

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 EPI-basierten Lichtfeldmessungen:

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 entwickelter Algorithmen getestet, weiterhin werden Depth-of-Defocus-Techniken getestet und verglichen. Zusätzlich wird ein Algorithmus basierend auf dem in der StereoVision verbreiteten SemiGlobal Matching Algorithmus (SGM) als Nachbearbeitungsschritt implementiert.

Optimization of EPI-based light-field 3D reconstructions:

This work deas 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 will be developed and tested, further depth-of-defocus techniques will be tested and compared. In addition, an algorithm based on the SemiGlobal Matching Algorithm (SGM) ,commonly used in Stereo vision, is implemented as a postprocessing step.

Contents

1	Introduction	8
2	Theory	9
2.1	Light Field Parametrization	9
2.1.1	The Lumigraph	9
2.1.2	Epipolar Plane Images	10
2.2	Depth from Structure Tensor	11
2.2.1	Implementation	12
2.2.2	The occlusion problem of the structure Tensor	15
2.3	Modifications on the structure tensor	15
2.3.1	Kernel sizes	15
2.3.2	Colour-awareness with bilateral filtering	18
2.3.3	Thresholding the gradients	18
2.3.4	Occlusion-awareness using segmentation of the EPI	19
2.3.5	An alternative to the coherence as the confidence measure	20
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	31
2.5.4	SGM as post-processing smoothing	31
2.5.5	Iterative post-processing	31
2.5.6	A new penalty function	32
2.5.7	Semi-global Matching implementation	34
2.6	Metrics for the evaluation of depth maps	35
2.6.1	dense and sparse maps	35
2.6.2	mean squared error	35
2.6.3	bad pixel score	35
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	38

3.2	Modification on the Structure Tensor	45
3.2.1	Sandglass kernel	45
3.2.2	Bilateral filter	45
3.2.3	Thresholding Gradients	48
3.2.4	Occlusion awareness segmentation	51
3.2.5	Alternative coherence	53
3.2.6	Alternative merging of x- and y-direction	56
3.3	Semi-Global Matching	63
3.3.1	Iterative SGM post-processing	68
3.3.2	Application on real data	72
3.4	Comparison of Results	73
4	Resumee and Outlook	77
A	Derivation of the structure tensor	80
B	Real Data	81
C	CNN Disparity Estimation implementation	83
D	Lists	85
D.1	List of Figures	85
D.2	List of Tables	86
E	Bibliography	87

1 Introduction

In the field of computer vision one big challenge is the *3D Reconstruction* based on 2D Images. This is reasoned by the sheer amount of different applications, reaching from industrial quality management to creations of CGI effects as well as 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**.

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, but also enable making statements about the surface characteristics. The real-time processing of a sufficient amount of 2D data however still remains a challenge. Further reconstructing the 3D image from 2D images tasks 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 lead makes 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.

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

$$P(V_x, V_y, V_z, \theta, \phi, \lambda), \quad (2.1)$$

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 pixel of a pinhole camera picture as the coordinate system of our choice, the plenoptic function would be parametrized as

$$P(x, y, V_x, V_y, V_z, \lambda). \quad (2.2)$$

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 [24].

2.1.1 The Lumigraph

In the form of 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 due to the fact, that light rays that are 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

$$L : \Omega \times \Pi \rightarrow R \quad x, y, s, t \rightarrow L(x, y, s, t), \quad (2.3)$$

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.

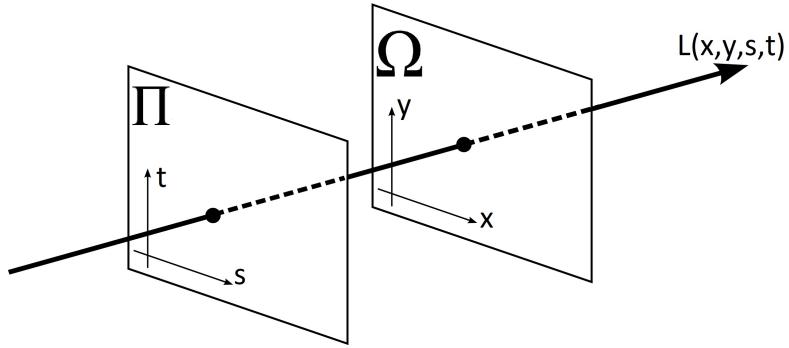


Figure 2.1: In a 4-dimensional two-plane parametrisation a light ray is characterized by the intersection with two parallel planes. we refer to the plane Π as the camera plane and the plane Ω as the image plane. From Xu et al. [2012]

2.1.2 Epipolar Plane Images

We measure the light field in order to obtain as much information about the scene we're looking at as possible. In order to obtain the three-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 of the line at each point we obtain the distance from the camera plane with the equation

$$\text{distance} = \frac{f \cdot b}{\Delta}, \quad (2.4)$$

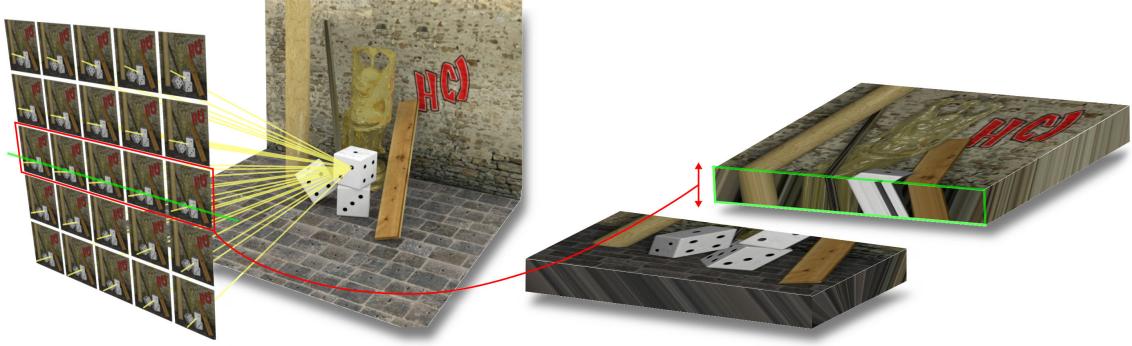


Figure 2.2: Visualization of an Epipolar Plane Image extractoin: A camera array takes images of the same scene from slightly different angles (left array). For a fixed image coordinate y^* (green) and a fixed camera coordinate t^* (red) the pixels are extracted and stacked up resulting in an EPI Σ_{y^*,t^*}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



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 . Marked in red one can identify the centre position of the camera viewpoints.

where f is the focal length of the camera and b is the baseline between the camera positions. One may notices 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.

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 paper „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 $g : \Omega \rightarrow R, \Omega \subset R^2$ is defined as

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

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:

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

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]:

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

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. 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

vertical view \rightsquigarrow Nr of rows in image coordinates \times Nr of cameras

horizontal view \rightsquigarrow Nr of columns in image coordinates \times Nr of cameras

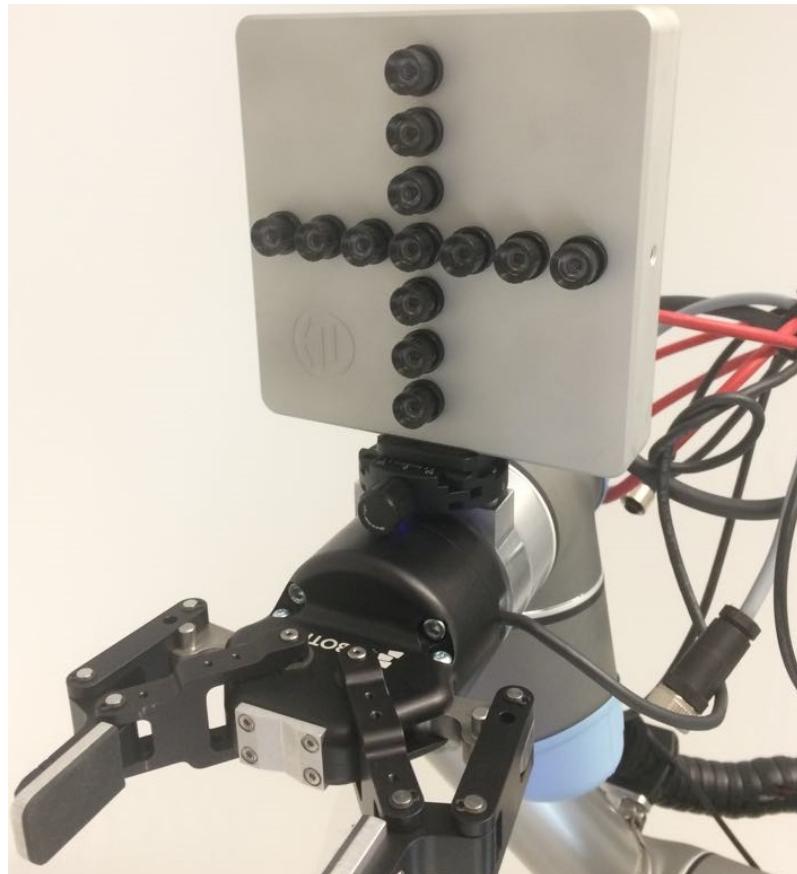


Figure 2.4: Instead of a complete camera array a crosshair camera reduces the number of cameras to one row and one column for faster processing. This *Lumi+* - Scanner from HDVision Systems is mounted d an a robot arm for industrial applications.

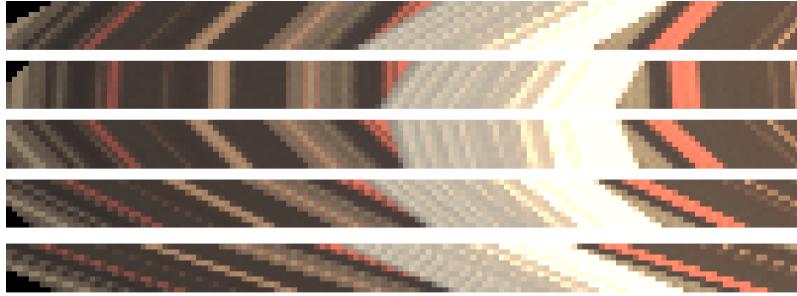


Figure 2.5: The EPI is refocussed by integer disparity steps. If the slope at a scene point is zero (vertical line) the EPI is perfectly focussed on that point. An integration of all views would still result in a sharp image at the corresponding depth.

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 obtain reasonable results when calculating the gradient in the next step. 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

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

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

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 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.

From all disparity shifts the value with the highest coherence measure (equation 2.6) is selected. 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

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

2.2.2 The occlusion problem of the structure Tensor

Having a look at the results of the 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 centre view pixels close to the edge of a depth discontinuity are at least partly occluded, this behaviour is to be 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. However, 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.

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, however the accuracy at edges in the scene will rapidly decrease.

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

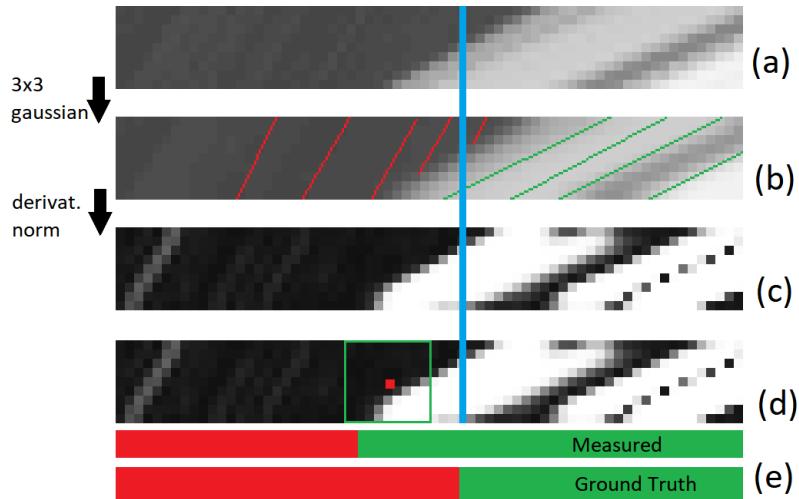


Figure 2.6: (a) The structure of an occlusion border in a grey-value EPI. (b) Before calculating the gradient, the EPI is smoothed with a 3×3 Gaussian kernel. One sees the smoothed EPI with coloured lines indicating the Ground Truth orientation. (c) shows the norm of the gradient calculated via the Scharr-filter. 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.

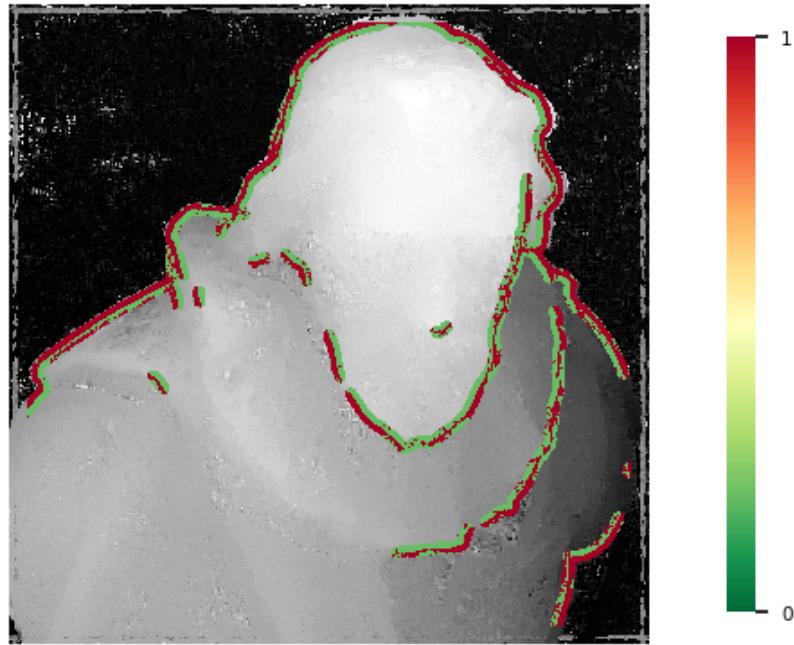


Figure 2.7: Evaluation of the deviation from Ground truth at the depth discontinuity for scene "cotton". The red border indicates that the depth map is erroneous at the outside of the edge.

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.

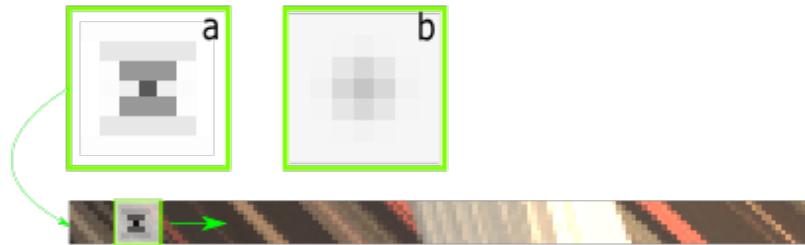


Figure 2.8: (a) shows a custom-shaped kernel in form of a sandglass. (b) shows a normal Gaussian kernel.

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 [in the EPI](#) are very likely to belong to the same orientation, in theory one only has to find all points with the same colour which then lie on the same line in the EPI. 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. Though 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

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

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

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

Nevertheless in 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. The only 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

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

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

$$(2.15)$$

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 the 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 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 that is segmenting occlusion/non-occlusion areas, however this approach does not need to calculate the disparity values beforehand and can be implemented in the work flow 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 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 resulting mask from the binary thresholding (e) using morphological image-processing operators as explained in detail [in hom.](#) It is made use of Morphological Closing (structured filling in of image region boundary pixels), which is explained in figure 2.10.

Having a look at the different scenarios in the given EPI that can happen when segmenting, the necessity for morphological closing becomes clear. 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.

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

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

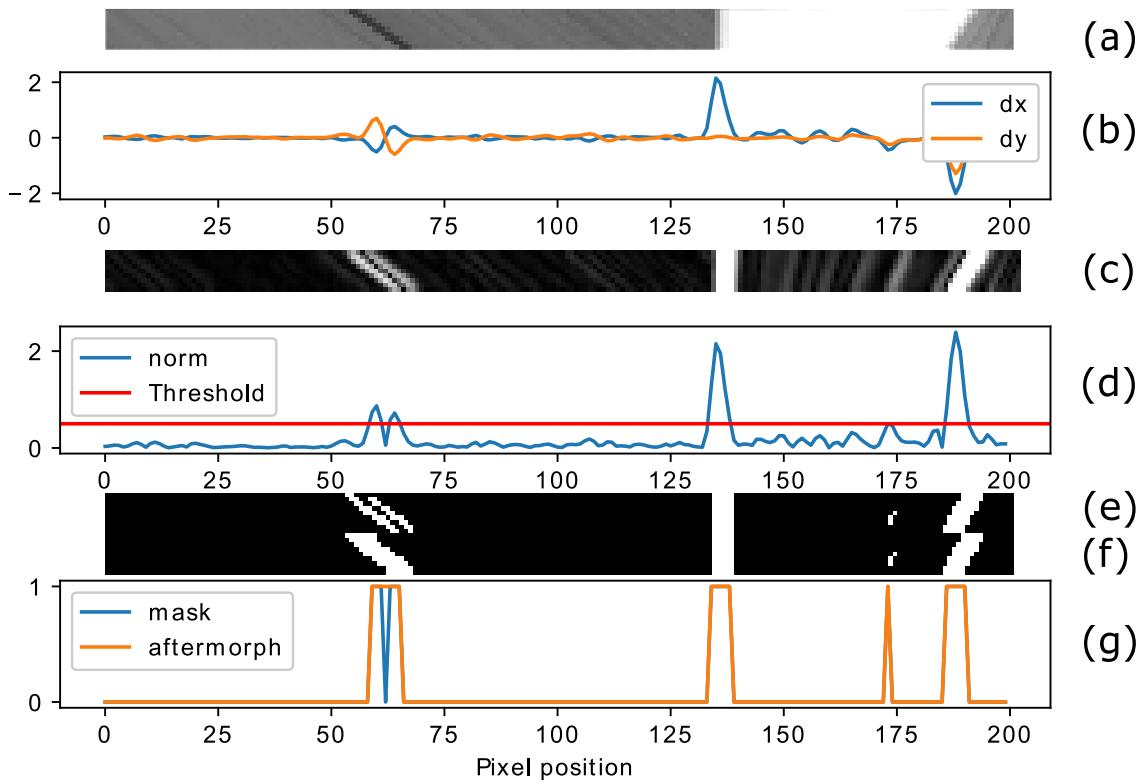


Figure 2.9: (a) Input EPI to be segmented. (b) show the local derivatives along the centre view line. (c) shows the vector norm of the derivative. (d) shows the centre view norm with the threshold that is applied to mask transitions. (e) shows the resulting mask. (f) shows the improved mask using morphological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

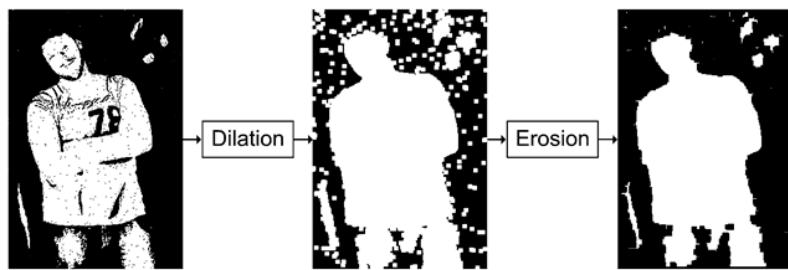


Figure 2.10: Morphological closing is a combination of Dilation and Erosion. Dilation uses a custom-sized kernel and turns any 0 to 1, if at least one 1 is found in the local environment. Erosion turns any 1 to 0, if at least one 0 is found in the local environment. Image from [wha](#)

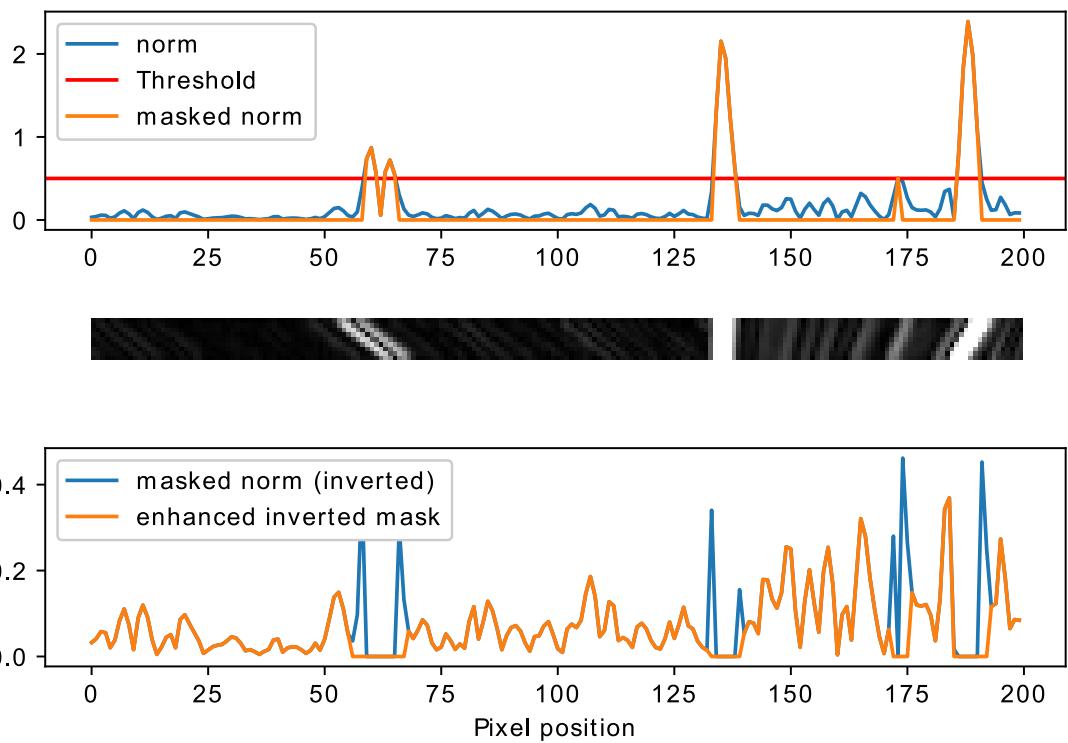


Figure 2.11: In the upper diagram the thresholded gradient norm of the EPI (middle) is seen. The norm (masked with the inverted mask) is depicted in the lower figure (blue), while the inverted mask with a dilation filter is depicted in orange.

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

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

When substituting

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

we obtain

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

With

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

inserting in 2.17:

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

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

$$\Delta J_{11} = G * |2\nabla_x| \Delta \nabla_x \quad (2.22)$$

$$\Delta J_{22} = G * |2\nabla_y| \Delta \nabla_y \quad (2.23)$$

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

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

$$\Delta \nabla_x = \sqrt{E[\nabla_x^2] - E[\nabla_x]^2} \quad (2.25)$$

$$\Delta \nabla_y = \sqrt{E[\nabla_y^2] - E[\nabla_y]^2} \quad (2.26)$$

with the expectation value defined as the Gaussian weighted mean:

$$E[I] = G * I \quad (2.27)$$

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:

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

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

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

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

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

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

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

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

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.

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. In our case we are working with a crosshair light field only providing two EPIS for each scene point. 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 such optimization scheme. He thus sticks to choose the value with higher coherence. Instead one simply chooses the disparity value which relates to the a larger depth,

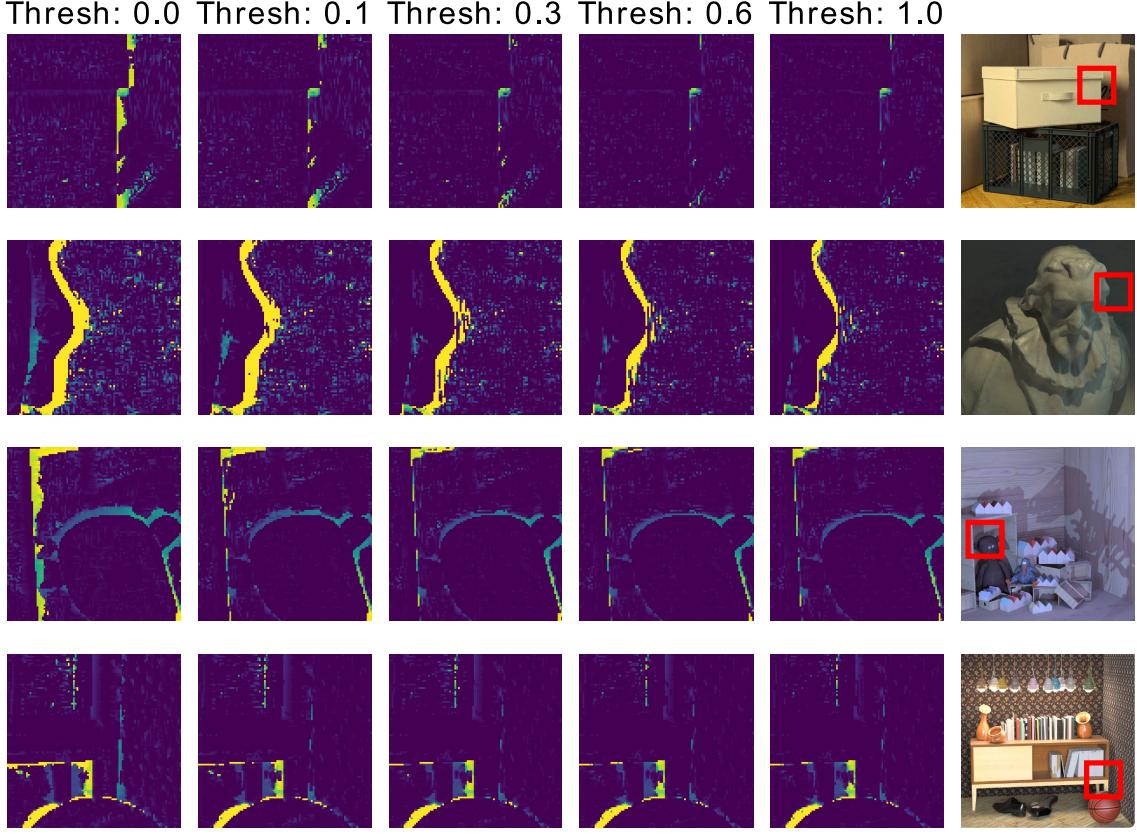


Figure 2.12: From left to right: Parameter dependent alternative merging. Up to down: scenes *boxes*, *cotton*, *dino*, *sideboard*.

and only chooses the closer disparity value, if the coherence difference is big enough. Therefore we introduce a threshold:

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

Figure 2.12 shows the effect of varying the threshold introduced in equation 2.34. The occlusion edges shrink significantly with a higher **threshold**.

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 [20] :

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

where d' describes the relative pixel shift. The disparity is directly related to the absolute depth of the focus (relate to PICTURE) 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

$$Z = \frac{f \cdot b}{d}. \quad (2.36)$$

We obtain

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

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 straight forward 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 to 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 as *photo consistency*; for more information read Tao et al. [2017]. The response value at a given depth is obtained from

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

where $P(x, y)$ is the centre view. For more robustness, it is averaged over a small window. We refer to this measuring technique as *photo consistency* in the following. Note that calculating the absolute 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:

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

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 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 in 2005 by Heiko Hirschmüller (Hirschmüller [2005]) that was described as „a very good trade off between runtime and accuracy“ (Hirschmüller [2011]): we speak of Semi-Global Matching.

In general, matching 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

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

The first 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 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. However, minimizing the global cost function involves computational cumbersome algorithms as it is a 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

$$L'_r(\vec{p}, D) = C(\vec{p}, D) + \min \begin{cases} L'_r(p_{\text{before}}, D) \\ L'_r(p_{\text{before}}, D + 1) + P1 \\ L'_r(p_{\text{before}}, D - 1) + P1 \\ \min_i L'_r(p_{\text{before}}, i) + P2 \end{cases} \quad (2.41)$$

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. It is to mention that the rolling sum can reach quite high numbers that are unpleasant to handle on the computer; a normalization is implemented by subtracting $\min_D L'_r(p_{\text{before}}, D)$ 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.

Summing along at least 8 path directions (crosshair + diagonals) results in disparity maps with reduced error pixel while maintaining clean edges.

2.5.2 Semi - Global Matching for light fields

Even though Hirschmüller describes Semi - Global Matching (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 those two methods:

1. The structure tensor algorithm is tuned to a much smaller disparity range. While in Hirschmüller [2005] Hirschmüller 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 2.13 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.

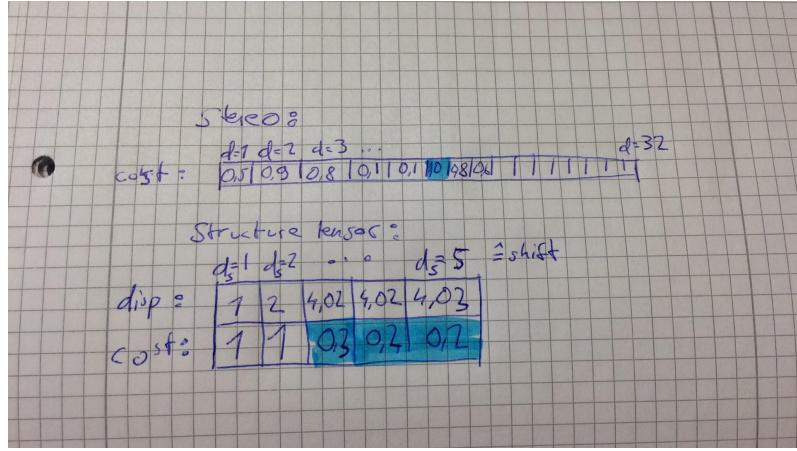


Figure 2.13: One point \vec{p} contains different values (values here: example values): For Stereo matching, the resolution is given by the discrete disparity steps. Each disparity value has a cost value assigned to it. Using the ST, we have a different sub-pixel accuracy for every disparity shift, while the sub-pixel accuracy can differ from the shift by up to 1.2

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 2.13, we have to process an 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 as depicted in figure 2.14 requires a new definition of the error propagation defined in equation 2.40.

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:

$$E(d) = \sum_{\vec{p}} \left(C(\vec{p}, d_{\vec{p}}) + \sum_{q \in N_p} \begin{cases} P1 \cdot |d_{\vec{p}} - d_q| & \text{if } |d_{\vec{p}} - d_q| \leq 1 \\ P2 & \text{if } |d_{\vec{p}} - d_q| > 1 \end{cases} \right). \quad (2.42)$$

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

$$L'_r(\vec{p}, s) = C(\vec{p}, s) + \min_i \begin{cases} L'_r(\vec{p}_{\text{before}}, s_i) + P1 \cdot |d_{\vec{p},s} - d_{\vec{p}_{\text{before}},s_i}| & \text{if } |d_{\vec{p},s} - d_{\vec{p}_{\text{before}},s_i}| \leq 1 \\ L'_r(\vec{p}_{\text{before}}, s_i) + P2 & \text{if } |d_{\vec{p},s} - d_{\vec{p}_{\text{before}},s_i}| > 1 \end{cases} \quad (2.43)$$

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

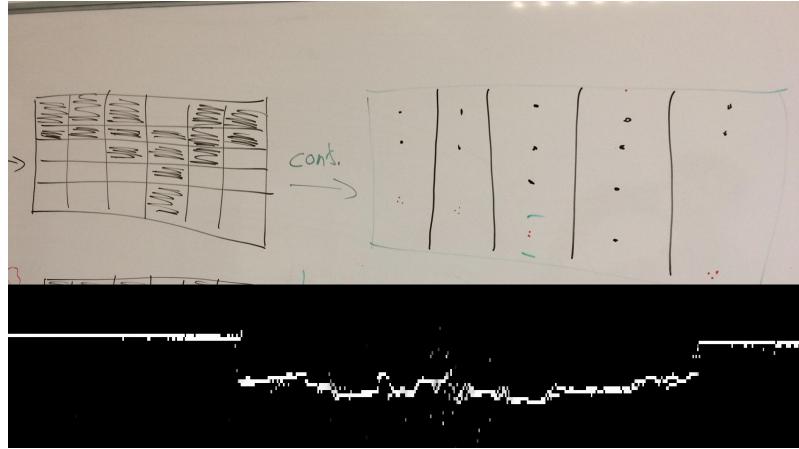


Figure 2.14: In this figure one can see the sketch of one arbitrary scanline of the structure tensor method. Under the Assumption that the ST algorithm recognizes the structure of the EPI perfectly, we either have two or three disparity shifts that have a high coherence (coloured white) (a) and result in approximately the same final disparity. This can be seen if we plot the exact disparity values in a continuous space(b). In (c) real data scanline cost is plotted with a resolution of ca. 100 pixels.

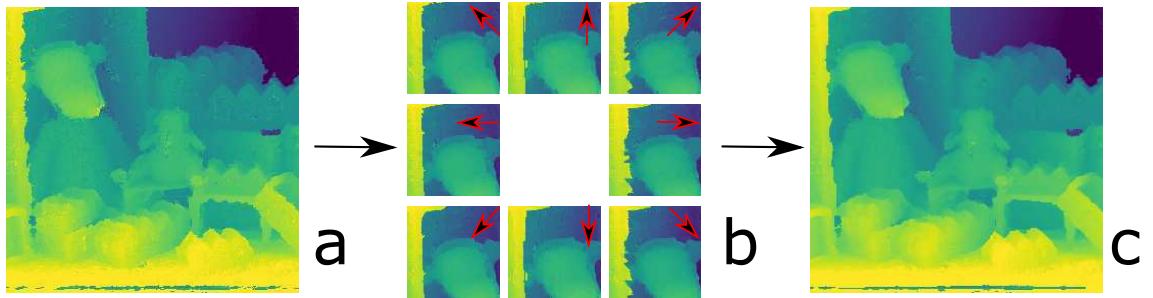


Figure 2.15: a: scene *dino* using the old pipeline,

$(|d_{\vec{p},s} - d_{\vec{p}_{\text{before}},s_i}| \leq 1)$ 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 in our implementation, $P2$ is modified by

$$P2' = \frac{P2}{\sqrt{(Im_b^2 + Im_r^2 + Im_g^2)}}, \quad (2.44)$$

with Im being the centre view of the light field and $Im_{b,g,r}$ being the 3 colour channels. If the colour intensity changes, the penalty for a disparity discontinuity is lowered.

2.5.3 Occlusion awareness in SGM for light Fields

If we take a close look at figure 2.14, one can see that at least at some discontinuities the ST pipeline manages to calculate the depth of the background structure near boundaries with good coherence, but the foreground structure is overlapping and quantitatively measured with higher coherence. Once we know that at least at some edges an improvement can be made by adapting the evaluation function in a sense that the highest coherence does not necessarily measure the right depth, we realize that SGM is doing the job already. The simple heuristic approach is to change the global minimization function 2.42 such that a positive disparity jump is less punished than a negative one. In fact, the function changes to

$$E(d) = \sum_{\vec{p}} \left(C(\vec{p}, d_{\vec{p}}) + \sum_{q \in N_p} \begin{cases} P1 \cdot |d_{\vec{p}} - d_{\vec{q}}| & \text{if } |d_{\vec{p}} - d_{\vec{q}}| \leq 1 \\ P2 & \text{if } d_{\vec{p}} - d_{\vec{q}} > 1 \\ P3 & \text{if } d_{\vec{p}} - d_{\vec{q}} < -1 \end{cases} \right). \quad (2.45)$$

2.5.4 SGM as post-processing smoothing

In general Semi-Global Matching refers to minimizing an energy function in one dimension for different paths in the given space. In case of disparity mapping, the disparity candidate values and its initial cost values have to be given. To use SGM as an efficient tool for smoothing, these disparity candidate values have to be chosen wisely. In 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 with any algorithm. In this sense the SGM is used as a pure post-processing step that can be used to improve any depth map. The effects of smoothing the disparity map while maintaining edges is depicted in figure 2.16 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 ? that uses graph algorithms 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 lowest cost is chosen and new candidates are formed for the next iteration. To reach convergence, this 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 2.17 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, setting the same Δ at each pixel leads to problems along any SGM scanline, as sketched in figure 2.17 b). Since all candidates are initialized with the same Δ one finds 3

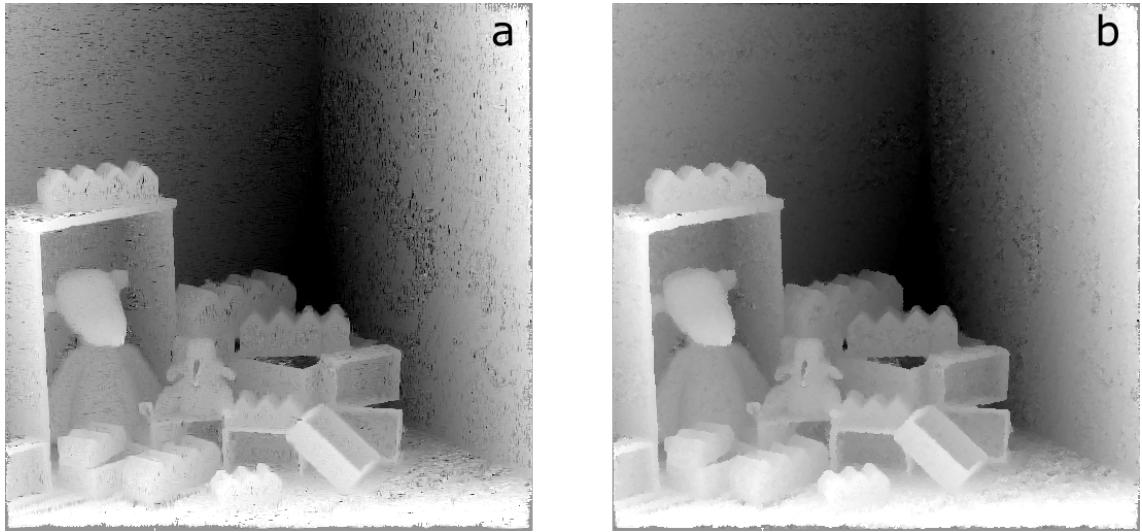


Figure 2.16: a: erroneous disparity map before smoothing b: Same disparity map after smoothing

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 2.17 c) only one surface line will have a low cost value. However doing this iteration 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 2.42, 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 in 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

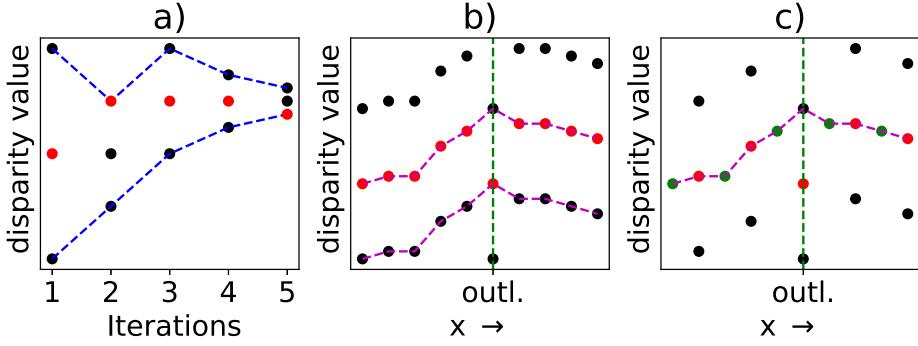


Figure 2.17: a) One exemplary pixel value in function of iteration steps. At each iteration, 3 candidate values for each pixel are chosen, the one with the lowest score is marked red. The candidate value window is marked with blue lines. b) One exemplary scanline through the depth map. Red marks the 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.

minimized then looks like

$$E(d) = \sum_{\vec{p}} \left(C(\vec{p}, D_{\vec{p}}) + \sum_{q \in N_p} P2 \cdot \left(1 - \exp \left(\frac{-|d_{\vec{p}} - d_{\vec{q}}|}{P1} \right) \right) \right). \quad (2.46)$$

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

$$E(d) = \sum_{\vec{p}} \sum_{q \in N_p} P2 \cdot \left(1 - \exp \left(\frac{-|d_{\vec{p}} - d_{\vec{q}}|}{P1} \right) \right) \quad (2.47)$$

with the 1D recursive form

$$L'_r(\vec{p}, s) = \min_i \left(L'_r(\vec{p}_{\text{before}}, s_i) + P2 \cdot \left(1 - \exp \left(\frac{-|d_{\vec{p}, s} - d_{\vec{p}_{\text{before}}, s_i}|}{P1} \right) \right) \right). \quad (2.48)$$

The equation 2.48 is plotted in figure 2.18 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 it is like before, see equation 2.44. Large image gradients are more likely to lie on a disparity discontinuity, therefore the error is reduced. In figure 2.19 the effect of different penalty functions is depicted. Figure 2.19 a) shows the result of the scene **cotton** after smoothing with the exponential penalty function 2.48, while figure 2.19 b) shows the disparity map after iterative SGM processing with a linear penalty function. One sees that the linear function

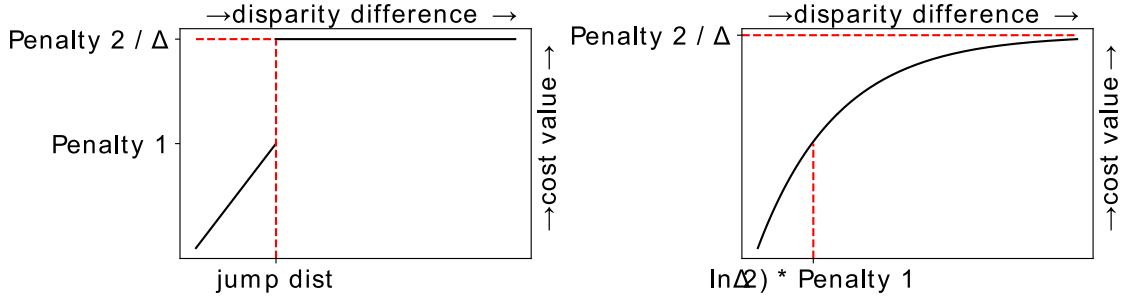


Figure 2.18: Sketch of different proposed Penalty functions. Left: Function as described in equation 2.43. Right function as described in 2.48.

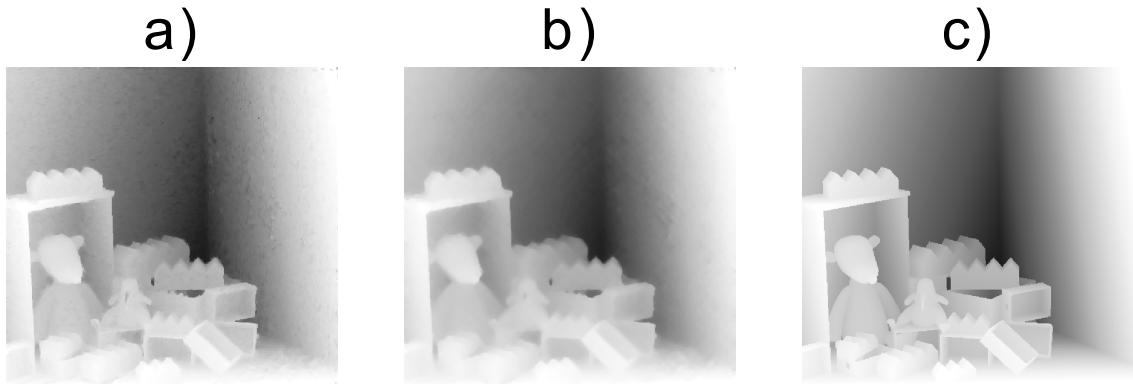


Figure 2.19: a) Depth map after Iterative SGM smoothing using an exponential penalty function. b) Instead a linear penalty function is used. c) Ground Truth

leads to smoother surfaces, however the smoothness constraint also oversmoothes the edges in the image. In a) the the edges are maintained sharp. The figure 2.19 c) shows the ground truth for comparison.

2.5.7 Semi-global Matching implementation

The implementation for using Semi-global matching in our work is oriented on the open source implementation published by Moratto. This code however uses only 8 instead of 16 paths along the image, further it struggles with time efficiency as it is about 20 times slower then the implementation of Hirschmüller [2005] as published in his paper. Our implementation thus will also be slow. 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 on parallel, a GPU implementation can be implemented in a straight forward manner. However GPU programming exceeds the boundaries of this work and will be part of future research and development. Instead, we focus on the feasibility of implementing Semi-global Matching 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 maps

The synthetic light field datasets are designed with Blender (Community [2014]) and provide photos from the scene in a crosshair pattern, similar to the camera configuration in figure 2.4. This allows to calculate a disparity map from only one point of view due to the restricted baseline of the cameras. 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

$$\text{MSE} = \frac{\sum_x (d(x) - gt(x))^2}{\text{Nr of Pixels}} \cdot 100 \quad (2.49)$$

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 bad pixel 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

$$\text{BadPixel}(t) = \frac{\{|d(x) - gt(x)| > t\}}{\text{Nr of Pixels}}. \quad (2.50)$$

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 missalignments and offsets:

$$\text{Bumpiness} = \frac{\min(0.05, \|H_f(x)\|_F)}{\text{Nr of Pixels}} \cdot 100 \quad (2.51)$$

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 challenges (planar surfaces or discontinuities) that are set by the scene. Given a mask M the BadPixel Score is then given as

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

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:

$$MRE = \frac{\sum_x (d(x) - gt(x))^2}{(\text{max. disp} - \text{min. disp}) * \text{Nr of Pixels}} \quad (2.53)$$

3 Evaluation

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 *consistent*, meaning that the number of pixels with low confidence should be the lowest possible.
2. It should result in a *fast* measure, ideally faster than 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:

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

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)

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

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

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

The directional gradients are simply added up to the response value

$$D'(x, y) = |G_x(x, y)| + |G_y(x, y)| \quad (3.4)$$

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

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

In the following we are going to compare the method 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 2 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 one at the right position. Having a look at the actual disparity maps 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 indicates that for all scenes either the Photo consistency or the Angular Correspondence method obtain the lowest mean relative error.

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

The 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. All disparity maps with integer disparity step size are compared to the ground truth in figure 3.4. Again the Angular Correspondence and Photo consistency measures result in the lowest MRE (figure 3.5), though the Angular Correspondence cue is about 1.5 times more time-consuming than the photo consistency measure under the same conditions (figure 3.6). Further testing 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

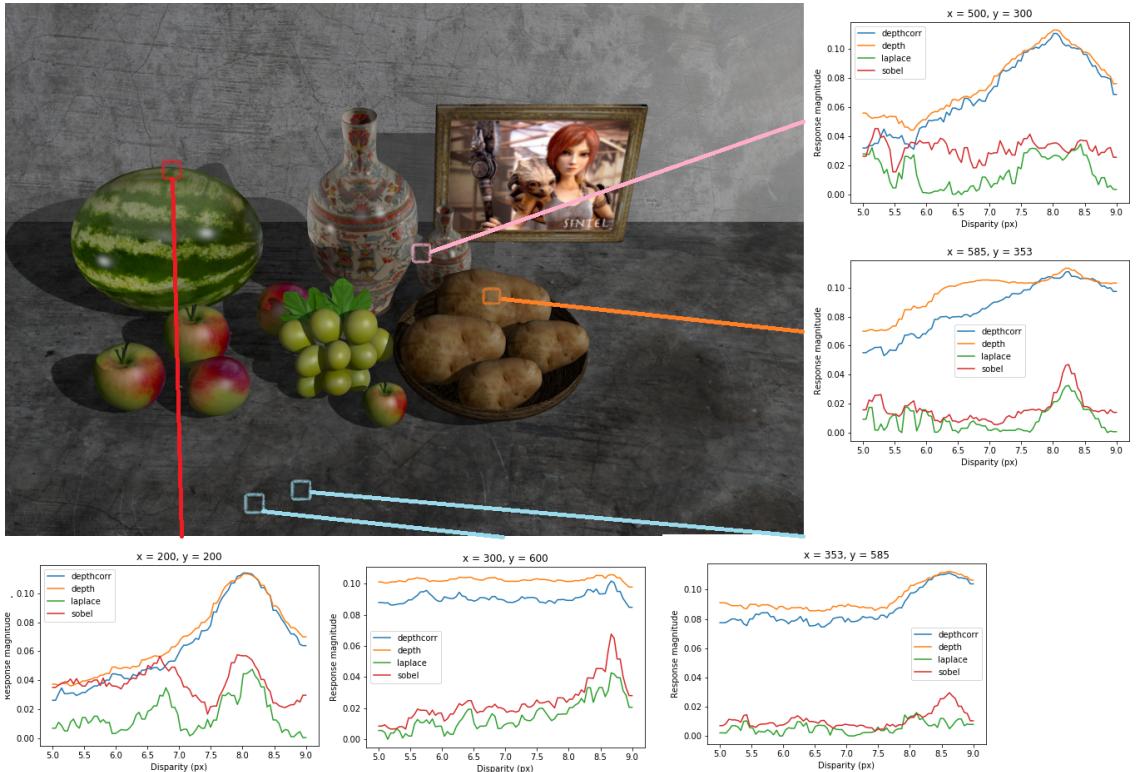


Figure 3.1: At different pixel positions we take a look on how the Pixel response value behaves for the four different depth-from-focus techniques: Photo consistency (orange), Angular correspondence (blue), First derivative (red) and The Laplace method (green) A high value means high confidence (low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

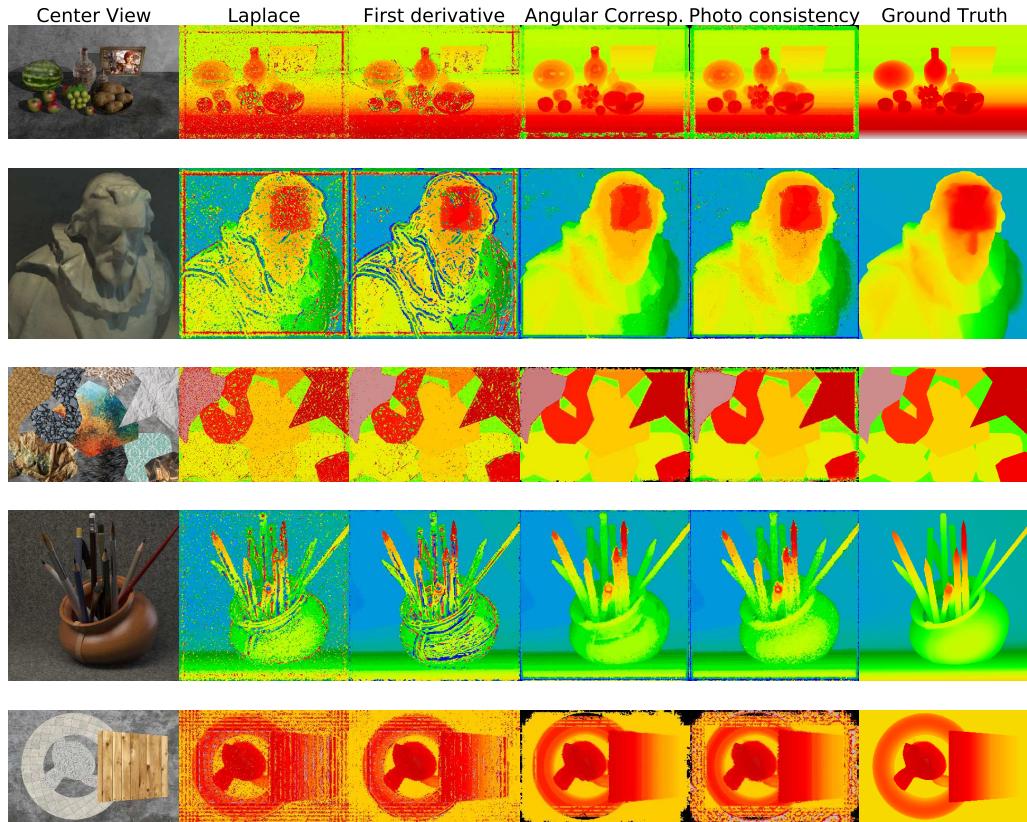


Figure 3.2: The depth maps of the scenes (from up to down) *complextestscene*, *cotton*, *occlusiontestscene*, *pens*, *testscene*, *tiltplane* is depicted for different depth-from-focus techniques. The Disparity step size is 1/20.

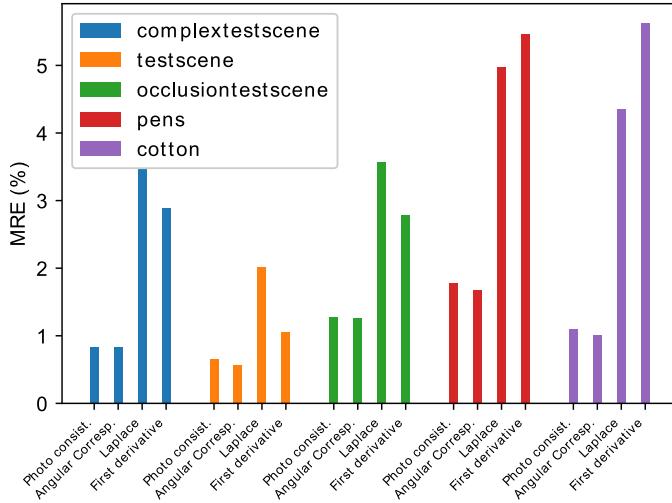


Figure 3.3: The mean squared error is depicted for depth-from-focus techniques. The step size is $1/20$.

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 3.7. Note that the edge of the depth images have been removed when calculating the MRE. Unfortunately the Photo Consistency pre-estimate pipeline yields worse results than the Structure Tensor itself. The reason is that artefacts from the pre-estimate maintain after the structure tensor pipeline has been applied to the EPI. Additionally one has to mention that only those pixels in the depth map that have a high coherence measure given by the ST pipeline are taken into account when calculating the errors. Hence a bias towards the Structure tensor pipeline in terms of optimal error results is expected.

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. The time difference between the scene is therefore enormous, the results can be found in table 3.1.1. If one jumps 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-estimate itself neither takes too long (about 2 seconds), but the EPI gets segmented into smaller parts when using the pre-estimate. A global smoothing scheme on the pre-estimate can possibly improve the runtime by minimizing the amount of disparity jumps, however this would result in even more artefact effects. We leave that aspect for future work.

In summary the obtained results using the Photo consistency measure as a pre-estimate for the structure tensor do not improve the obtained results regarding the Mean squared error and runtime-wise.

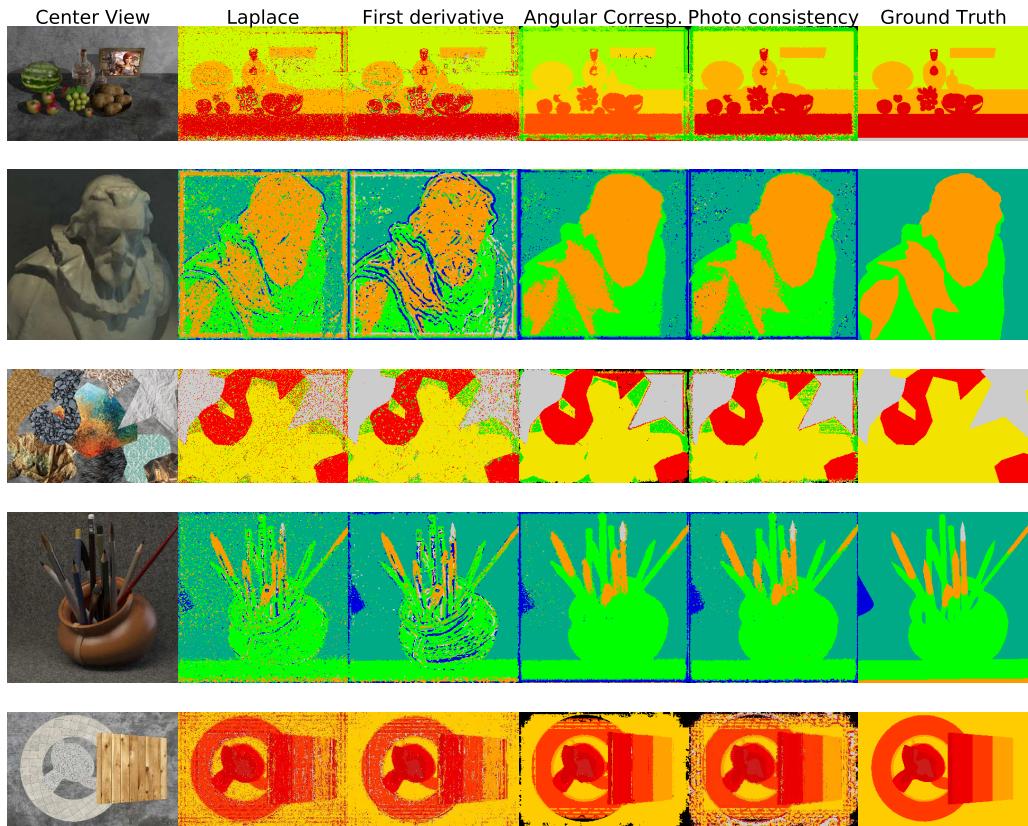


Figure 3.4: The depth maps of the scenes (from up to down) *complextestscene*, *cotton*, *occlusiontestscene*, *pens*, *testscene* is depicted for different depth-from-focus techniques. On the right the ground truth is depicted. The disparity step size is 1.

Scene	Structure tensor Pipeline	Pre-estimate+ ST	Use 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

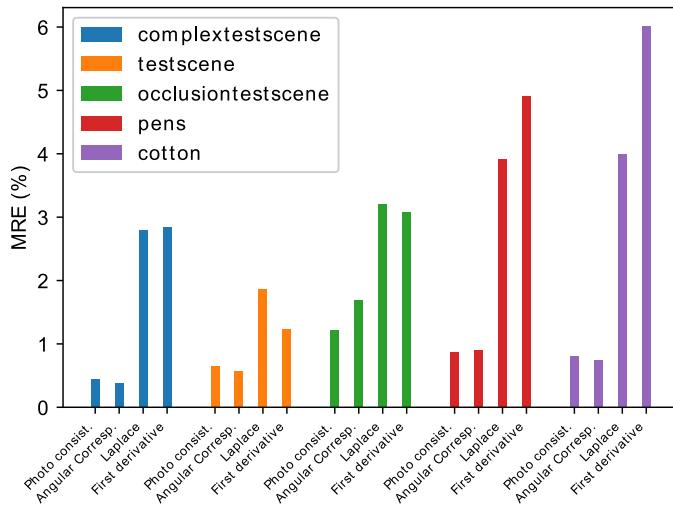


Figure 3.5: The mean squared error is depicted for depth-from-focus techniques. The step size is 1.

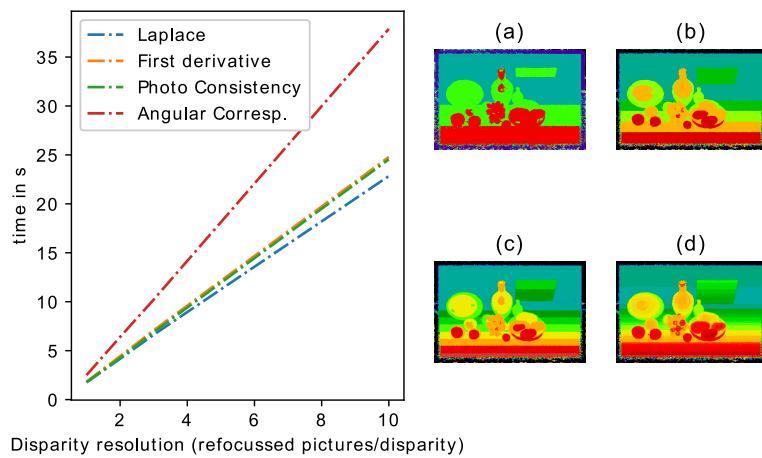


Figure 3.6: On the left the speed of different depth-from-focus techniques is depicted in function of the resolution chosen. (a)-(d) show the disparity map of the scene *complextestscene* for the resolutions 1, $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{10}$ respectively.

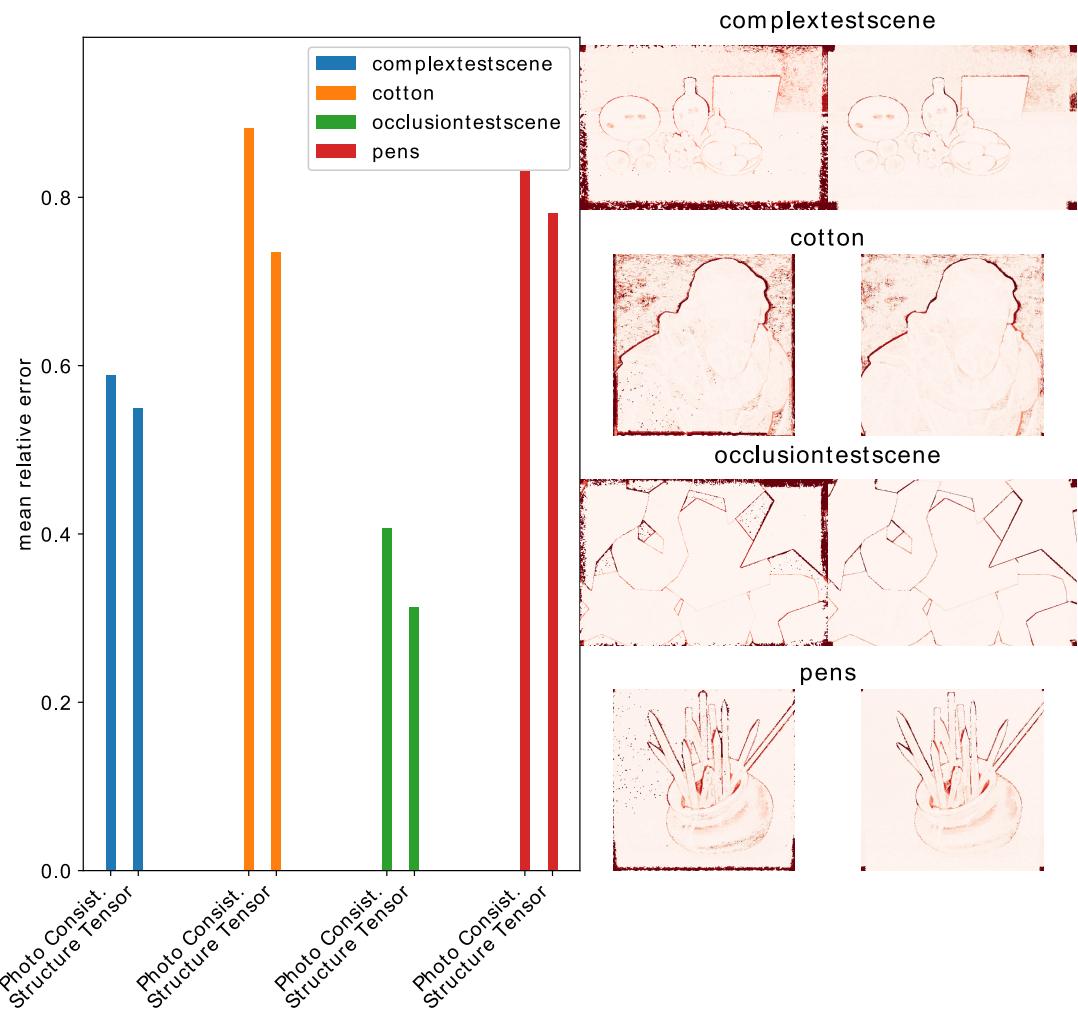


Figure 3.7: Comparison between the Structure Tensor Pipeline with and without a Photo Consistency pre-estimate. On the left the MRE is depicted for different scenes. The right images show the absolute difference of the depth map to the ground truth, while the left column represents the results with the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

3.2 Modification on the Structure Tensor

As depth from focus techniques didn't work out well for us, we examine different approaches by directly modifying the Structure Tensor Pipeline, with the aim on solving the occlusion problem while maintaining a dense disparity map.

In the following section we concentrate on benchmark data from the dataset of 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, lower score means always 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.

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 Mean Squared Error 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 Structure tensor pipeline. For large σ_{Color} both converge to 1, because in this limit the weighting based on the colour space distance (see equation 2.11) is a constant function. Thus one again 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 ($\sigma_{\text{Space}} = 7 \text{ pix}$). 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. (PICTURE). 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. 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 still unaffected since the edge fattening is still visible. Since the results do not satisfy the expectation and the guided bilateral filter also takes about 5 times

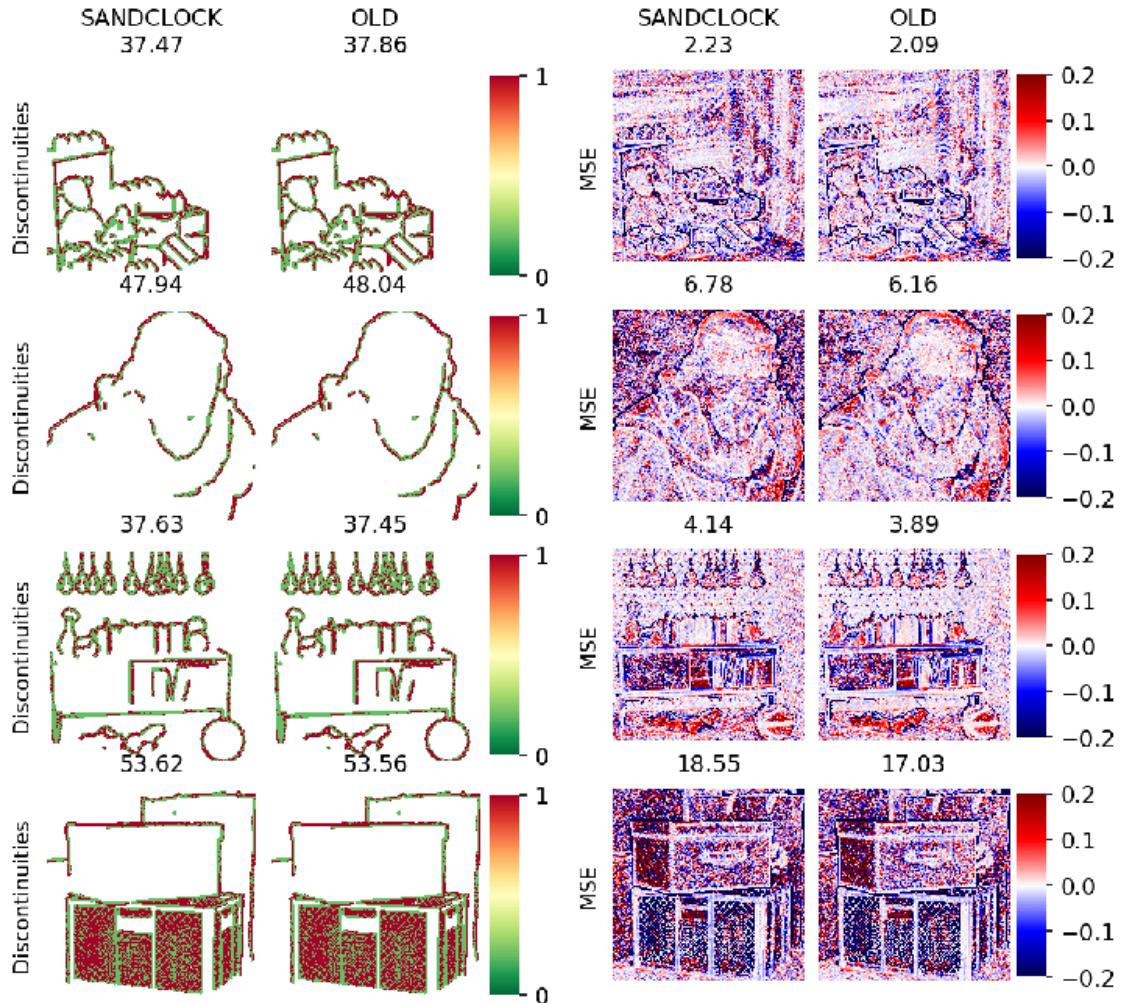


Figure 3.8: Comparison between the normal structure tensor and the sandglass-like kernel for the scenes *dino*, *cotton*, *sideboard*, *boxes*. The left side compares the mean squared errors in the disparity map, the right side shows the error where discontinuities in the disparity map are masked explicitly.

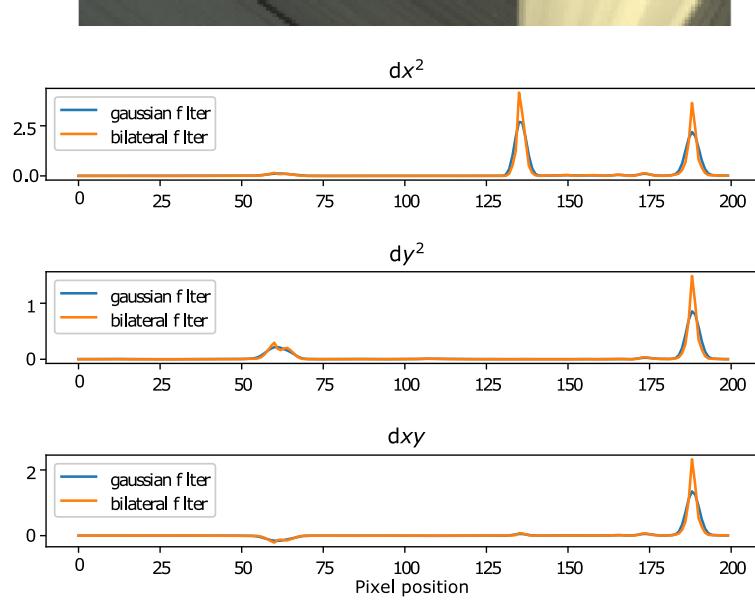


Figure 3.9: Bilateral filtering of an EPI

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

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

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. However, for each scene the score is dependent 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 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

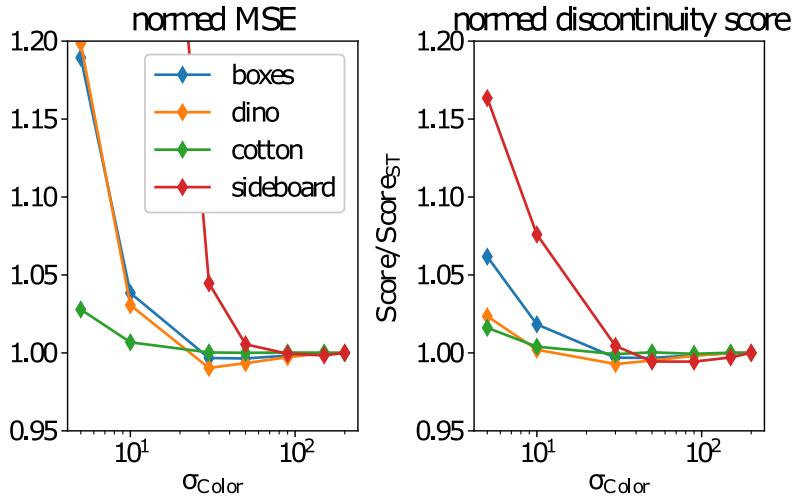


Figure 3.10: Bilateral filtering with original EPI as the guide: On the left the Mean Squared error for different scenes in function of the σ_{Color} of the bilateral filtering is shown, divided by the MSE with using a Gaussian filter. On the right the Discontinuity score based on Honauer et al. [2016]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

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 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 then above. The bilateral filter however does produce a consistent error that may be better to handle for post-processing

3.2.3 Thresholding Gradients

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 however underlay noise effects, therefore they are not upscaled. A threshold is set at which bigger gradient norms are normed to that threshold. 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

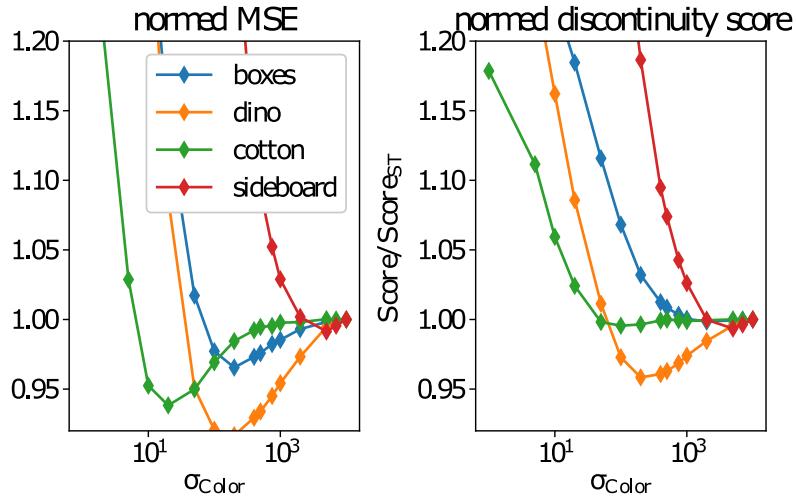


Figure 3.11: Bilateral filtering with Structure Tensor Components themselves as the guide: On the left the Mean Squared error for different scenes in function of the σ_{Color} of the bilateral filtering is shown, divided by the MSE with using a Gaussian filter. On the right the Discontinuity score based on Honauer et al. [2016] is shown, also normed to the Score using the standard Gaussian filter.

improves the MSE, since the gradient strength of the transition weights less into the structure tensor. The normed discontinuity score in the middle of figure 3.13 is counter-intuitive since it is even worse for low threshold values. The reason is that the mask used for calculating the discontinuity score by Honauer et al. [2016] is smaller than the edge fattening even after normalizing. Thus the discontinuity score cannot represent a better discontinuity behaviour if the edges do not lie in the mask.

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 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.

Another advantage of using the Thresholded Gradient ST becomes clear in figure 3.15, where 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

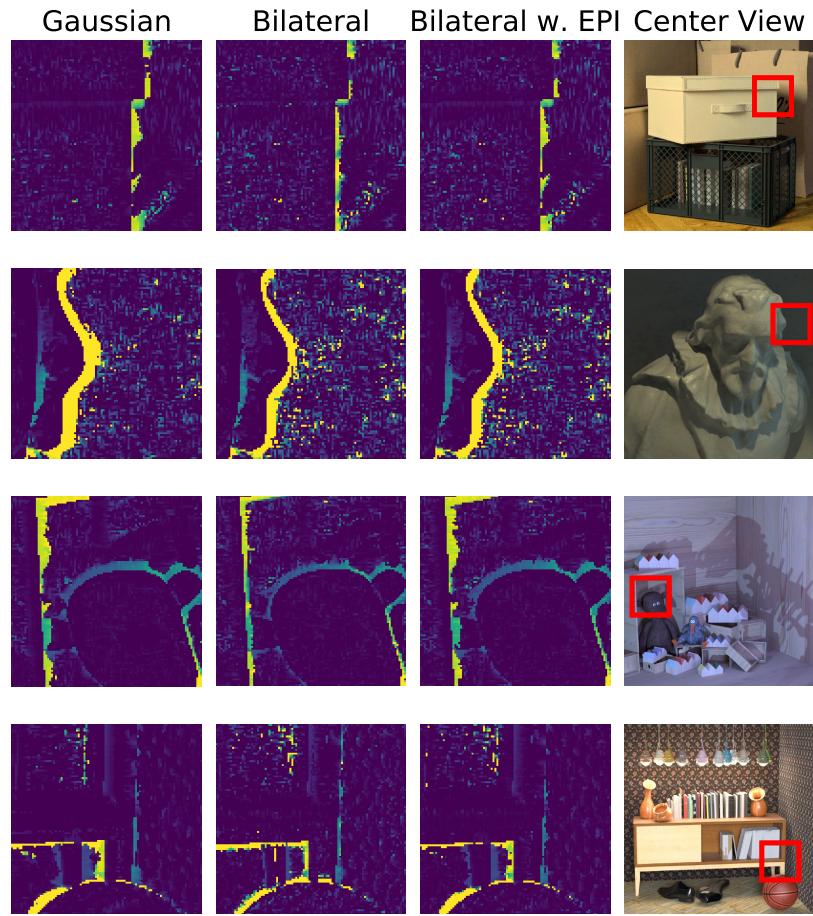


Figure 3.12: Results of Bilateral filtering for the scenes *boxes*, *cotton*, *dino*, *sideboard*.

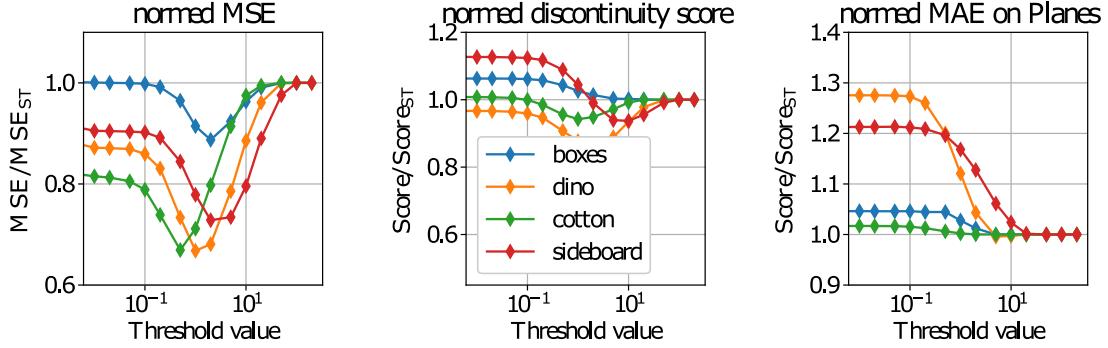


Figure 3.13: Results when thresholding the **Gradients** in the EPI are shown. All results are divided by the score obtained from the normal structure tensor algorithm. On the left the **Mean Squared Error** is shown, in the middle the discontinuity score is shown, on the right we see the **Mean Average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

affected by noise in both cases. Still the **Thresholded Gradient ST** is less affected by the discontinuity problems when increasing the kernel size, leading to a quasi-independence from the kernel size when looking at the MSE Score. **This fact is to be enjoyed with care as the kernel size does have a significant impact on the resulting image.**

3.2.4 Occlusion awareness segmentation

In the next step we are trying to detect occlusions in the scan directly via the **Gradient norm**, as a high gradient is a good indicator for an occlusion edge. The result however 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 *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 go for bigger kernels.

The larger the kernel size is, the **bigger** the connected parts belonging to the same segment in the EPI are. More segmentations have 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 approved 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.

(FIGURE SHOWING PIXEL Distance's / ZOOMED IN EPI NORM)

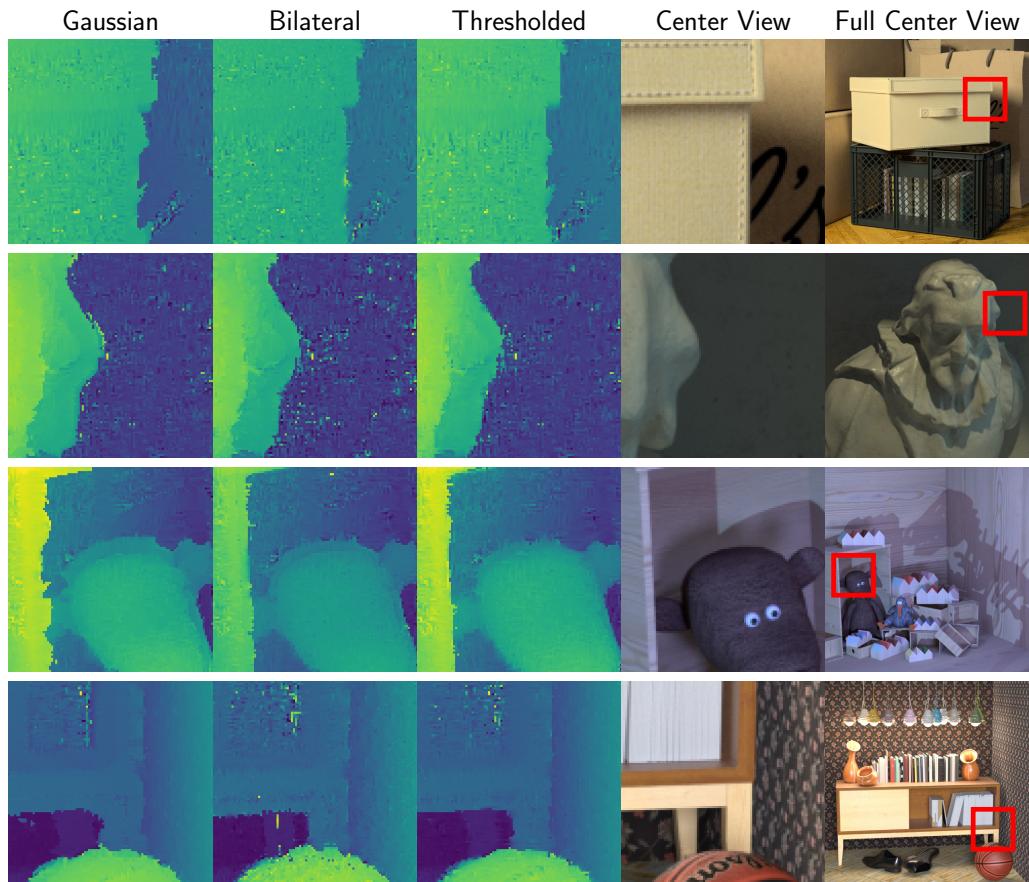


Figure 3.14: Disparity map **Zoom-ins** for different **Structure Tensor** filtering. From Left to right Gaussian filtering, bilateral filtering and Gaussian filtering when thresholding the gradients.

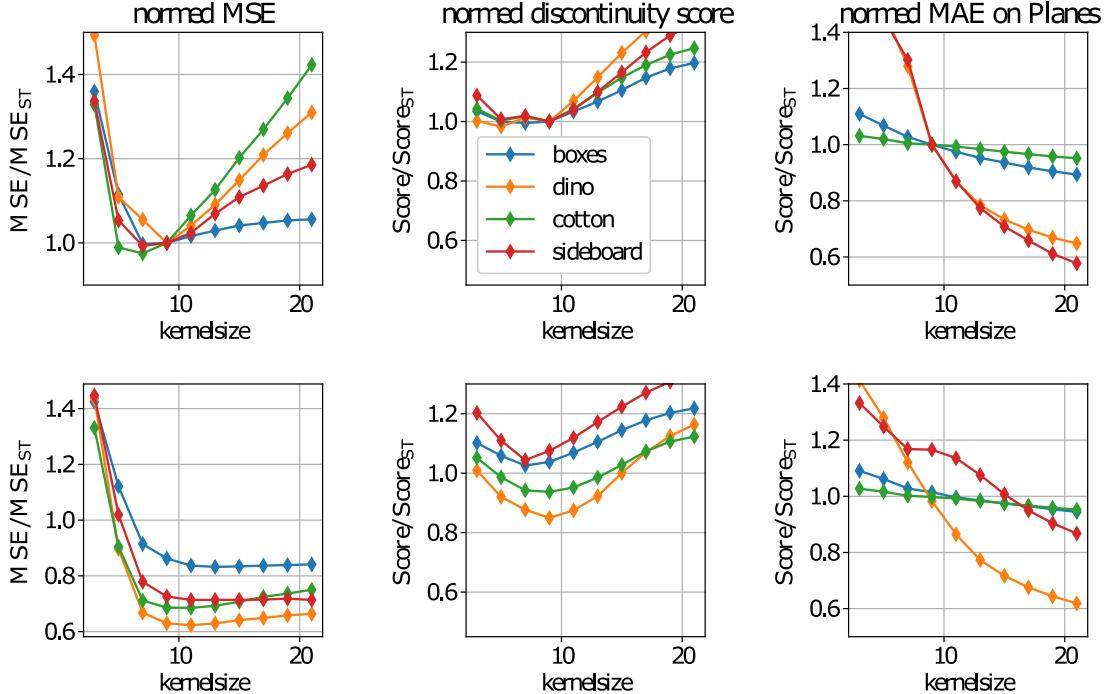


Figure 3.15: Top: Varying the kernel size, using the old ST pipeline. Bottom: Varying the kernel size with thresholding the gradients.

In figure 3.17 the segmentation algorithm is compared to the old pipeline as well as to the thresholded norm pipeline that is evaluated on 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 **occlusion handling**, 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 **the 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). However in the other scenes the segmentation does not seem to work well as the algorithm only achieves minor improvements.

3.2.5 Alternative coherence

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

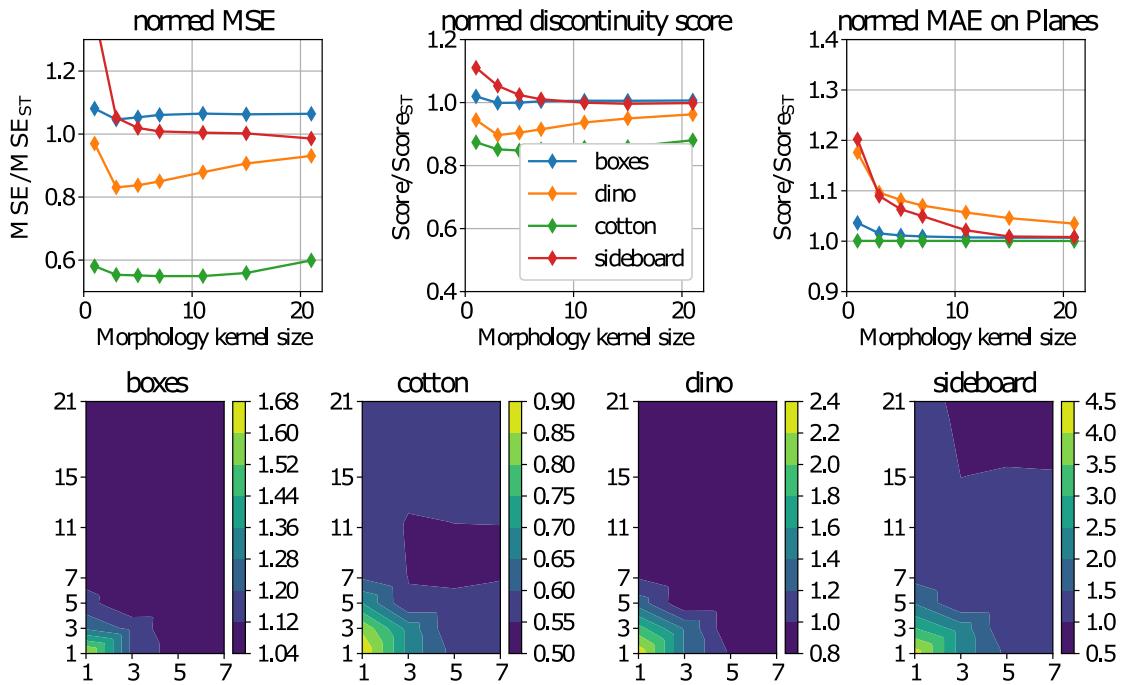


Figure 3.16: Upper row: Metric scores in dependence of the size in y -direction of the morphology kernel, x -direction size is 7.
 Lower row: dependence of the MSE for the size of the kernel. the y -size is depicted on the x -axis.

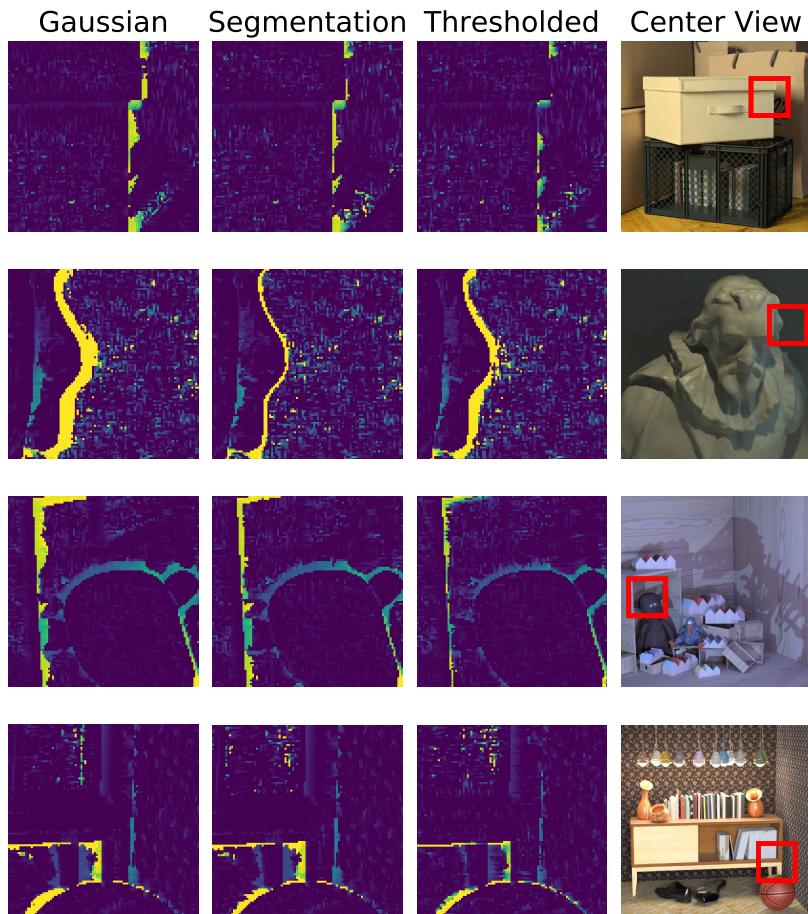


Figure 3.17: Disparity map Zoom-ins for different Structure Tensor filtering. From Left to right Gaussian filtering, segmentation filtering and Gaussian filtering when thresholding the gradients.

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 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 amount 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 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. Note that in all scenes zero is circa the mean depth of the scene. 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 seems to be equal in both cases.

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, because statistically "random" outliers are easier to handle in a post-processing step than systematic errors such as edge fattening. The trade-off is controlled via the threshold in equation 2.34 and is set to 0.7 in figure 3.23. In the close-up views of the deviation from the ground truth in all four scenes (column 3 and 4 in figure 3.23) 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 however the noise has increased. On planar areas the higher coherence value is mostly the better one while on occlusions the coherence does not serve as a good measure to decide whether to take the x-directional or the y- directional depth estimation. As a result, one faces noise as

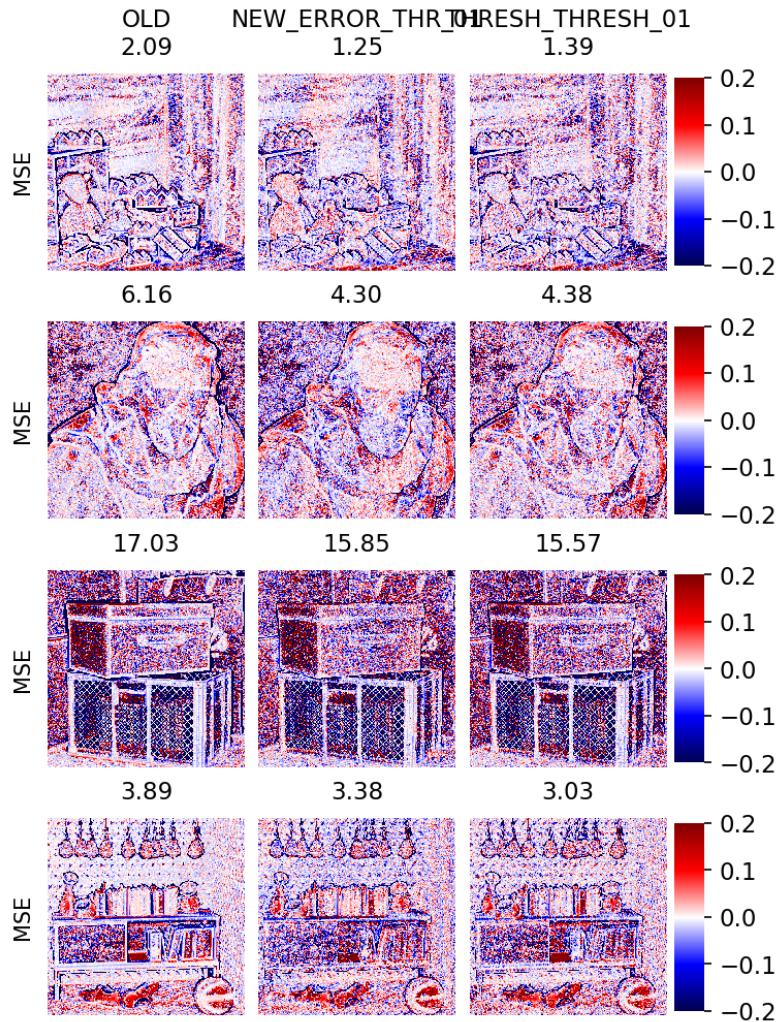


Figure 3.18: Left: The Mean squared error using the old ST pipeline. Middle: Using the alternative error measure + thresholding the gradients. Right: Using the old coherence measure + thresholding the gradients.

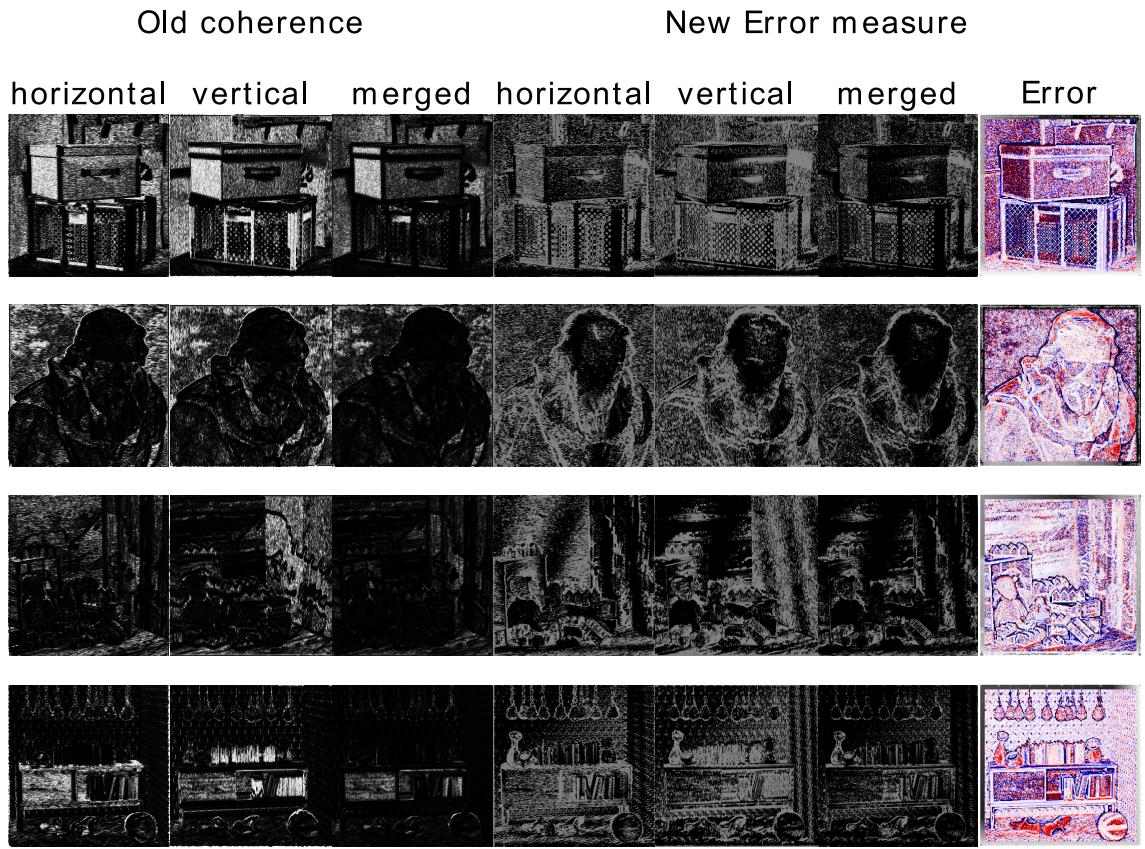


Figure 3.19: Columns on the left: The coherence value from the horizontal EPI, vertical EPI and the merged coherence map. Middle to the right columns: Alternative error measure based on error propagation. On the very right: Actual deviation from the ground truth.

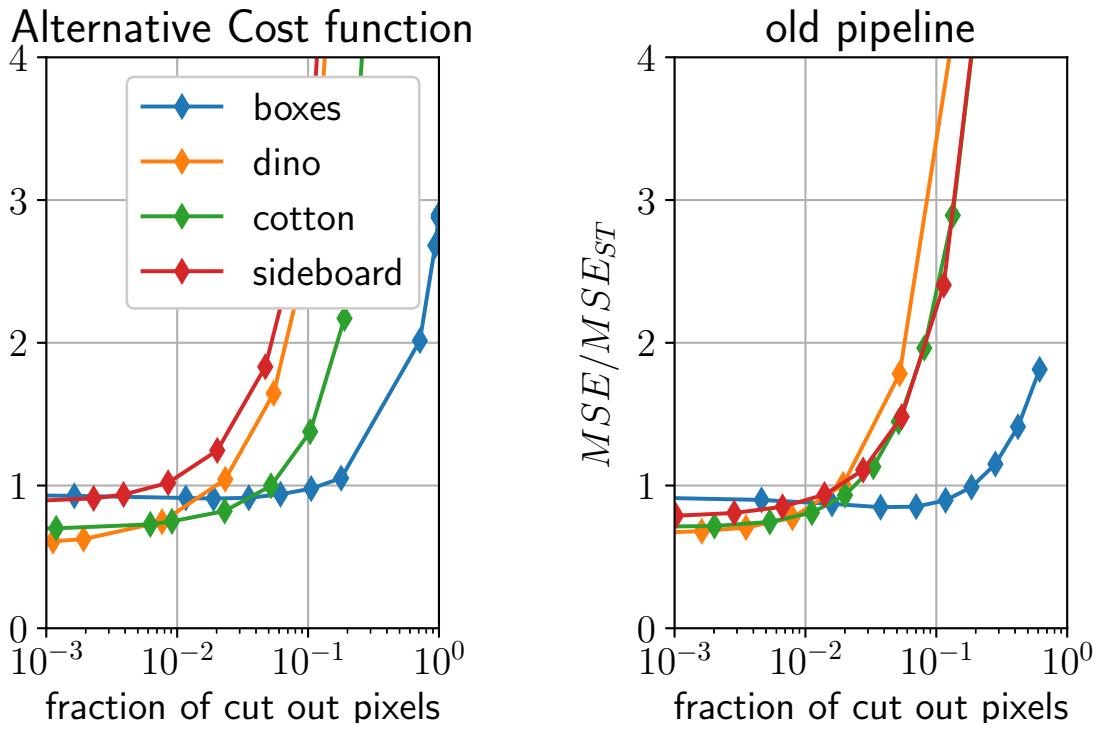


Figure 3.20: The relative MSE is plotted on the percentage of pixels that are set to 0 at different coherence values. Left the New error metric is used, right the old pipeline is used.

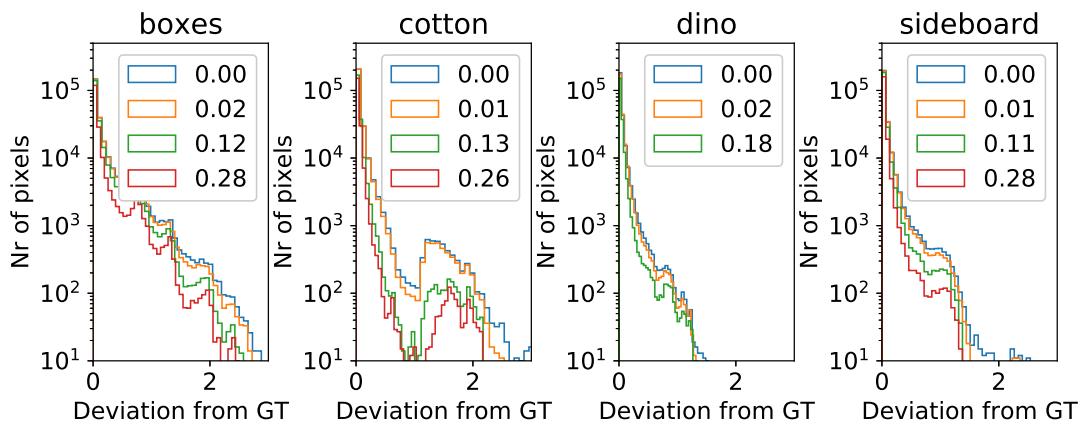


Figure 3.21: The error histogram for different masks based on thresholding the coherence map. The labelling indicates the fraction of masked out pixels.

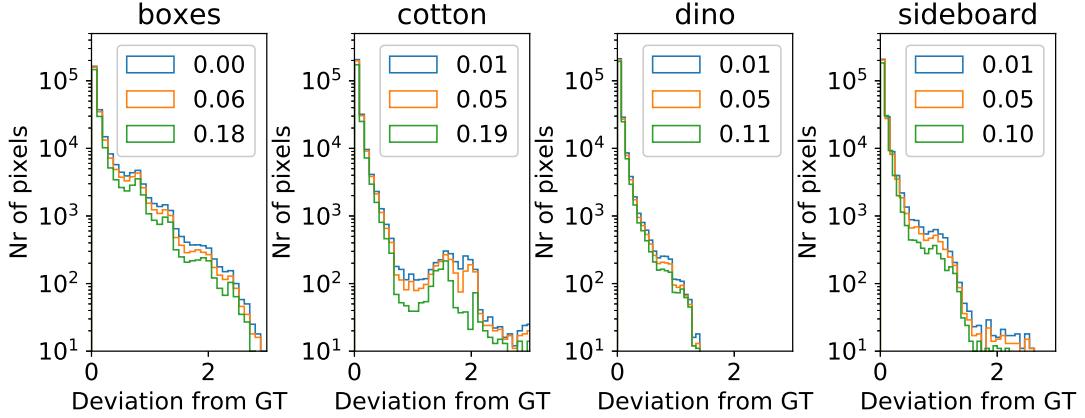


Figure 3.22: The error histogram for different masks based on thresholding the new error map. The labelling indicates the fraction of masked out pixels.

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 3.24 on the left side. The threshold value 0 corresponds to the old pipeline, the threshold value 1 corresponds to always choosing the lower disparity value. Choosing big threshold values however leads to increased noise, thus worsening the MSE score. Note that we can see from figure 3.24 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 3.25 where the algorithm is combined 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 [Semi-global Matching](#) algorithm in section 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 by 1. Even though in figure 3.24 and figure 3.23 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 3.25 on the left. For a higher [Merging](#) parameter the BadPixel Score is always increasing due to higher noise coming from low coherence disparity values.

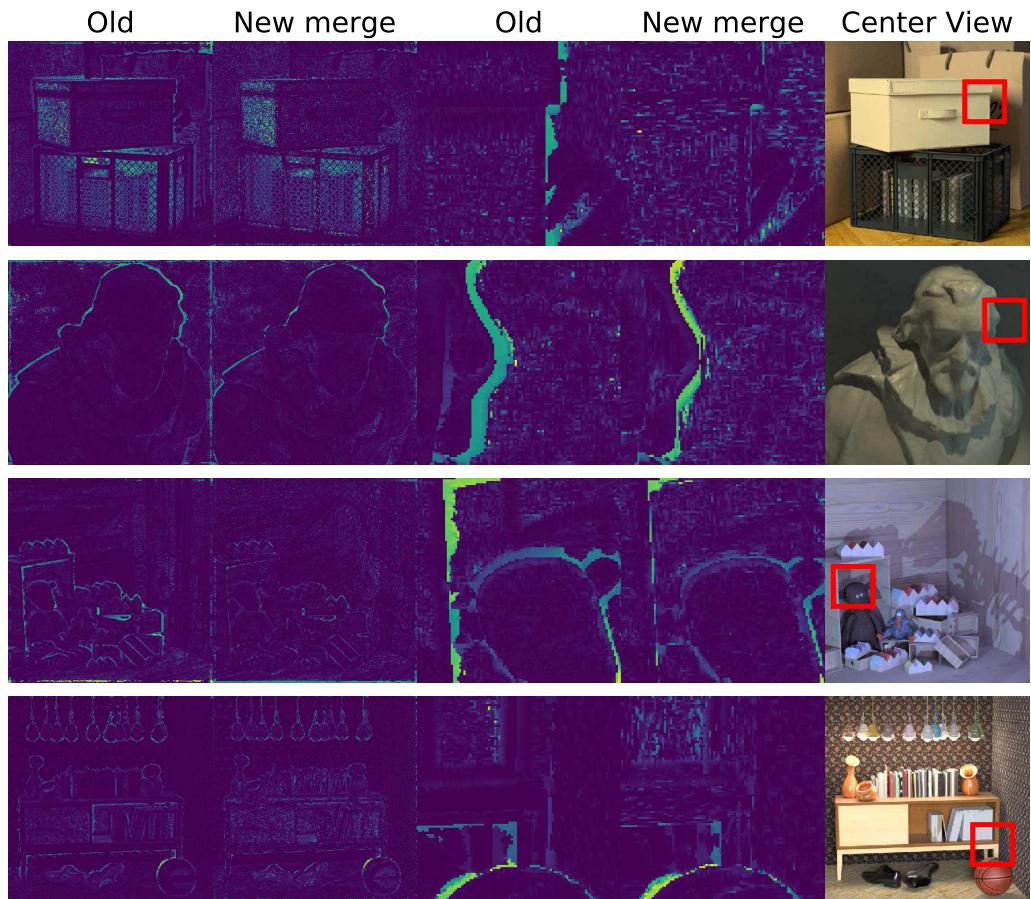


Figure 3.23: Left columns: Comparison between the old merging and the new merging for the whole depth map. The absolute deviation from the ground truth is shown. Middle columns: Close-up view on occlusion edges. Right column: Localisation of the close-up view in the scene. Parameter from equation 2.34 is set to 0.7.

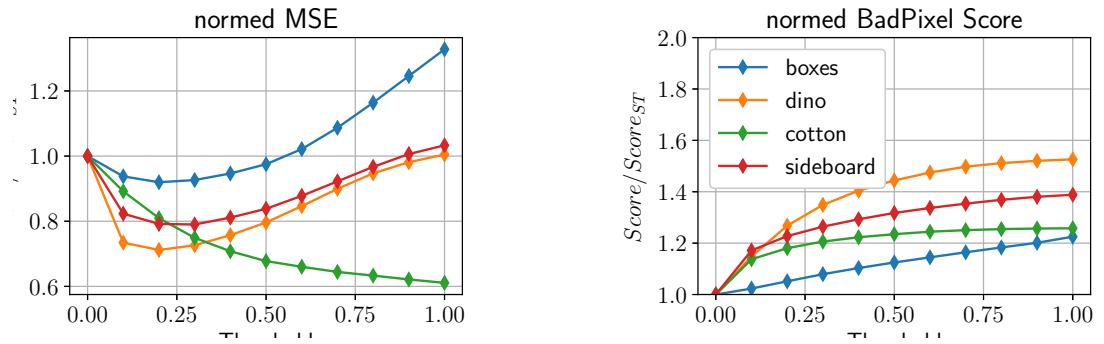


Figure 3.24: Alternative Merging: Parameter dependence. Left: Mean Squared Error. Right: BadPixel Score

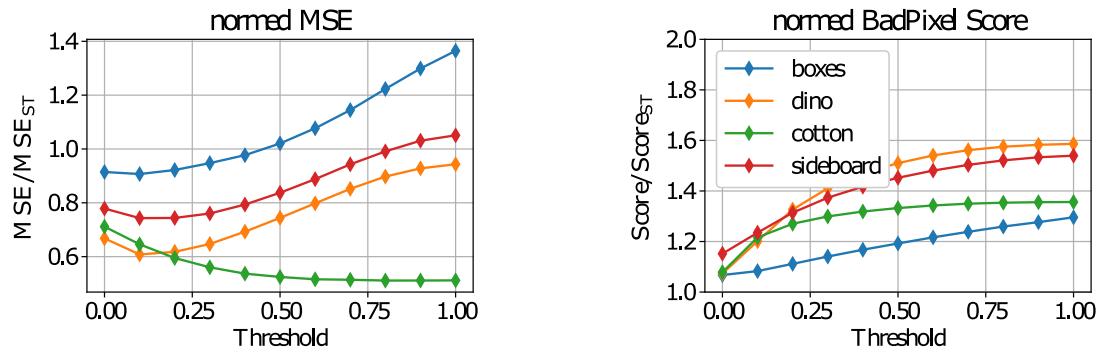


Figure 3.25: Alternative Merging + thresholded gradients: Parameter dependence. Left: Mean Squared Error. Right: BadPixel Score

3.3 Semi-Global Matching

First we evaluate the SGM algorithm as described in section 2.5.3. The Semi-Global Matching algorithm as implemented here is dependent on 3 parameters from equation 2.45.

We tried to perform a grid search to find the best parameters. The illustration of the grid search can be found in the appendix in figure ??, ?? and ?? for different metrics. Even though one may conclude from figure ?? that the score for planes is improved by setting a high parameter 1, the difference between the maximum and minimum score in the grid search is too small to see any 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 whom 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 3.26, where the old pipeline is compared to the result using with the SGM algorithm. The depth map is improved at surfaces with noisy values, thus the SGM has a smoothing effect. However at occlusion edges the SGM pipeline does not perform significantly better.

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 3.27 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 1 pixel shift and they are only punished by a small penalty. The cost function itself could be used to mask out possibly erroneous pixels. To test the quality of such a masking the error histogram for all 4 scenes is plotted in figure 3.28. It becomes obvious for scene *dino* and *sideboard* that large errors are masked out more often than it did with using the old coherence measure as depicted in figure 3.21. In scene *boxes* the disparity map is quite noisy which makes it hard to mask out those regions.
However the usability of an error measure based on SGM is restricted due to the dependence on the penalty parameters mentioned in equation 2.45.

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 3.29 shows how MSE, BadPixel Score and processing time

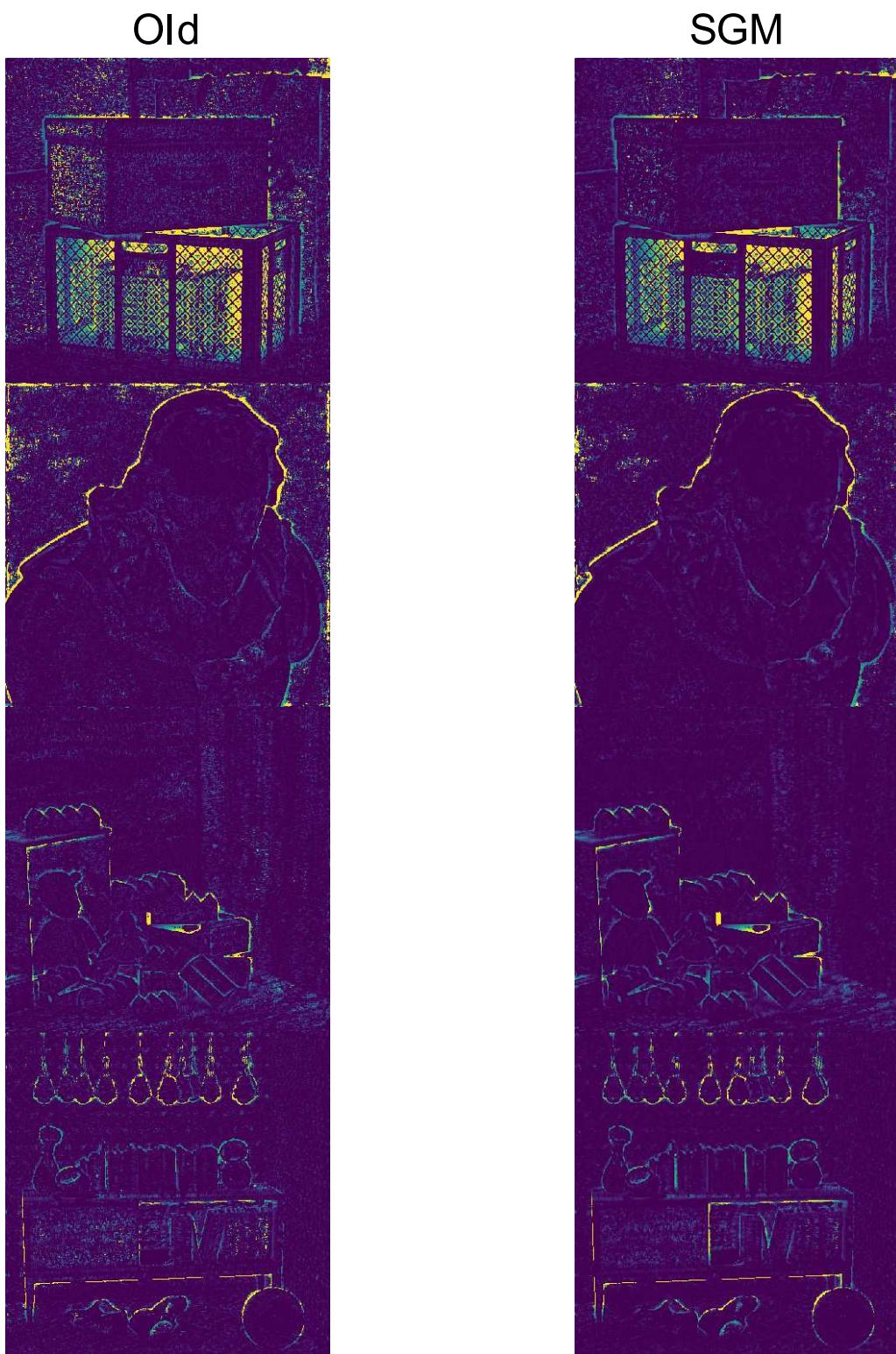


Figure 3.26: Disparity error maps using the old pipeline (left) and the [Semi-Global Matching Algorithm](#) (Right). Yellow is a big deviation, [dark-blue](#) is no deviation.

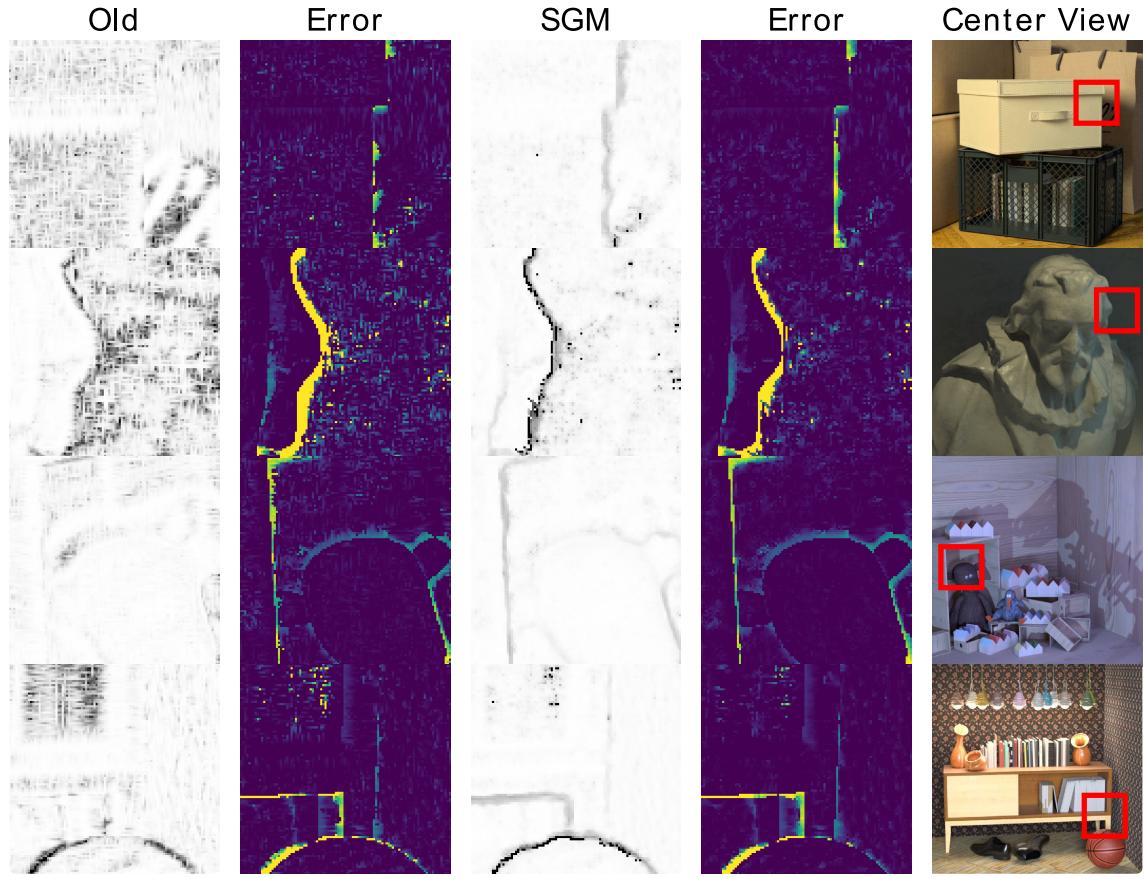


Figure 3.27: The assumed error (grey-valued) when calculating depth without using SGM (left) and with SGM (right). In Colour the actual error for both pipelines is depicted.

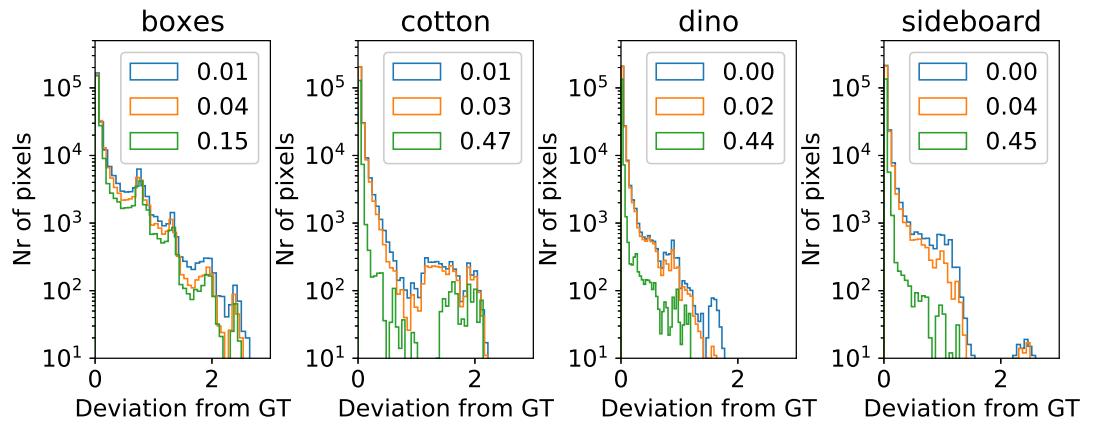


Figure 3.28: The error histogram for different masks based on thresholding with the cost function . The labelling indicates the fraction of masked out pixels.

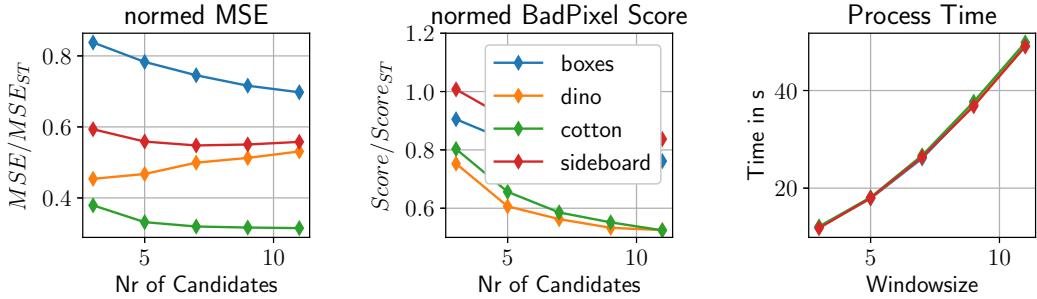


Figure 3.29: SGM as pure post-processing combined with Alternative merging: The Number of candidates that is chosen from the direct neighbourhood is varied. Left/Middle: The MSE/BadPixel Score in dependence of the Nr of candidates. Right: the processing time of the algorithm dependent on the Nr of candidates. 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 can not 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 3.30 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 large monochrome areas leading to noisy depth maps. One solution to handle 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 ??](#).

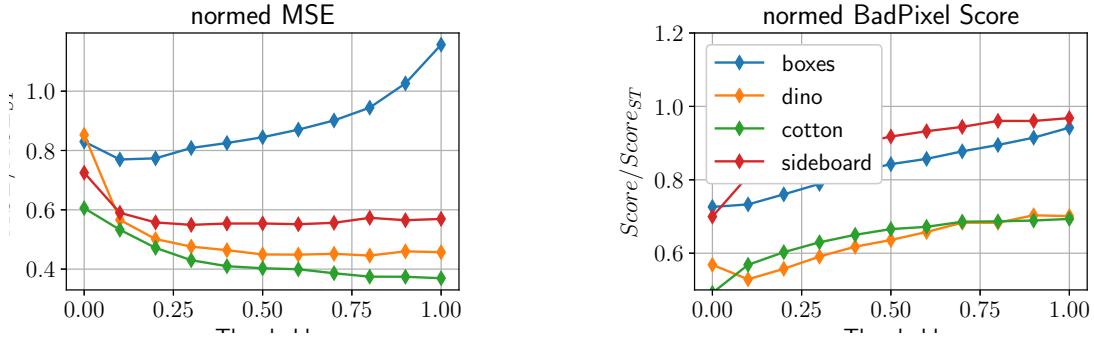


Figure 3.30: Alternative Merging with SGM post-processing Left/Right: MSE/BadPixel Score as a function of the merging threshold.

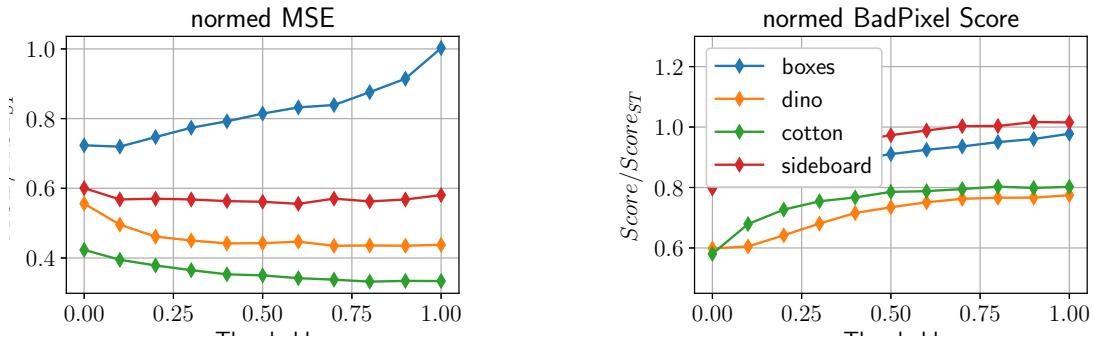


Figure 3.31: Alternative Merging with SGM post-processing while using thresholded gradients. Left/Right: MSE/BadPixel Score as a function of the merging threshold.

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 3.31 the MSE and the BadPixel Score are shown in dependence of the merging parameter. While the BadPixel Score behaves approximately as without a gradient threshold (figure 3.30), the MSE stays approximately constant for all scenes besides scene *boxes*. No improvement big improvement can be obtained, since the thresholding of the gradients already handles occlusions well. In figure 3.30 as well as in figure 3.31 the MSE does not rise for a larger merging parameter as it does in figure 3.24 and in 3.25, 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 3.32 the MSE is shown in dependence of Nr of candidates and window size for all four scenes, in figure 3.33 the BadPixel Score is shown and in 3.34 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 fixed the window size for all calculations to 5x5 pixels. In general A higher Number of candidates promises

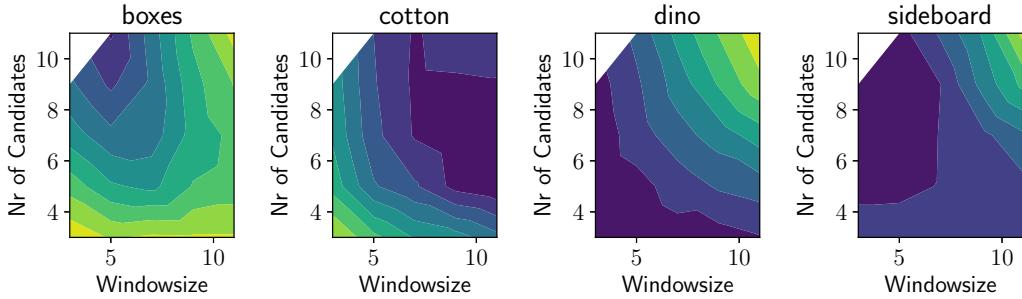


Figure 3.32: MSE Parameter dependence for post-processing SGM using alternative merging+thresholded gradients.

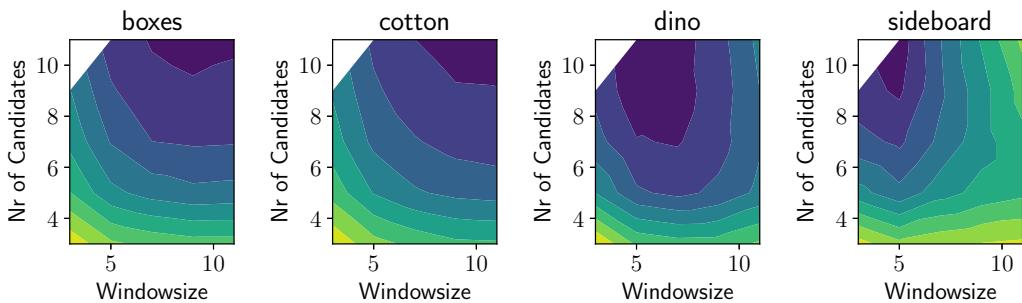


Figure 3.33: BadPixel score parameter dependence for post-processing SGM using alternative merging+thresholded gradients.

for better results, however the processing time scales linearly with the number of candidates. Thus we stick to Nr of candidates of 5, as shown in figure 3.29 on the right.

3.3.1 Iterative SGM post-processing

In the following we evaluate the Iterative to 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 we define as the mean cost value at all pixels. In figure 3.35 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 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 3.35 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 3.36. On the left the normed MSE is plotted. It becomes obvious that all

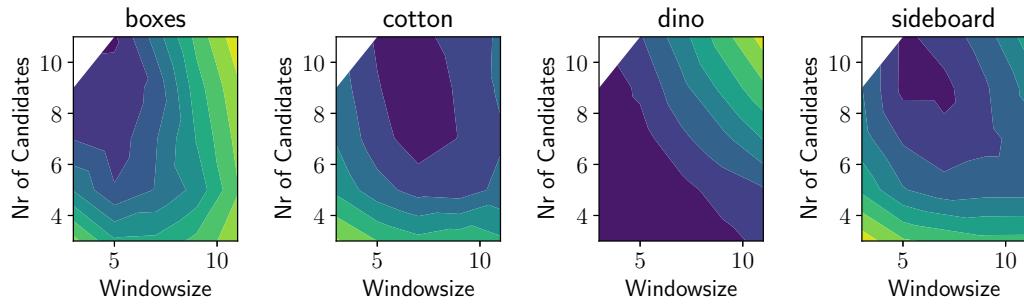


Figure 3.34: Discontinuity score parameter dependence for post-processing SGM using alternative merging+thresholded gradients.

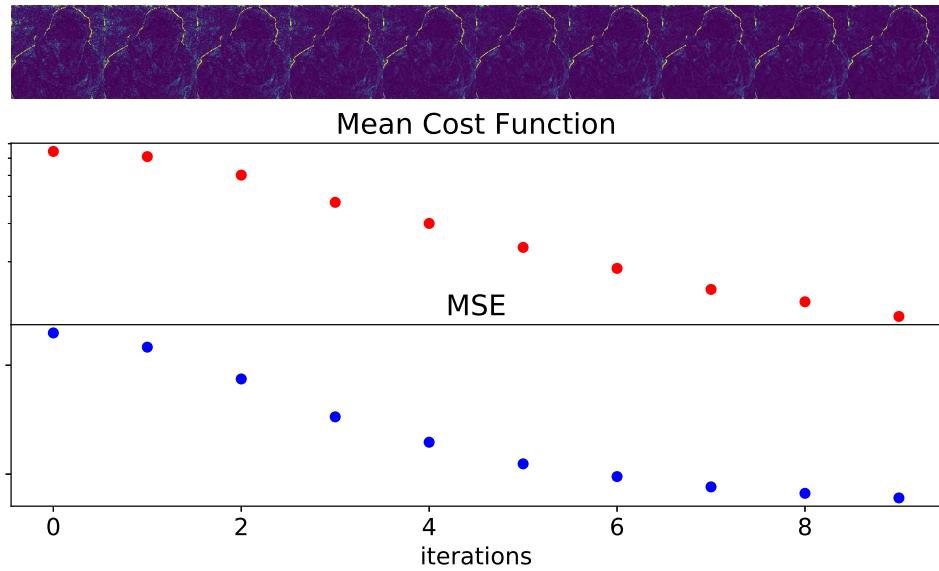


Figure 3.35: In the upper row the resulting depth map deviation from ground truth of the iterative SGM pipeline is shown for scene *cotton*. Below the Mean cost function averaged over all pixels as well as the MSE is depicted in dependence of the number of iterations. Note that the MSE is not normed as in figure 3.36.

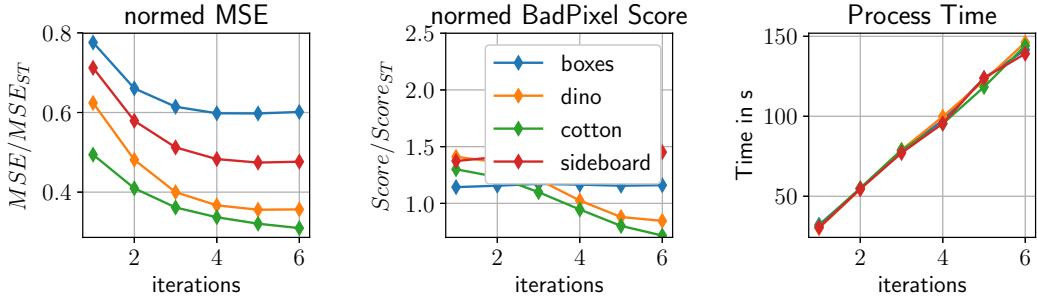


Figure 3.36: Iterative SGM pipeline dependence on Number of Iterations. Left: normed Mean Squared Error for all four scenes in function of Nr of iterations. Middle: Normed BadPixel Score in function of Nr of iterations. Right: Processing Time

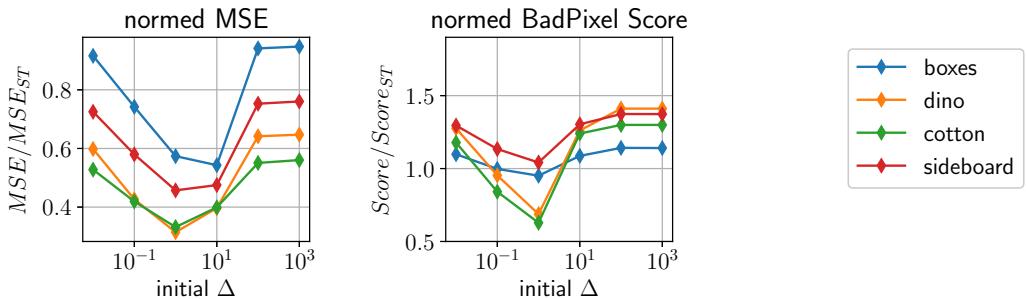


Figure 3.37: Iterative SGM pipeline dependence on Initial Δ with fixed Number of iterations (5). Left: normed Mean Squared Error for all four scenes in function of Δ . Middle: Normed BadPixel Score in function of Δ . Note that the x axis scale is logarithmic.

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 MSE as well as on the BadPixel Score. In Figure 3.37 The MSE and BadPixel Score are plotted in function of Δ_{initial} . The iteration number is set to five. For very low as well as for high initial Δ no convergence can be reached in a reasonable amount of iterations. Doing only 5 iterations 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.

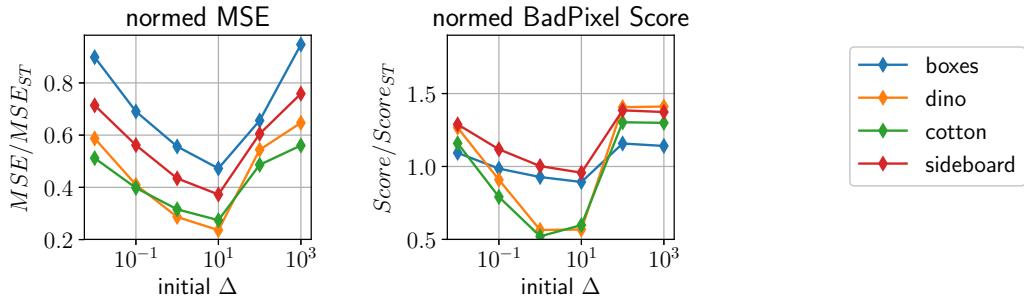


Figure 3.38: Same plot as figure 3.37, doing 10 iterations instead of 5.

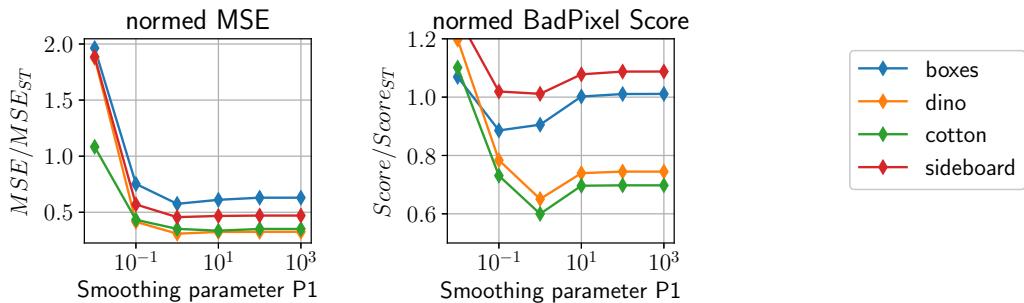


Figure 3.39: Iterative SGM pipeline dependence on Smoothing parameter P1 from equation 2.48 with fixed Number of iterations (5). Left: normed Mean Squared Error for all four scenes in function of P1. Middle: Normed BadPixel Score in function of P1. Note that the x axis scale is logarithmic.

The same plot as in figure 3.37 can be found in figure 3.38, 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 big 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 2.48. This changes the form of penalizing disparity deviations from 2 neighbouring points on a SGM scanline. For very small P1 the penalty is close to being a constant regardless of the disparity deviations, thus no convergence can be reached. In figure 3.39 this can be seen in the left

Figure 3.40: 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



Figure 3.41: 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.

3.3.2 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 HDVision 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 3.41 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 white pixels in between the black area, which we aim to include in the SGM path-wise optimization. Hence the masked 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 more dense (d)). e) shows the centre view of the camera in comparison .

Another application example on real data is the Lego truck from the Stanford light field archive depicted in figure 3.42 (h)). We focus on a small clipping of the scene which is depicted in figure 3.42 (g)). One sees the effects of different coherence thresholds, merging parameters and gradient thresholds in figure a) - f). Figure a) shows the **Basic structure tensor result** with a 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 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. In the Appendix one finds the **Full** depth map without clipping (section B).

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 Gauss 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

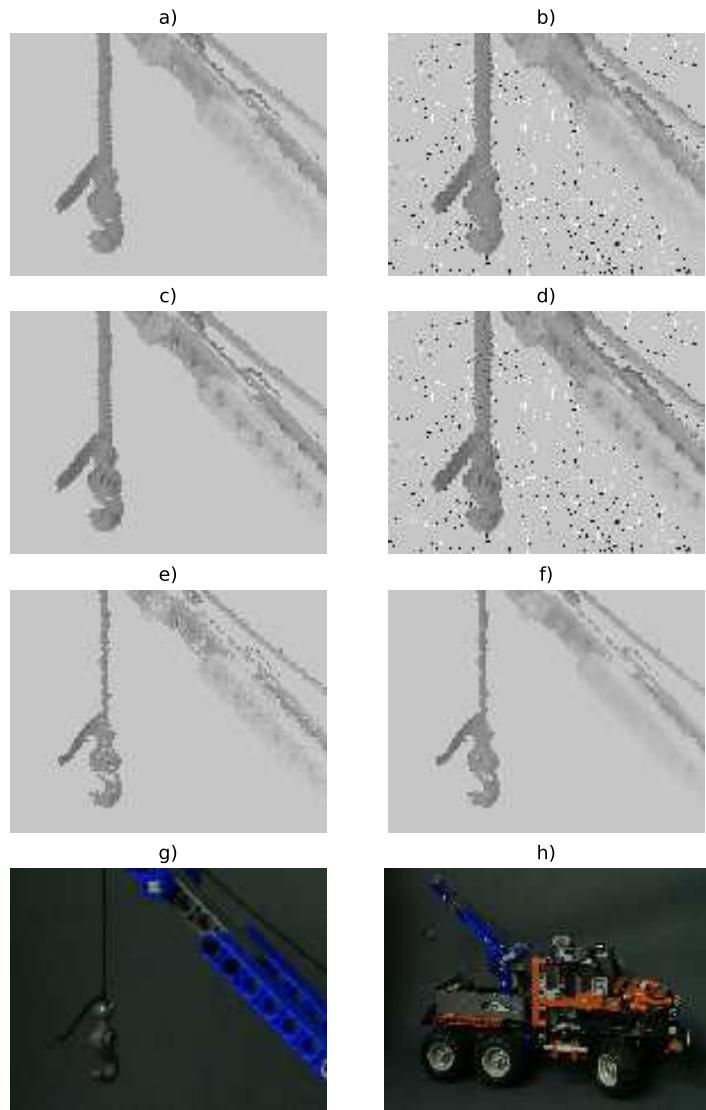


Figure 3.42: a) Depth map prod. by the structure tensor pipeline, with minimal coherence set to 0.85. b) Depth map prod. by the structure tensor pipeline, with minimal coherence set to 0.5. c) Depth map when using alternative merging, minimal coherence set to 0.85. d) Depth map when using alternative 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.

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 (7x7)	4.82	11.51	4.31	1.50	2.58
PPR Gauss filtering (9x9)	4.84	11.34	4.16	1.46	2.52
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 (3x3)	4.80	13.21	4.96	1.83	3.35
PPR Median filtering (5x5)	4.83	12.73	4.76	1.76	3.20
PPR Median filtering (3x3) + Gradient Threshold (0.1)	6.09	10.68	3.14	1.06	2.29
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

iterations. Applying a filter afterwards only worsens the MSE.

4 Resumee and Outlook

In the following we will give an overview over the different methods that were implemented and 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, thus fastening 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 Structure tensor). 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*, however 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 process, however we realized that results were better when proposing disparity candidate values artificially in a post-processing step solely dependent on the estimated depth map. In an iterative manner we minimized the cost function to get a smooth result while retaining clear edges. 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 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 this work tried to locate and

separate glossy surfaces in the scene based on the work of Tao et al. [2017]. However this side project did not work out successfully to present any results, leaving a branch of possibilities for further research.

However, while working on the thesis, recent publications (Luo et al. [2017], Changha Shin [2018]) 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 $g : \Omega \rightarrow R, \Omega \subset R^D$, the preferred local direction $\vec{n} \subset R^D$ must satisfy the following equation:

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

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:

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

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

$$\vec{n}_{\text{preferred}} = \operatorname{argmax}_n (\vec{n} J \vec{n}) \quad (\text{A.3})$$

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

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

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

In two dimensions we can write

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

where „*“ describes a convolution.

B Real Data

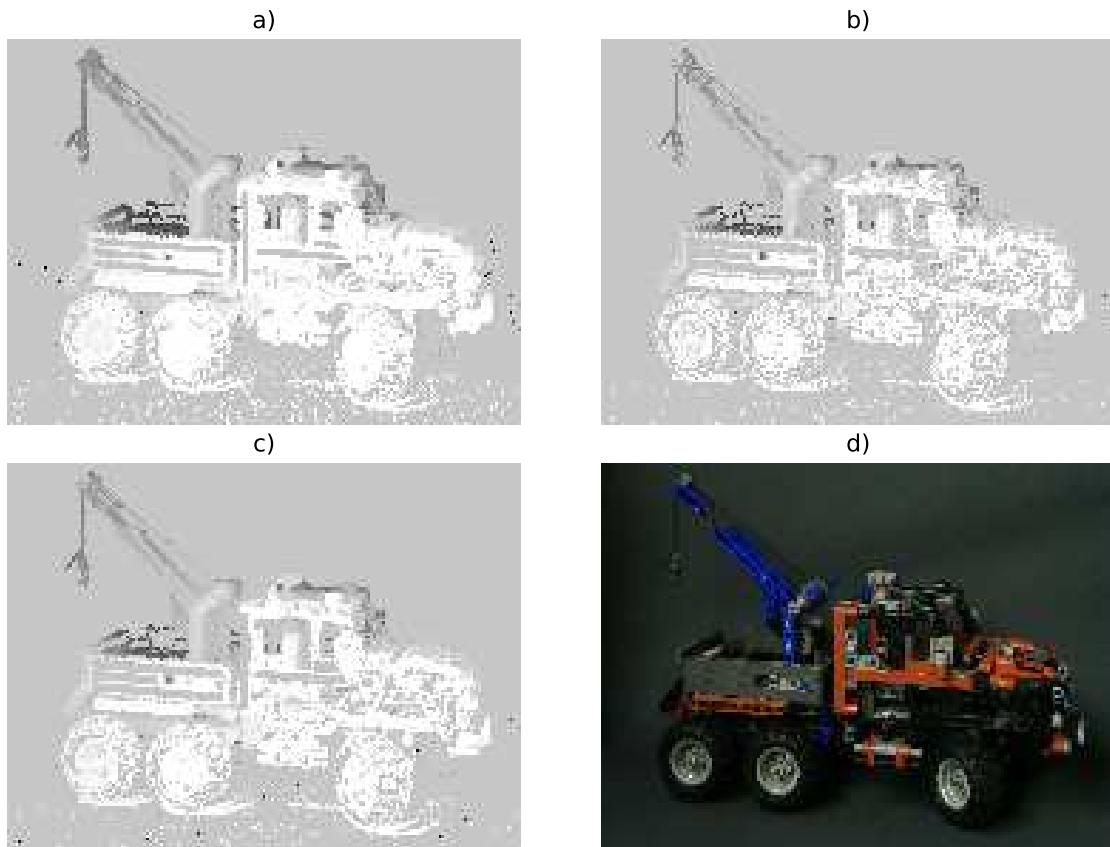


Figure B.1: a) Depth map produced by the structure tensor pipeline, with minimal coherence set to 0.85. b) Depth map when thresholding gradients, minimal coherence set to 0.85. c) Depth map when thresholding gradients and using iterative SGM post-processing, minimal coherence set to 0.85. d) full centre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

C CNN Disparity Estimation implementation

The approach of using machine learning for 3D reconstruction from light field 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 ?? 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, however 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 ??, 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 ?? on the right the merged depth map is depicted in comparison to the Structure tensor 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.

D Lists

D.1 List of Figures

2.1	Two-plane parametrisation	10
2.2	Visualization of an EPI extraction	11
2.3	Example Epipolar Plane image	11
2.4	LumiPlus Scanner from HDVision Systems	13
2.5	Refocussed EPI	14
2.6	Occlusion in an EPI	16
2.7	Discontinuity evaluation	17
2.8	Sandglass Kernel	17
2.9	Segmenting an EPI	21
2.10	Morphological closing	21
2.11	Enhanced segmentation mask	22
2.12	Alternative Merging:Parameter dependence	25
2.13	Example values: One point \vec{p} contains different values	29
2.14	Discrete scanline to continuous scanline	30
2.15	SGM from different directions	30
2.16	Example for post-processing smoothing	32
2.17	Iteration steps	33
2.18	Penalty function	34
2.19	Linear Cost function results	34
3.1	Pixel response for depth from focus techniques	39
3.2	Depth from focus: depth maps	40
3.3	Mean squared error for depth-from-focus techniques	41
3.4	Depth from focus: depth maps with Resolution 1	42
3.5	Mean squared error for depth-from-focus techniques	43
3.6	Speed of different depth-from-focus techniques	43
3.7	Photo-consistency Pre-estimate for the structure tensor	44
3.8	Results with custom sized kernel	46
3.9	Bilateral filtering	47
3.10	Parameter dependency for bilateral filtering using EPI	48
3.11	Parameter dependency for bilateral filtering	49
3.12	Results of the bilateral filtering	50
3.13	Results when thresholding the Gradients in the EPI	51
3.14	Disparity map Zoom-ins for different methods	52
3.15	Varying kernel size	53

3.16 Segmentation of EPI dependence on the morphology kernel	54
3.17 Disparity map Zoom-ins for different methods	55
3.18 Comparing alternative coherence to new coherence	57
3.19 Old error vs New Error	58
3.20 Dependence of error on cut out pixels	59
3.21 Histogram when masking errors	59
3.22 Histogram when masking errors with new error map	60
3.23 Results from alternative merging of x- and y- direction	61
3.24 Alternative Merging: Parameter dependence	62
3.25 Alternative Merging: Parameter dependence	62
3.26 Semi-Global Matching Results	64
3.27 SGM Coherence measure	65
3.28 Histogram when masking errors with new error map	65
3.29 SGM as pure post-processing combined with Alternative merging parameter dependence	66
3.30 Alternative Merging with SGM post-processing	67
3.31 Alternative Merging with SGM post-processing with thresholded gradients	67
3.32 Parameter dependence for post-processing SGM using alternative merging+thresholded gradients.	68
3.33 BadPixel Parameter dependence for post-processing SGM using alternative merging+thresholded gradients.	68
3.34 Discontinuity score Parameter dependence for post-processing SGM using alternative merging+thresholded gradients.	69
3.35 Iterative Improvement of scene <i>cotton</i>	69
3.36 Iterative SGM dependence on Number of Iterations	70
3.37 Iterative SGM dependence on Initial Delta with 10 iterations	70
3.38 Iterative SGM dependence on Initial Delta with 10 iterations	71
3.39 Iterative SGM dependence on Smoothing Parameter	71
3.40 Iterative SGM Images	71
3.41 Silicium Scan with iterative SGM	72
3.42 Truck clipping with iterative SGM	74
B.1 Truck with iterative SGM	82

D.2 List of Tables

3.1 Time	42
3.2 Resumee Table	75

E Bibliography

Hci benchmark dataset. <http://hci-lightfield.iwr.uni-heidelberg.de/benchmark/>. Accessed: 2018-010-16.

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

Website of the iwr heidelberg. <https://klimt.iwr.uni-heidelberg.de/Staff/swanner/>. Accessed: 2018-04-06.

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.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.

Edward H Adelson, James R Bergen, et al. The plenoptic function and the elements of early vision. 1991.

Josef Bigun. Optimal orientation detection of linear symmetry, 1987.

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

Youngjin Yoon In So Kweon Seon Joo Kim 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.

Blender Online Community. Blender—a 3d modelling and rendering package, 2014.

Maximilian Diebold. *Light-Field Imaging and Heterogeneous Light Fields*. PhD thesis, 2016.

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

Heiko Hirschmüller. 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.

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

Katrin Honauer, Ole Johannsen, Daniel Kondermann, and Bastian Goldluecke. A dataset and evaluation methodology for depth estimation on 4d light fields. In *Asian Conference on Computer Vision*. Springer, 2016.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017.

Bernd Jähne. *Digitale Bildverarbeitung*. Springer-Verlag, 2013.

Yaoxiang Luo, Wenhui Zhou, Junpeng Fang, Linkai Liang, Hua Zhang, and Guojun 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.

Zachary Moratto. Sgm implementation. <http://lunokhod.org/?p=1403>. Accessed: 2018-10-13.

Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera.

Hao Sheng, Pan Zhao, Shuo Zhang, Jun Zhang, and Da Yang. Occlusion-aware depth estimation for light field using multi-orientation epis. *Pattern Recognition*, 74:587–599, 2018.

Michael W Tao, Pratul P Srinivasan, Sunil Hadap, Szymon Rusinkiewicz, Jitendra Malik, and Ravi 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.

Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998.

Sven Wanner. *Orientation Analysis in 4D Light Fields*. PhD thesis, 2014.

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

Bennett Wiburn. *High performance imaging using arrays of inexpensive cameras*. PhD thesis, Citeseer, 2004.

Christian Wolf. 3d-object recognition and detection with convolutional neural networks, 2018.

Gaochang Wu, Belen Masia, Adrian Jarabo, Yuchen Zhang, Liangyong Wang, Qionghai Dai, Tianyou Chai, and Yebin Liu. Light field image processing: An overview. *IEEE Journal of Selected Topics in Signal Processing*, 11(7):926–954, 2017.

Zhimin Xu, Jun Ke, and Edmund 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)