradient threshold additionally. In Figure 3.28 the MSE and the BadPixel score are shown in dependence of the merging parameter. While the BadPixel score

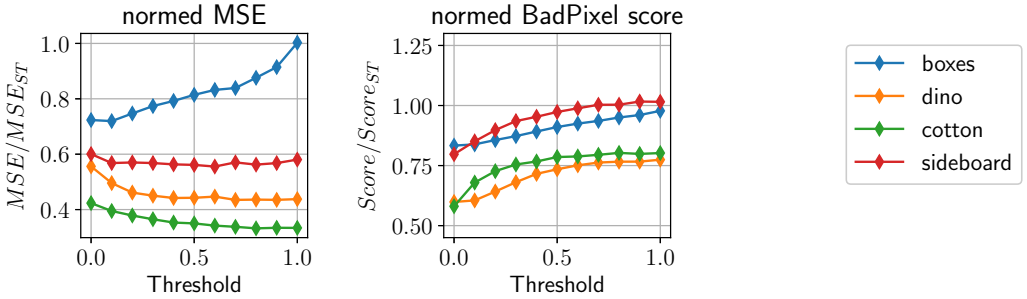

Figure 3.28: Alternative merging with SGM post-processing while using thresholded gradients. Left/Right: MSE/BadPixel score as a function of the merging threshold.

behaves approximately as without a gradient threshold (figure 3.27), the MSE stays approximately constant for all scenes besides scene *boxes*. No big improvement can be obtained since the thresholding of the gradients already handles occlusions well. In figure 3.27 as well as in figure 3.28 the MSE does not rise for a larger merging parameter as it does in figure 3.22 and in 3.23, where no SGM has been applied. The SGM can handle the noisy effects which occur for a high merging parameter.
Using a grid search we tried to investigate the impact of varying the Nr of candidates as well as the window size on different metrics. In figure 3.29 the MSE is shown in dependence of Nr of candidates and window size for all four scenes, in figure 3.30 the BadPixel score is shown and in 3.31 the Discontinuity score is depicted. It becomes clear that the ideal kernel size varies strongly from scene to scene as well

as from metric to metric. Based on the shown results we fix the window size to 5x5 pixels. In general a higher number of candidates promises better results. However, the processing time scales linearly with the number of candidates. Thus we stick to a number of candidates of 5, as shown in figure 3.26 on the right.

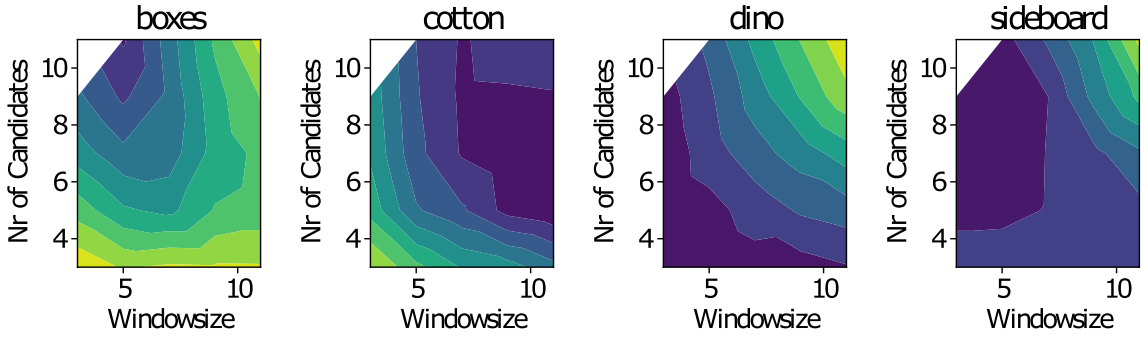

Figure 3.29: MSE Parameter dependence for post-processing SGM using alternative merging+thresholded gradients.

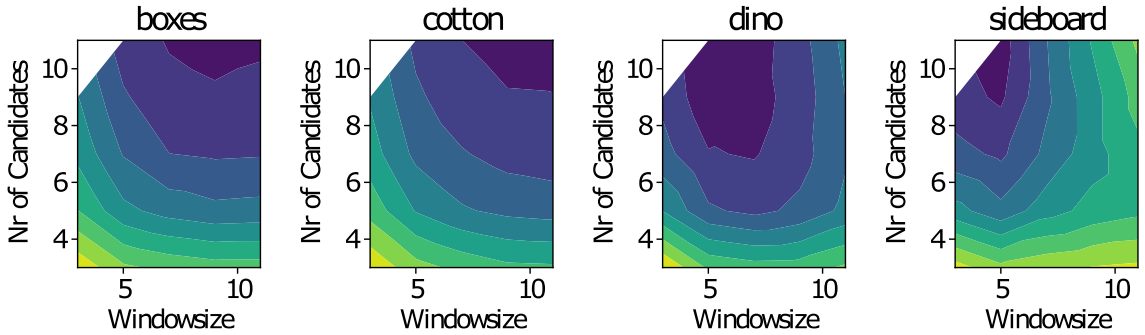

Figure 3.30: BadPixel score parameter dependence for post-processing SGM using alternative merging+thresholded gradients.

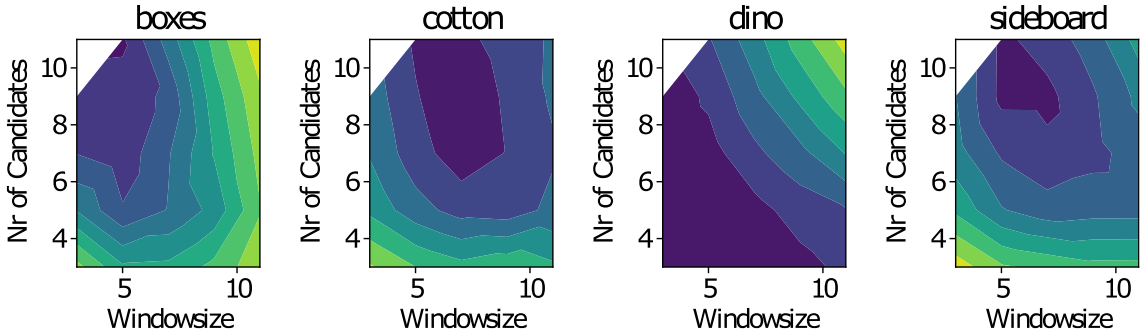

Figure 3.31: Discontinuity score parameter dependence for post-processing SGM using alternative merging+thresholded gradients.

3.3.4 Iterative SGM post-processing

In the following, we evaluate the iterative SGM post-processing as described in section 2.5.5. The iterative optimization should lead to a minimization of the global cost function. The global cost function is defined as the mean cost value at all pixels. In figure 3.32 we confirm the iterative convergence of the mean cost function on the example scene *cotton*. Parallel to the mean cost function the MSE also converges to

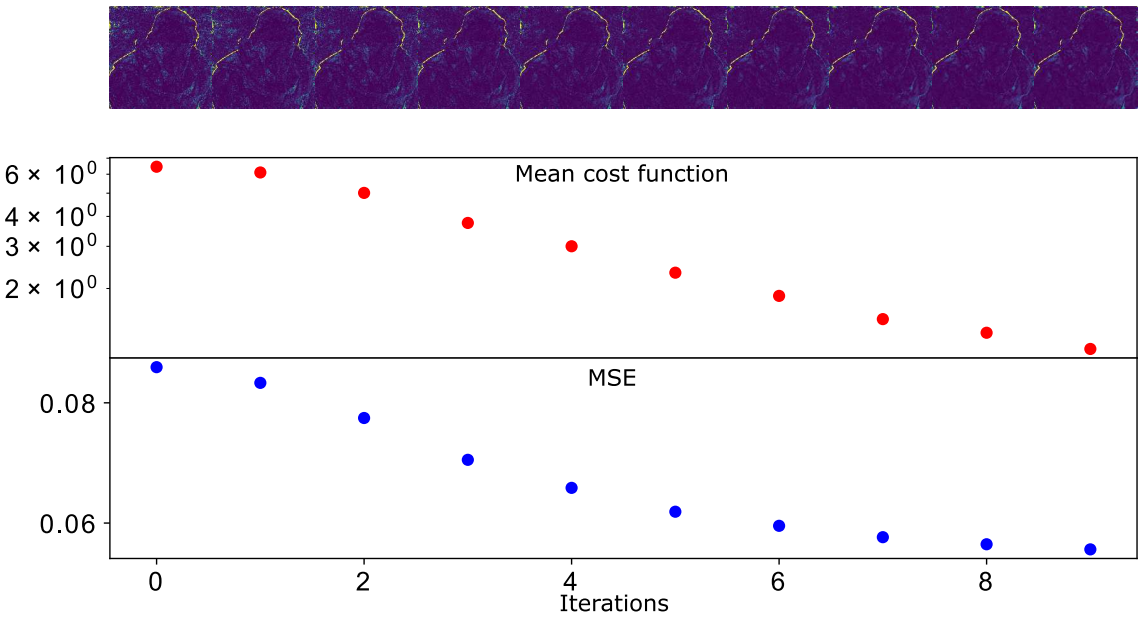

Figure 3.32: In the upper row the resulting depth map deviation from ground truth of the iterative SGM pipeline is shown for scene *cotton*. Below the mean cost function averaged over all pixels as well as the MSE is depicted in dependence of the number of iterations. Note that the MSE is not normed as in figure 3.33.

a minimum. With the given set of parameters one can conclude that approximately 8 iterations are necessary to converge closely to the minimum value. In the upper row of figure 3.32 one can see how the iterative process smoothens outliers in the depth map.
The effect of the number of iterations on the MSE can also be extracted from figure 3.33. On the left, the normed MSE is plotted. It becomes obvious that all scenes converge to a minimal MSE, however at different speed. Also for scene *cotton* and *dino* the BadPixel score drops, while for scene boxes and *sideboard* it stays constant. However, the process time increases linearly with the number of iterations. Thus we stick to a maximum of 5 iterations in the upcoming calculation. This is still far from being comparable to the structure tensor pipeline; however, as mentioned in section 2.5.7 we focus on the feasibility of such method and try to adapt the process time to a reasonable amount of time for our measuring purposes. In the following, we want to examine the effects of changing the initial Δ (see section 2.5.5) on the

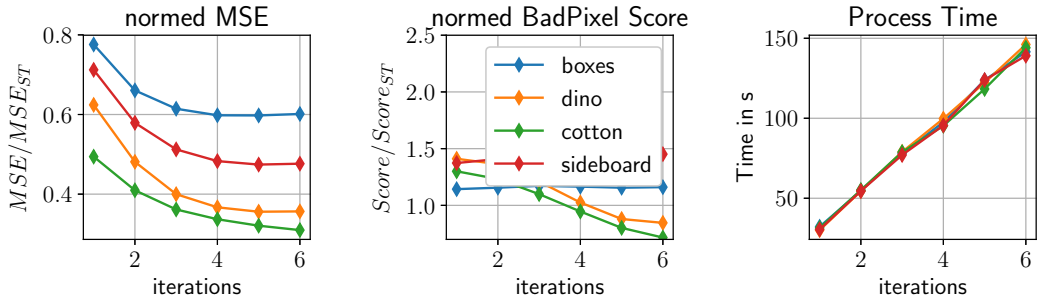

Figure 3.33: Iterative SGM pipeline dependence on number of iterations. Left: Normed mean squared error for all four scenes in function of the number of iterations. Middle: Normed BadPixel score in function of the number of iterations. Right: Processing time.

MSE as well as on the BadPixel score. In Figure 3.34 The MSE and BadPixel score are plotted in function of Δ_{initial}. The iteration number is set to five. For very low

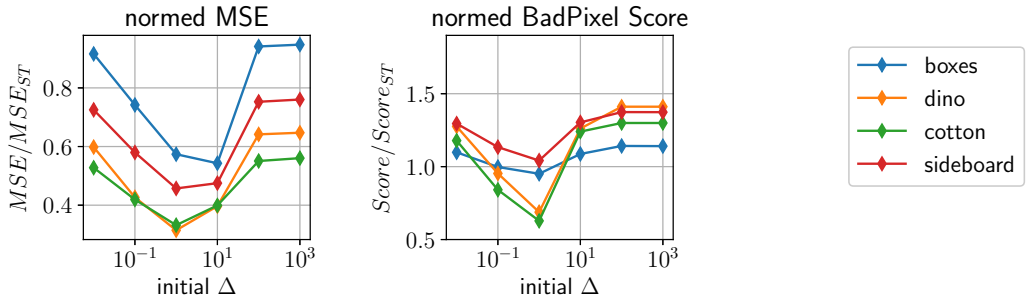

Figure 3.34: Iterative SGM pipeline dependence on initial Δ with fixed number of iterations (5). Left: normed Mean Squared Error for all four scenes in function of Δ. Middle: Normed BadPixel score in function of Δ. Note that the x axis scale is logarithmic.

as well as for high initial Δ no convergence can be reached in a reasonable amount of iterations. Doing only 5 iterations we find the ideal $\Delta_{\text{initial}} \approx 1$. It makes sense to choose the parameter at the same magnitude of the image disparity range to reach any disparity value. The same plot as in figure 3.34 can be found in figure 3.35, with a different number of iterations (10). While the MSE for small initial Δ doesn't change, we observe convergence for larger Δ. Interestingly the overall MSE minimum shifts to the right. Outliers that are distinct from their ground truth can only converge with a large number of iterations as well as a large enough Δ_{initial}. This is now given with $\Delta_{\text{initial}} = 10$. Further, we examine the effect of changing the smoothing parameter P1 from equation 2.50. This parameter changes the form of penalizing disparity deviations from 2 neighbouring points on an SGM scanline. For very small P1 the penalty function is close to being a constant regardless of

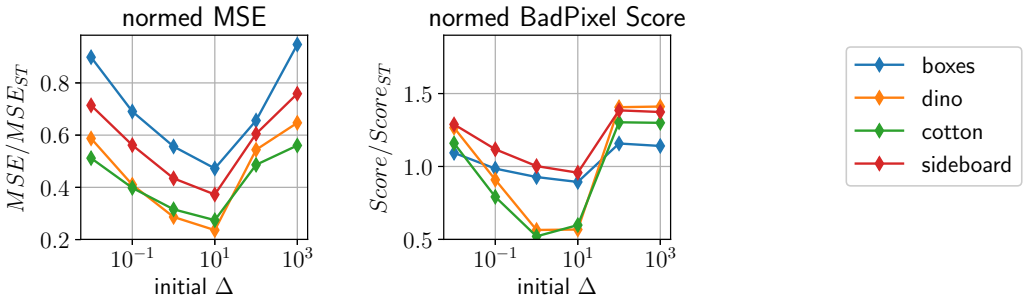

Figure 3.35: Same plot as figure 3.34, doing 10 iterations instead of 5.

the disparity deviations, thus no convergence can be reached. In figure 3.36 the MSE as function of the smoothing parameter P1 can be seen on the left. In the

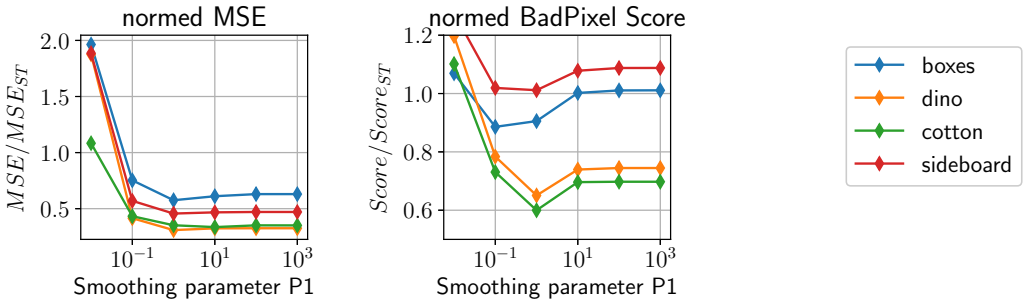

Figure 3.36: Iterative SGM pipeline dependence on smoothing parameter P1 from equation 2.50 with fixed number of iterations (5). Left: Normed mean squared error for all four scenes in function of P1. Middle: Normed BadPixel score in function of P1. Note that the x-axis scale is logarithmic.

limit of a very large parameter P1 the penalty function takes the form of a linear increase as function of the disparity difference between 2 depth values. This in general oversmoothens the edges as shown in figure 2.20. While oversmoothing has a small impact on the MSE it does have impact on the BadPixel score. In the middle of figure 3.36 we see that the BadPixel score reaches a minimum for a smoothing parameter of the magnitude of 1. All scenes are plotted in figure 3.37 in comparison to the ground truth as wee as to the ST pipeline. While outliers such as in the background from scene *cotton* are handled well, noisy surfaces as in scene *boxes* cannot be handled. In all scenes surfaces are smoother after using SGM while edges are still maintained.

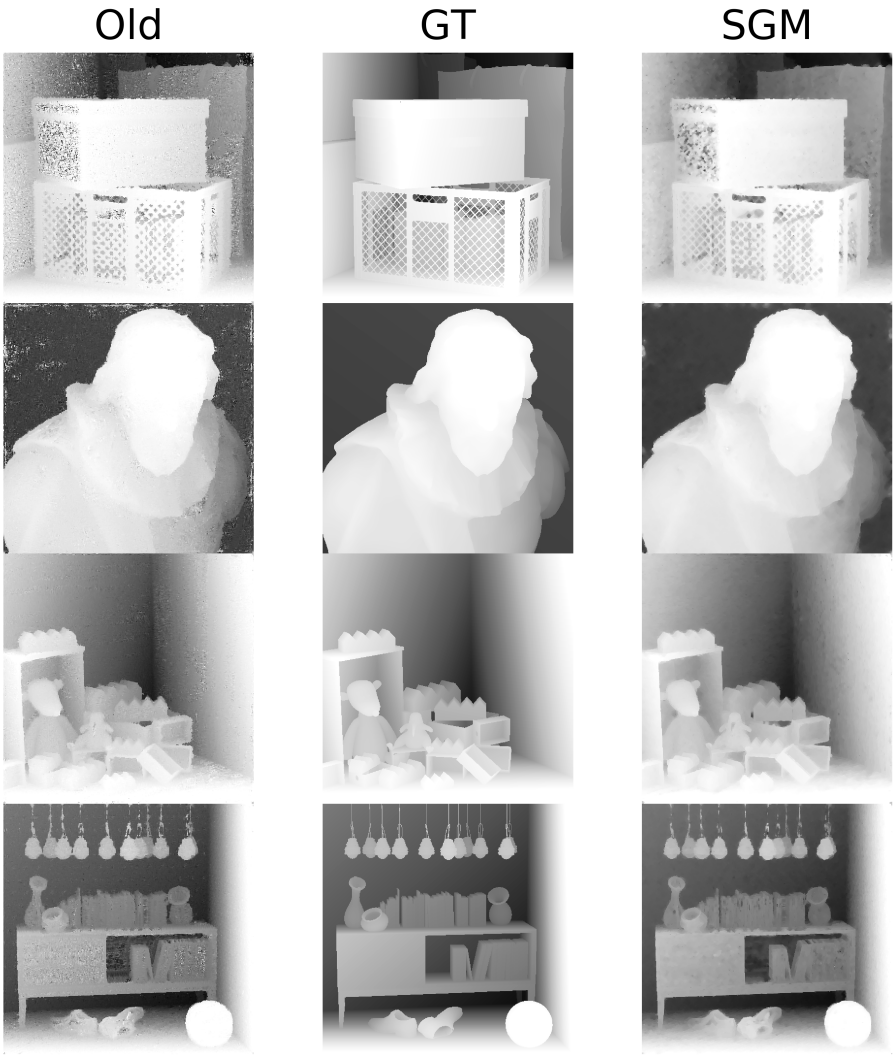

Figure 3.37: Depth map of scenes *boxes, cotton, dino, sideboard* from up to down. Left: Old pipeline depth map. Middle: Ground truth. Right: SGM iterative post-processing.

3.3.5 Application on real data

In this section, we want to evaluate qualitatively the use of SGM post-processing on real captured data instead of synthetic rendered images. Therefore, we have a look at data from the Stanford light field archive (Wiburn [2004]) as well as captured light fields with the *Lumi+* sensor in course of an internship with *HD Vision Systems*. In contrast to the synthetic scenes real captured light fields suffer from bad illumination, slight rectification errors and lens aberrations. Thus the structure tensor coherence is mostly lower, leading to sparse disparity maps (after filtering out disparity values related to low coherence). One could simply run the SGM iterative post-processing algorithm only on pixels with sufficient coherence. However,

we want to also smoothen outliers with low coherence instead of filtering them out. To include outliers for correction while excluding larger areas of bad coherence, we apply morphological opening to the binary mask which has been created based on a coherence threshold. If not mentioned differently, the threshold is set to 0.85. We choose to use a rather small morphological kernel (3x3) to make sure every pixel has at least one neighbour initially which has been measured with sufficient coherence. In figure 3.38 one gets an impression on how SGM post-processing changes the depth map. One can see the sparse depth map (a)) produced by the structure tensor (included gradient threshold already) of a glossy silicium fragmentary used in chemical industry to produce electrical devices. The accompanying mask can be seen in (b)), only pixels in black are evaluated. One can see a lot of white pixels in between the black area which we aim to include in the SGM path-wise optimization. Hence the mask is enhanced with a 3x3 morphological opening kernel such that the new mask defines adjoint areas of low/high coherence (c)). We then run the SGM iterative post-processing only on the pixels masked in black. The resulting depth map is denser (d)). e) shows the centre view of the camera in comparison.

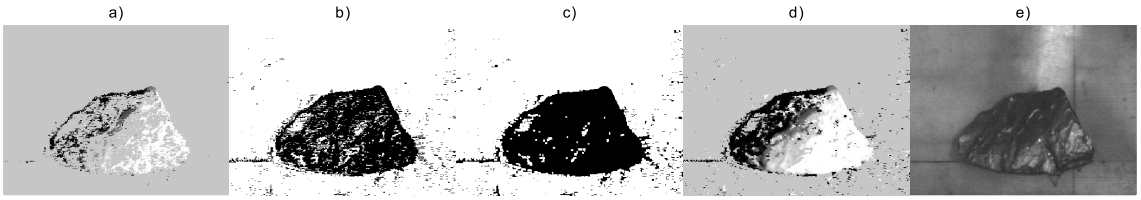

Figure 3.38: Glossy fragmentary scan captured with the *Lumi+* camera. a) Depth map obtained from structure Tensor b) Mask based on coherence value. All white pixels are masked out in [a)]. c) Enhanced mask using morphological opening d) depth map after applying Iterative SGM post-processing on pixels masked in [c)]. e) Centre view of the camera array.

Another application example on real data is the Lego truck from the Stanford light field archive depicted in figure 3.39 (h)). We focus on a small clipping of the scene which is depicted in figure 3.39 (g)). One sees the effects of different coherence thresholds, merging parameters and gradient thresholds in figure a) - f). Figure a) shows the basic ST result with minimal coherence set to 0.85. In contrast figure b) shows the same result with a coherence of 0.5. Values with low coherence added to the depth map lead to a more dense map however the added pixels are mostly noisy. Obviously, the wires in the image are fattened (edge fattening), also the holes on the blue lego part are not visible. In c) and d) we see the same depth maps as in a) and b) despite that we use 'alternative merging'. Still, the wires appear to be a lot wider than they actually are, yet one can observe an improved refinement at the holes in the blue Lego part. In d) (lower minimal coherence) even more details on the crane are visible since the 'alternative merging' implicitly is restricted by minimal coherence. However this comes at the cost of noise. The figure e) shows the disparity map with a 'gradient threshold' applied. This, in general, leads to a sparse structure

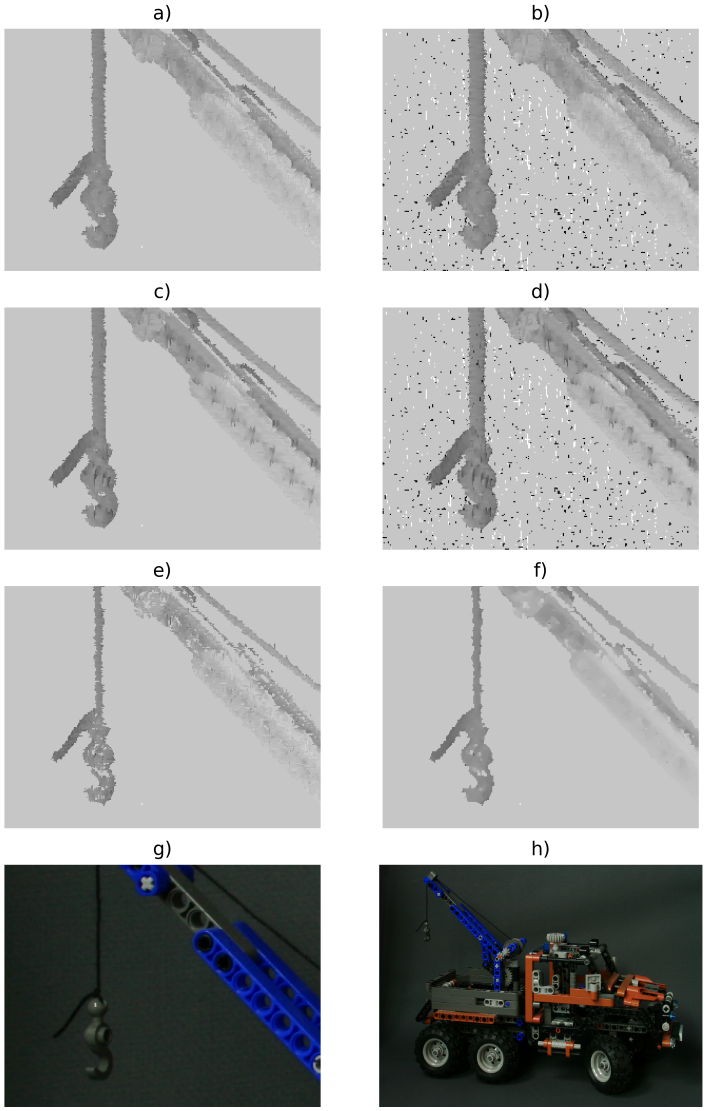

Figure 3.39: a) Depth map prod. by the structure tensor pipeline, with minimal coherence set to 0.85. b) Depth map prod. by the structure tensor pipeline, with minimal coherence set to 0.5. c) Depth map when using alternative merging, minimal coherence set to 0.85. d) Depth map when using alternative merging, minimal coherence set to 0.5. e) Depth map when thresholding gradients, minimal coherence set to 0.85. f) Depth map when thresholding gradients and using iterative SGM post-processing, minimal coherence set to 0.85. g) Centre view clipping of the scene h) full centre view of the scene.

compared to a), while details are finer, especially the edge fattening is drastically reduced. All wires on the crane in the clipping are as wide as they are meant to be. After that we apply post-processing SGM (5 iterations) in figure f) in the disparity

map obtained in e). Here the surface is closed without oversmoothing the structure. One finds the full depth map without clipping in the appendix (section D).

3.4 Comparison of results

The table 3.2 shows all MSE values as well as the processing time for all modifications and algorithms we implemented. While some ideas (e.g. the Sandglass shaped kernel) did not bring the results we assumed beforehand, others did. **Gradient thresholding** does improve the result by ca. 30 % while taking about 30% more processing time. To differentiate between improvement related to object details and overall smoothness, we applied a Gaussian filter as well as a Median filter on the results in a post-processing step. After filtering, the thresholded gradient pipeline performs even better and outmatches the old pipeline (with PPR filtering) by ca. 40 % in both cases. The **alternative merging** does only improve on very small details of the depth map, since occlusion problems are already handled by **gradient thresholding**. Nevertheless the MSE is improved slightly when adding **alternative merging** to the gradient threshold pipeline.
Without post-processing the **semi-global matching + gradient thresholding** pipeline reaches the best scores. Implementing the alternative coherence measure in the structure tensor pipeline only improves the result slightly, it even worsens in the case of scene *sideboard*.
As overall best performs the **iterative post-processing semi-global matching** pipeline with 10 iterations, however the implementation acquires almost 2 minutes for calculation. With only 5 iterations, the result is only a little worse than with 10 iterations. Applying a filter afterwards only worsens the MSE.

Method	Time(s) (cotton)	MSE			
		boxes	cotton	dino	sideboard
Old Pipeline	4.78	17.03	6.16	2.09	3.89
Sandglass kernel	6.81	18.55	6.78	2.23	4.14
EPI-Bilateral filter ($\sigma = 90$)	43.96	19.26	6.41	2.22	4.16
Bilateral filter($\sigma = 750$)	20.18	18.95	6.38	2.10	4.32
Gradient threshold (0.1)	6.06	15.57	4.38	1.39	3.03
Gradient threshold (0.1) + Alternative merging(0.3)	6.91	16.14	3.45	1.35	2.95
Gradient threshold (0.1) + Occlusion segmentation (7x7 morph. Filter)	11.28	18.08	3.39	1.78	3.91
Alternative coherence + gradient threshold (0.1)	14.15	15.85	4.30	1.25	3.38
Semi-global matching Params: (4000,64000,1000)	8.90	14.98	3.84	1.75	3.18
Semi-global matching Params: (4000,64000,1000) + gradient threshold (0.1)	12.75	13.31	2.96	1.11	2.43
PPR Gauss filtering (3x3)	4.80	12.59	4.80	1.68	2.92
PPR Gauss filtering (5x5)	4.80	11.95	4.54	1.59	2.72
PPR Gauss filtering (3x3) + gradient threshold (0.1)	6.12	10.29	3.04	0.97	2.04
PPR Gauss filtering (5x5) + gradient threshold (0.1)	6.12	9.63	2.81	0.91	1.90
PPR Semi-global matching Params: (4000, 256000, 400) + gradient threshold (0.1)	13.2	12.0	2.15	1.08	1.98
PPR Median filtering (5x5)	4.83	12.73	4.76	1.76	3.20
PPR Median filtering (5x5) + gradient threshold (0.1)	6.11	10.15	2.95	1.01	2.17
PPR Semi-global matching+ 5 iterations	54	9.8	2.05	0.66	1.78
PPR Semi-global matching+ 10 iterations	114	**8.71**	**1.94**	**0.53**	**1.59**
PPR Semi-global matching+ 10 iterations + Median filtering (3x3)	114	9.02	2.50	0.75	1.64
PPR Semi-global matching+ 10 iterations + Median filtering (5x5)	114	9.28	2.50	0.75	1.64

Table 3.2: Comparison of all methods on MSE metric. PPR stands for post-processing. The relevant parameters are found in parentesis.

4 Resumee and outlook

In the following, we will give an overview of the different methods that were implemented an tested to improve the depth map obtained from EPI disparity estimation using the structure tensor.
Firstly, we tested different depth-from-defocus techniques to pick out the most suitable for creating a coarse depth map. This depth map was used as a pre-estimate to limit the disparity space. The idea was to fasten up the structure tensor pipeline while avoiding outliers beforehand. However, it turned out that on the one hand, the obtained coarse depth maps were not precise enough. On the other hand, no significant speed-up could be obtained even with perfect pre-estimates. For that reasons we dropped the idea and continued to implement modifications in the structure tensor pipeline, which lead to significant improvements in the MSE metric. Some methods did not bring the results we hoped for (sandglass shaped kernel, bilateral filtering in the ST). An alternative coherence measurement has been developed based on first order error propagation, leading to slightly improved results to a cost of processing time. Occlusion handling with segmenting the EPIs based on the gradient norm showed some good results in scene *cotton*, though it needs clear contrast edges in the scene to identify depth discontinuities. Implementing an alternative merging of x- and y- directional disparity estimations allowed for higher detail resolution and occlusion handling. However, occlusions are taken care of with implementing a truncation threshold to the gradient in the EPI. This modification is easily implemented, does not significantly take longer in processing and noticeably improves the MSE for all scenes.
In accordance to its original use we implemented the semi-global matching algorithm in the pipeline, though we realized that results were better when proposing disparity candidate values artificially in a post-processing step solely dependent on the estimated depth map. We minimized the cost function to get a smooth result while retaining clear edges in an iterative manner. However in its present form, this algorithm takes almost 2 minutes to calculate on 512x512 images.
Based on other publications it is expected that the SGM algorithm can be implemented with higher efficiency. As future work, we propose to implement the iterative post-processing SGM algorithm on a GPU to reach the highest efficiency and possibly allow SGM to be applied in real-time.
Further, we propose to do quantitative real data tests with the structure tensor by using a laser-scanned test scene. By using only synthetic data, all effects coming from aberrations and calibration uncertainties are ignored.
Next to sparse mapping and occlusion handling another issue of the structure tensor depth estimation are glossy surfaces. As part of a side project we tried to locate

and separate glossy surfaces in the scene based on the work of Tao et al. [2017]. Unfortunately, we did not yield satisfying results, leaving glossy surface handling for further research.

However, while working on this thesis, papers have been published (Luo et al. [2017], Changha Shin [2018]) that present machine-learning based methods to reconstruct depth map from light field data yielding very good results in the HCI 4D light field benchmark ([1]).
In another side project, we trained a neural network based on Tensorflow ([5]) which takes as input a single EPI (9x512) and assigns 512 depth values to it. It has been trained with the EPIs from the synthetic data we used in this work. Qualitative results can be found in the appendix. As our CNN already produced acceptable results and state of the art methods are machine-learning based, the future research on training neural nets for light field depth estimation is promising in our eyes.

Appendix

A Derivation of the structure tensor

The derivation is taken from Jähne [2013]. Taking a function $g : \Omega \to R, \Omega \subset R^D$, the preferred local direction $\vec{n} \subset R^D$ must satisfy the following equation:

$$(g^T \vec{n})^2 = |\nabla g|^2 \cos^2(\sphericalangle(g, \vec{n})) \tag{A.1}$$

If ∇g is parallel or antiparallel to $\vec{n}$, the expression on the right side reaches a maximum. Therefore one needs to maximise the left hand expression in a local environment:

$$\vec{n}_{\text{preferred}} = \underset{n}{\text{argmax}} \left(\int w(\vec{x} - \vec{x'}) \left(\nabla g(\vec{x'})^T \vec{n} \right)^2 d^D x' \right), \tag{A.2}$$

w is a window function defining the size of the local environment. Multiplying with $\vec{n}$ we obtain:

$$\vec{n}_{\text{preferred}} = \underset{n}{\text{argmax}} \left(\vec{n} J \vec{n} \right) \tag{A.3}$$

$$J = \int w(\vec{x} - \vec{x'}) \left(\nabla g(\vec{x'}) \nabla g(\vec{x'})^T \right) d^D x' \tag{A.4}$$

This results in a $D \times D$ tensor of the form

$$J_{pq} = \int_{-\infty}^{\infty} w(\vec{x} - \vec{x'}) \left(\frac{g(\partial \vec{x'})}{\partial x'_p} \frac{g(\partial \vec{x'})}{\partial x'_q} \right) d^D x'. \tag{A.5}$$

In two dimensions we can write

$$J = \begin{pmatrix} w * \frac{\partial g}{\partial x} \frac{\partial g}{\partial x} & w * \frac{\partial g}{\partial x} \frac{\partial g}{\partial s} \\ w * \frac{\partial g}{\partial s} \frac{\partial g}{\partial x} & w * \frac{\partial g}{\partial s} \frac{\partial g}{\partial s} \end{pmatrix}, \tag{A.6}$$

where '$*$' describes a convolution.

B Alternative Coherence: Visualizing the quality of a mask

In the work we propose an alternative coherence measurement which creates a confidence map. As we want to use the confidence map also to mask out erroneous depth map pixels, we are looking for a metric to evaluate the quality of such a mask with dependence on the scene. The confidence map is thresholded to obtain the binary mask.
In the evaluation in section 3.2.5 we observe the behaviour of the MSE when varying the threshold (see figure 3.20). Masked out pixels are set to zero. Thus, they are worsening the MSE. We also choose another approach to visualize the quality of the depth map. We simply show the histogram of the deviations of ground truth for each scene, using different mask thresholds. Supposed we would have a perfect

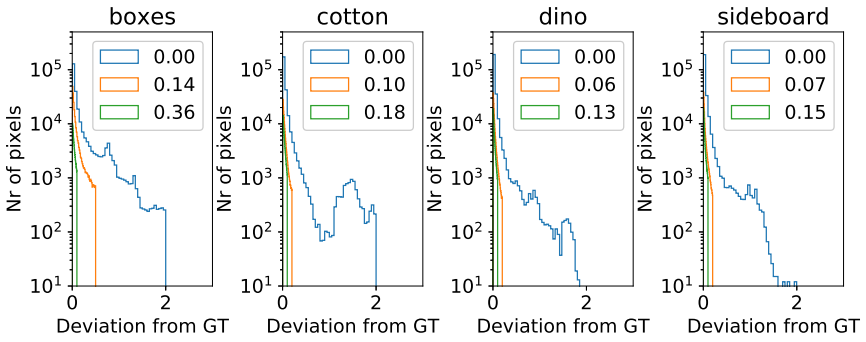

Figure B.1: The error histogram for different masks supposed one has found the ideal mask . The labelling indicates the fraction of masked out pixels.

confidence map that is shows low confidence at points that have a high deviation of ground truth, one would expect the histograms to look as in figure B.1. When varying the threshold such that the fraction of masked out pixels rises, the number of pixels with high deviation should decrease. On the other hand, when the confidence map is random, all pixels are masked out at the same fraction regardless of their deviation, as seen in figure B.2. In figure B.3 and B.4 the histograms for the old coherence and the new confidence measure are shown. As we observed in section 3.2.5, the results are fairly equal. With different confidence maps, thresholding the confidence map leads to different fractions of cut-out pixels. Two confidence maps are only comparable when the fraction is about equal.

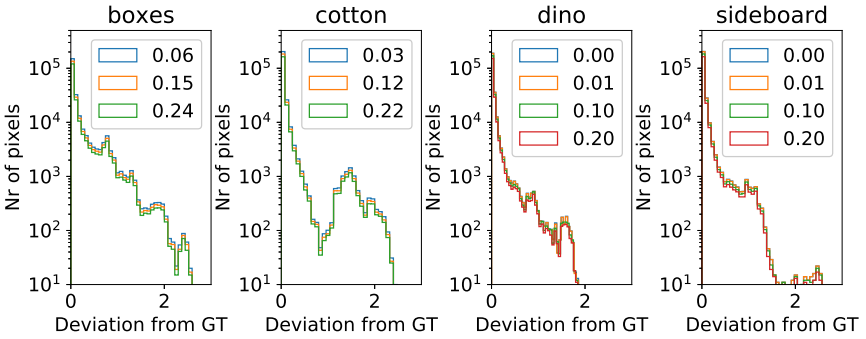

Figure B.2: The error histogram for different masks supposed the mask is completely random. The labelling indicates the fraction of masked out pixels.

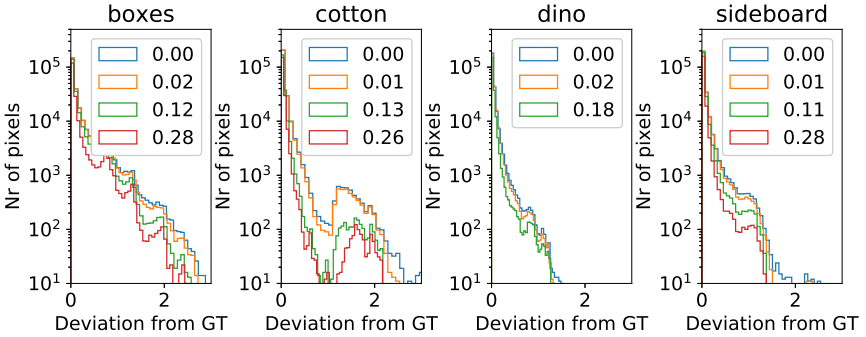

Figure B.3: The error histogram for different masks based on thresholding the coherence map. The labelling indicates the fraction of masked out pixels.

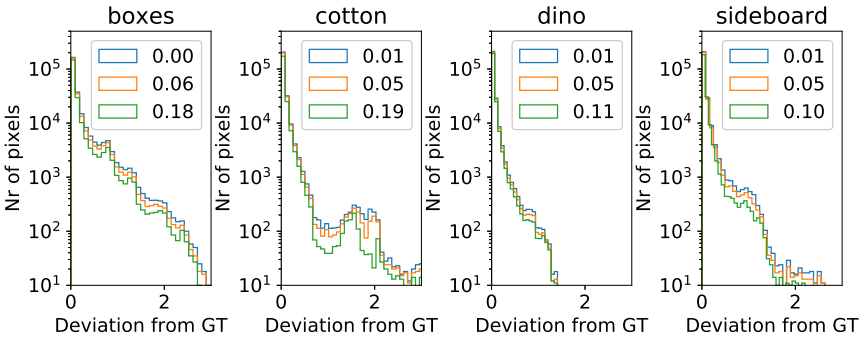

Figure B.4: The error histogram for different masks based on thresholding the new error map. The labelling indicates the fraction of masked out pixels.

C SGM parameter testing

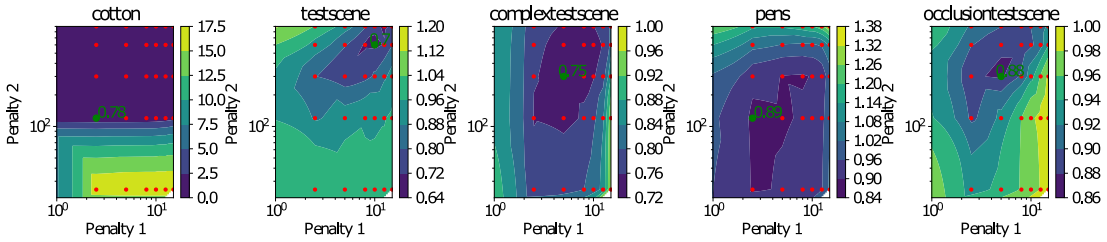

Figure C.1: Parameter grid test for semi global matching showing the MSE in dependence of 2 parameters from equation 2.40. The red points mark the grid points used to create the contour plot. The green dots show the minimum MSE, respectively.

The Semi-global matching algorithm is parameter-dependent. Using grid-search we try to find the best parameter based on different metrics. In the form of equation 2.40, it is dependent on 2 parameters. See figure C.1 regarding the related contour plot. The ideal parameter combination is quite scene-dependent. Even worse is finding a triplet of parameters when doing 3-D grid search to find the parameters of equation 2.47. In figure C.2, C.4 and C.3 different error metrics are plotted in function of all 3 parameters. If we take a look at the colourbar of the contour plots, we see that in a wide spectrum of parameters the results are almost the same.

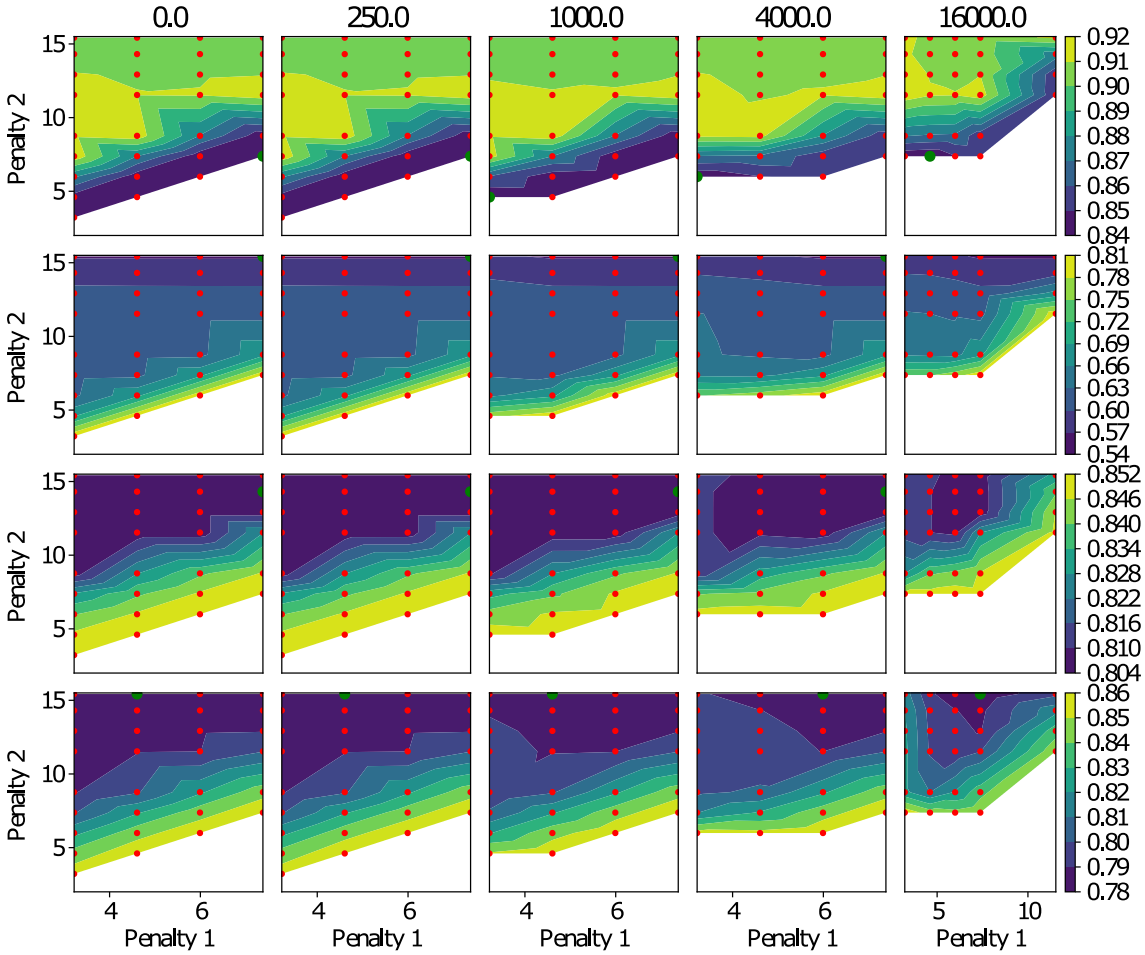

Figure C.2: Mean squared error for different (3) parameters. The red points mark the grid points used to create the contour plot. The green dots show the minimum MSE, respectively. From upper row to lower row: Scenes *boxes, cotton, dino, sideboard.*

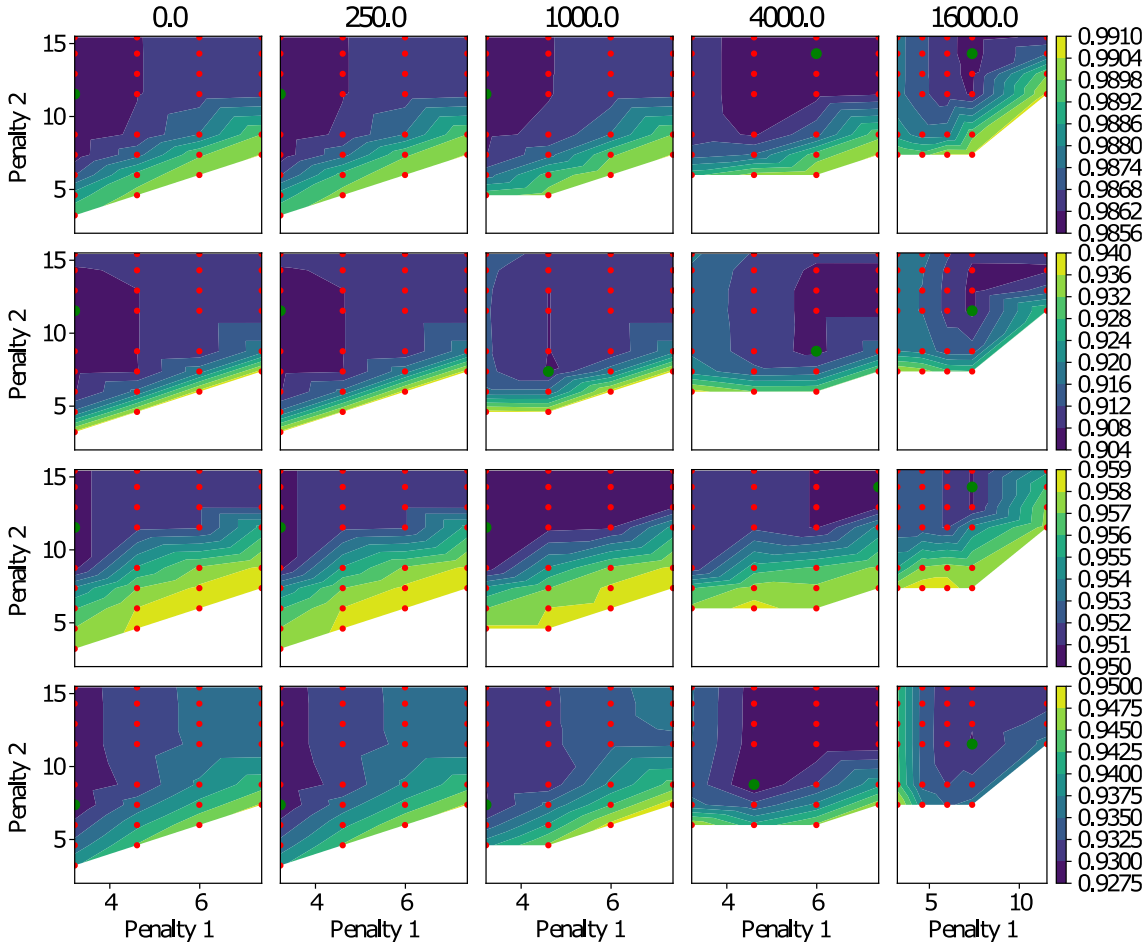

Figure C.3: Discontinuity score for different (3) parameters. The red points mark the grid points used to create the contour plot. The green dots show the minimum MSE, respectively. From upper row to lower row: Scenes *boxes, cotton, dino, sideboard.*

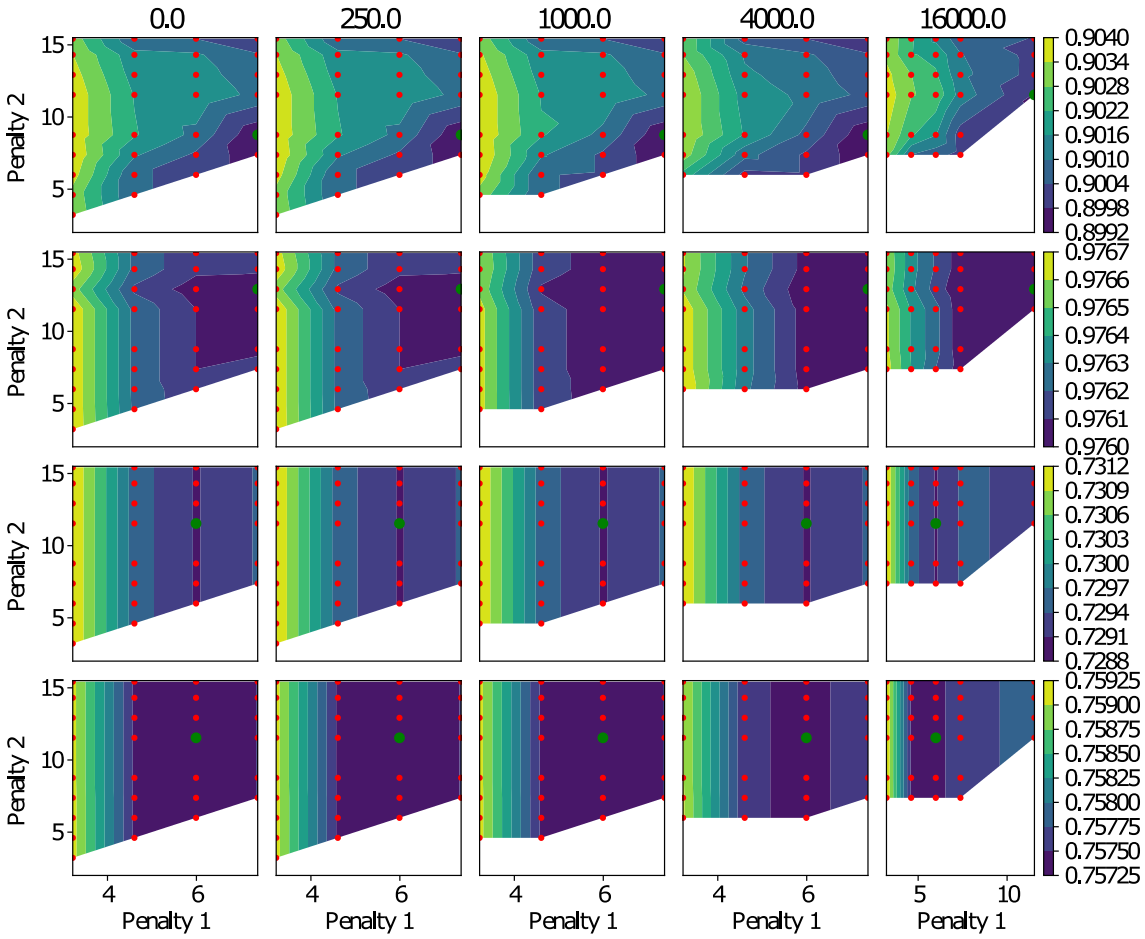

Figure C.4: Planar score for different (3) parameters. The red points mark the grid points used to create the contour plot. The green dots show the minimum MSE, respectively. From upper row to lower row: Scenes *boxes, cotton, dino, sideboard.*

D Real Data

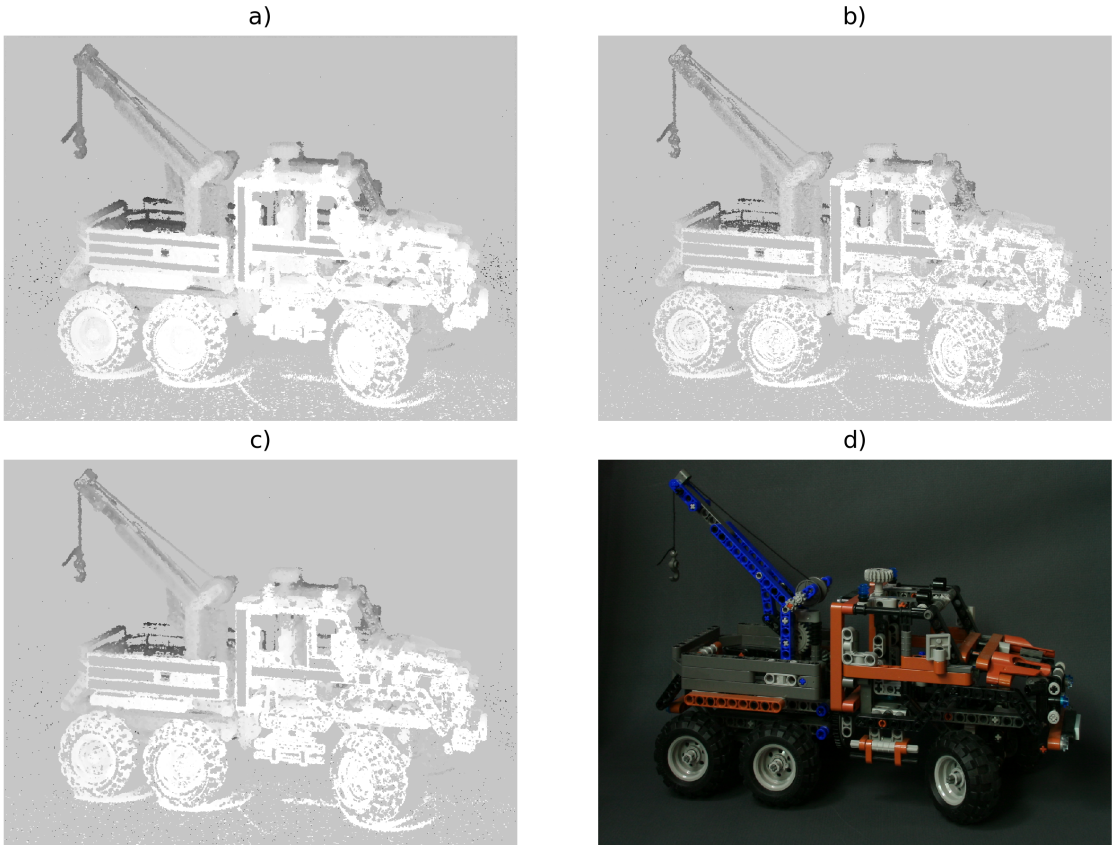

Figure D.1: a) Depth map prod. by the structure tensor pipeline, with minimal coherence set to 0.85. b) Depth map when thresholding gradients, minimal coherence set to 0.85. c) Depth map when thresholding gradients and using iterative SGM post-processing, minimal coherence set to 0.85. d) full centre view of the scene.

In figure D.1 the complete truck disparity map related to figure 3.39 is shown.

E CNN disparity estimation implementation

The approach of using machine learning for 3D reconstruction from light fields is a growing area in research nowadays. Our approach is based on supervised learning, using state-of-the-art convolutional neural net (CNN) architectures. We used a DenseNet CNN architecture that has been developed by Huang et al. [2017] and has been applied to various tasks in the field of computer vision, e.g. for 3D-object recognition and detection based on RGB-D data by Wolf [2018]. In particular, the same architecture from this work has been used, adapted such that the input format fits the format of the EPIs used. As input data we use the 9x512x3 RGB EPI data, where 9 view angles are used, 512 is the pixel size in the direction of the viewing positions and 3 describes the RGB-space. The output is a list with 512 entries, representing the depth values along the EPI.
As we did before, we use a crosshair light field input yielding EPIs in x- and y-direction. For the sake of simplicity, our CNN only handles EPIs, such that the merging of x-and y-direction is done without ML. In fact, semi-global matching is applied, taking both directional values as candidates and scanning through the diagonal paths in the image for the best fitting depth.
For training the network we used 24 synthetic scenes from the hci benchmark dataset, resulting in 512*24 EPIs for each direction, in total 24 576 EPI training sets.

Results

In figure E.1 the results of the trained CNN can be seen. In the lower left, the scene *pens* is depicted, which has been used for training of the Neural Net. It is expected to yield good results, though overfitting cannot be ruled out based on testing training data. In this scene, one can already see tearing effects along x- and y-direction. As the depth estimation happens row-(column-)wise, this effect is expected. In this example, the depth estimation based on x- and y-directional EPIs is done by simply taking the in-between value at each pixel. The synthetic testscene *herbs* in the lower right already shows stronger tearing effects. However, small details are captured surprisingly well. The real world scene *trucks* is depicted in the upper row of figure E.1, horizontal and vertical EPI depth map estimations are separated. Here the smearing is even worse. Instead of merging x- and y-direction by the in-between value of both depth estimations, we merge them using the SGM algorithm with 2 candidates at each point, which are given by the directional CNN depth estimations. In figure E.2 on the right the merged depth map is depicted in

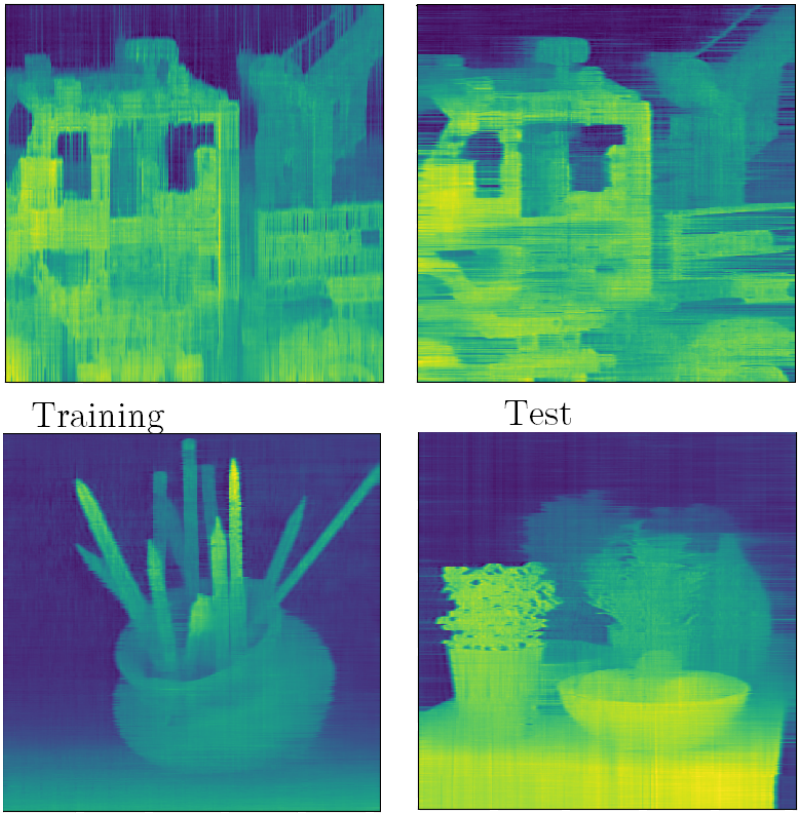

Figure E.1: Upper left: Depth map created from trained CNN using vertical EPIs. Upper right: Depth map created from trained CNN using horizontal EPIs. Lower left: Depth map from training dataset. Lower right: Depth map from a synthetic scene that is not part of the training data.

comparison to the ST pipeline result. Even though tearing is still visible, the result is a dense depth map showing a highly detailed depth map on textured areas, while big unicoloured areas are also solved well.
This side project is a proof-of-concept that reveals the power of CNN in light field depth reconstruction.

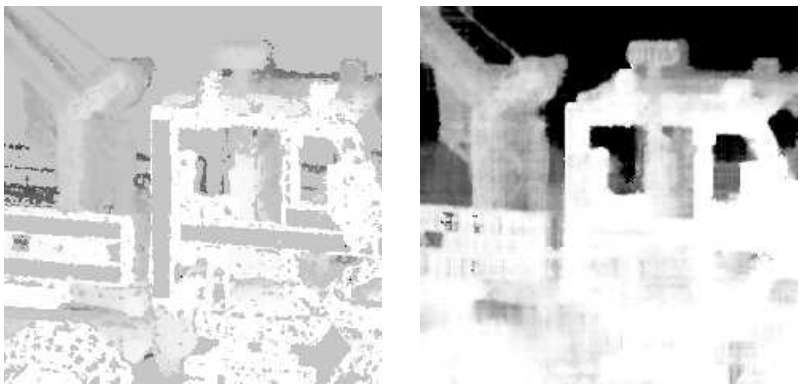

Figure E.2: Left: Clipping of the depth map of scene *truck*, produced by the structure tensor. Right: Merged depth map from the CNN, using SGM for merging.

F Lists

F.1 List of Figures

2.1 Two-plane parametrisation 11
2.2 Visualization of an EPI extraction 12
2.3 Example epipolar plane image 12
2.4 LumiPlus Scanner from HDVision Systems 14
2.5 Refocussed EPI 15
2.6 Occlusion in an EPI 16
2.7 Discontinuity evaluation 17
2.8 Sandglass kernel 18
2.9 Segmenting an EPI 20
2.10 Morphological closing 21
2.11 Enhanced segmentation mask 22
2.12 Alternative Merging: Parameter dependence 25
2.13 SGM Path explanation 27
2.14 SGM data to be storaged 29
2.15 Exemplary SGM path 29
2.16 SGM depth map from different directions 31
2.17 Example for post-processing smoothing 31
2.18 Iterative SGM: Convergence 32
2.19 Iterative SGM: Penalty function 34
2.20 Linear cost function vs. exponential cost function results 34

3.1 Pixel response for depth from focus techniques 38
3.2 Depth from focus: Depth maps 39
3.3 Mean squared error for depth-from-focus techniques 40
3.4 Depth from focus: depth maps with resolution 1 41
3.5 Mean squared error for depth-from-focus techniques 42
3.6 Speed of different depth-from-focus techniques 42
3.7 Photo-consistency pre-estimate for the structure tensor 43
3.8 Results with custom sized kernel 44
3.9 Bilateral filtering 45
3.10 Parameter dependency for bilateral filtering using EPI 45
3.11 Parameter dependency for bilateral filtering 46
3.12 Results of the bilateral filtering 47
3.13 Results when thresholding the gradients in the EPI 48
3.14 Disparity map zoom-ins for different methods 49

3.15 Varying kernel size . . . 50
3.16 Segmentation of EPI dependence on the morphology kernel . . . 50
3.17 Disparity map zoom-ins for different methods . . . 51
3.18 Comparing alternative coherence to new coherence . . . 52
3.19 Old coherence vs new error . . . 53
3.20 Dependence of error on cut out pixels . . . 54
3.21 Results from alternative merging of x- and y- direction . . . 55
3.22 Alternative merging: Parameter dependence . . . 55
3.23 Alternative merging: Parameter dependence . . . 56
3.24 Semi-global matching results . . . 57
3.25 SGM Coherence measure . . . 58
3.26 SGM as pure post-processing combined with alternative merging parameter dependence . . . 59
3.27 Alternative merging with SGM post-processing . . . 60
3.28 Alternative merging with SGM post-processing with thresholded gradients . . . 60
3.29 Parameter dependence for post-processing SGM using alternative merging+thresholded gradients. . . . 61
3.30 BadPixel parameter dependence for post-processing SGM using alternative merging+thresholded gradients. . . . 61
3.31 Discontinuity score parameter dependence for post-processing SGM using alternative merging+thresholded gradients. . . . 61
3.32 Iterative improvement of scene *cotton* . . . 62
3.33 Iterative SGM dependence on number of iterations . . . 63
3.34 Iterative SGM dependence on initial Δ with 5 iterations . . . 63
3.35 Iterative SGM dependence on initial Δ with 10 iterations . . . 64
3.36 Iterative SGM dependence on smoothing parameter . . . 64
3.37 Iterative SGM: depth maps . . . 65
3.38 Silicium scan with iterative SGM . . . 66
3.39 Truck clipping with iterative SGM . . . 67

B.1 Histogram when masking errors with new error map . . . 74
B.2 Histogram when masking errors with new error map . . . 75
B.3 Histogram when masking errors . . . 75
B.4 Histogram when masking errors with new error map . . . 75

C.1 Parameter grid test for semi global matching . . . 76
C.2 Mean squared error for different (3) parameters . . . 77
C.3 Discontinuity score for different (3) parameters . . . 78
C.4 Planar score for different (3) parameters . . . 79

D.1 Truck with iterative SGM . . . 80

E.1 CNN results . . . 82
E.2 Truck with CNN +SGM . . . 83

F.2 List of Tables

3.1 Depth-from-focus: Time for pipelines 43
3.2 Comparison of runtime and MSE of different methods. 69

G Bibliography

[1] Hci benchmark dataset. http://hci-lightfield.iwr.uni-heidelberg.de/benchmark/. Accessed: 2018-010-16.

[2] Website of the hypermedia image processing. http://homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm. Accessed: 2018-04-16.

[3] Website of the iwr heidelberg. https://klimt.iwr.uni-heidelberg.de/Staff/swanner/. Accessed: 2018-04-06.

[4] what-when-how.com. http://what-when-how.com/introduction-to-video-and-image-processing/morphology-introduction-to-video-and-image-processing-part-2/. Accessed: 2018-04-16.

[5] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[6] E. H. Adelson, J. R. Bergen, et al. The plenoptic function and the elements of early vision. 1991.

[7] J. Bigun. Optimal orientation detection of linear symmetry, 1987.

[8] Blender. Blender—a 3d modelling and rendering package, 2014.

[9] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International journal of computer vision*, 1(1):7–55, 1987.

[10] Y. Y. I. S. K. S. J. K. Changha Shin, Hae-Gon Jeon. Epinet: A fully-convolutional neural network using epipolar geometry for depth from light field images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[11] M. Diebold. *Light-Field Imaging and Heterogeneous Light Fields*. PhD thesis, 2016.

[12] I. Ernst and H. Hirschmüller. Mutual information based semi-global stereo matching on the gpu. In *International Symposium on Visual Computing*, pages 228–239. Springer, 2008.

[13] B. Freist. 3d reconstruction in lightfields, 2018.

[14] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814. IEEE, 2005.

[15] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30 (2):328–341, 2008.

[16] H. Hirschmüller. Semi-global matching-motivation, developments and applications. *Photogrammetric Week 11*, pages 173–184, 2011.

[17] K. Honauer, O. Johannsen, D. Kondermann, and B. Goldluecke. A dataset and evaluation methodology for depth estimation on 4d light fields. In *Asian Conference on Computer Vision*. Springer, 2016.

[18] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017.

[19] B. Jähne. *Digitale Bildverarbeitung*. Springer-Verlag, 2013.

[20] Y. Luo, W. Zhou, J. Fang, L. Liang, H. Zhang, and G. Dai. Epi-patch based convolutional neural network for depth estimation on 4d light field. In *International Conference on Neural Information Processing*, pages 642–652. Springer, 2017.

[21] Z. Moratto. Sgm implementation. http://lunokhod.org/?p=1403, 2013. Accessed: 2018-10-13.

[22] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan. Light field photography with a hand-held plenoptic camera.

[23] Schönpflug. Optimizing depth maps and point clouds from light field imaging, 2017.

[24] H. Sheng, P. Zhao, S. Zhang, J. Zhang, and D. Yang. Occlusion-aware depth estimation for light field using multi-orientation epis. *Pattern Recognition*, 74: 587–599, 2018.

[25] M. W. Tao, P. P. Srinivasan, S. Hadap, S. Rusinkiewicz, J. Malik, and R. Ramamoorthi. Shape estimation from shading, defocus, and correspondence using light-field angular coherence. *IEEE transactions on pattern analysis and machine intelligence*, 39(3):546–560, 2017.

[26] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998.

[27] S. Wanner. *Orientation Analysis in 4D Light Fields*. PhD thesis, 2014.

[28] M. Watanabe and S. K. Nayar. Rational filters for passive depth from defocus. *International Journal of Computer Vision*, 27(3):203–225, 1998.

[29] B. Wiburn. *High performance imaging using arrays of inexpensive cameras*. PhD thesis, Citeseer, 2004.

[30] C. Wolf. 3d-object recognition and detection with convolutional neural networks, 2018.

[31] G. Wu, B. Masia, A. Jarabo, Y. Zhang, L. Wang, Q. Dai, T. Chai, and Y. Liu. Light field image processing: An overview. *IEEE Journal of Selected Topics in Signal Processing*, 11(7):926–954, 2017.

[32] Z. Xu, J. Ke, and E. Y. Lam. High-resolution lightfield photography using two masks. *Opt. Express*, 20(10):10971–10983, May 2012. doi: 10.1364/OE.20.010971. URL http://www.opticsexpress.org/abstract.cfm?URI=oe-20-10-10971.

Erklärung:

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den (Datum) .....