Master Thesis

Comparison of Disparity Algorithms for Stereoscopic Video

Ben John

May 2016

Supervisor: Dr. Stephan Kopf

Department of Computer Science IV
Prof. Dr.-Ing. W. Effelsberg

School of Business Informatics and Mathematics
University of Mannheim

Contents

List of Figures v

List of Tables vii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Assignment . . . 3
- 1.3 Outline . . . 3

2 Foundations 5
- 2.1 Computer vision . . . 5
- 2.2 Stereo correspondence . . . 13
- 2.3 Disparity map between stereo images . . . 15
- 2.4 Disparity algorithms . . . 19
- 2.5 Sub-pixel accuracy . . . 26
- 2.6 Optical flow . . . 27

3 Related work 29
- 3.1 Semi-global matching . . . 29
- 3.2 ELAS: Efficient large-scale stereo matching . . . 31
- 3.3 Middlebury MRF library . . . 31
- 3.4 Disparity algorithms applied on videos . . . 37
  - 3.4.1 Spatiotemporal consistency . . . 37
  - 3.4.2 Remapping the disparity range of stereoscopic videos . . . 40

4 Implementation 43
- 4.1 Preliminaries . . . 43
- 4.2 Overview . . . 44
- 4.3 Integration of existing algorithms . . . 47
- 4.4 Fine-grained evaluation via masks . . . 50
- 4.5 Image diminisher to simulate real use cases . . . 54
- 4.6 Simple stereo matcher . . . 57
- 4.7 Web result viewer of the evaluation suite . . . 61

5 Evaluation and results **65**
5.1 Datasets . . . 65
5.2 Quality metrics . . . 69
5.3 Measurement . . . 70
5.4 Results . . . 73
5.4.1 Against reference dataset . . . 73
5.4.2 General performance . . . 76
5.4.3 Impact of noise . . . 81
5.4.4 Impact of video compression . . . 85
5.4.5 Runtime . . . 87
5.4.6 SVDDD . . . 88
5.5 Discussion . . . 92

6 Conclusion **95**
6.1 Thesis summary . . . 95
6.2 Outlook and future work . . . 97

Declaration of Honour **99**

Abtretungserklärung **101**

List of Figures

2.1 Example of an RGB raster image . . . 7
2.2 Binocular vision with horopter principle . . . 9
2.3 Epipolar geometry . . . 11
2.4 Epipolar geometry after image rectification . . . 12
2.5 Stereo matching on a 1D search space . . . 15
2.6 Tsukuba benchmark stereo image pair . . . 17
2.7 Depth calculation from disparity . . . 18
2.8 Basic processing flow of disparity algorithms . . . 20
2.9 Disparity space image . . . 22
2.10 Block matching along scanlines . . . 25
2.11 Sub-pixel estimation of a disparity value around adjacent pixels . . . 26
2.12 Optical flow estimation . . . 27

3.1 Simple undirected unweighted graph . . . 33
3.2 Stereo matching model by three coupled MRF's . . . 35
3.3 Spatiotemporal disparity refinement . . . 38
3.4 Examples of disparity maps . . . 40

4.1 Composition and processing pipeline of the implementation . . . 45
4.2 Architectural overview on the disparity interface . . . 48
4.3 Depth-discontinuity mask . . . 51
4.4 Textured regions recognition . . . 52
4.5 Ground-truth disparity maps . . . 53
4.6 Saliency detection . . . 54
4.7 Flow of the image diminisher. . . . 55
4.8 Gaussian normal distribution . . . 56
4.9 Flow of FFmpeg as image diminisher. . . . 57
4.10 Screenshot of overview page of the web result viewer . . . 61
4.11 Screenshot of sequences in the web result viewer . . . 62
4.12 Detail of one result in the web result viewer . . . 63

5.1 Tsukuba stereo dataset . . . 67
5.2 Cambridge stereo dataset example . . . 67
5.3 SVDDD high-resolution stereo dataset . . . 68
5.4 OpenCV autumn color scale . . . 72

5.5 Example heatmaps 72
5.6 SNSM heatmaps for Tsukuba scene 75
5.7 Chart of depth-discontinuity mask 77
5.8 Chart of textureless region mask 78
5.9 Heatmaps for the tunnel scene 79
5.10 Examples for salient masks 81
5.11 Chart of salient region mask 82
5.12 Chart of general outliers in a sequence 82
5.13 Examples for general outliers in a sequence 83
5.14 Chart of the impact of Gaussian noise 83
5.15 Examples for image diminishing effects with Gaussian noise . . . 84
5.16 Examples for image diminishing effects with video compression . 85
5.17 Chart of the impact of video compression 86
5.18 Example of computed disparity maps with video compression . . 87
5.19 Comparison of the runtime of different disparity algorithms . . . 88
5.20 Computed disparity map with negative disparity 89
5.21 Performance of SVDDD rabbit scene 90
5.22 Performance of SVDDD apple scene 91

List of Tables

2.1 Most common similarity measures 21

5.1 Identifier for results 74
5.2 Result table for reference dataset 75
5.3 Result table for general performance 76
5.4 Result table for general performance 77
5.5 Result table for tunnel scene 79
5.6 Result table for general performance of SVDDD 89
5.7 Result table for runtime of SVDDD 92

1 Introduction

1.1 Motivation

Computer vision establishes itself on the consumer market as more research is done. The upcoming iPhone supports this as it will feature a dual camera system [1]. In the year 2011 LG and HTC released the LG Optimus 3D[2] and corresponding the HTC Evo 3D[3]. Both had a stereo camera implemented and an auto-stereoscopic display attached. This enables one to view photos or videos taken in stereographic 3D without the actual need for additional peripheral like 3D glasses. Both can be seen as an experiment as there was no big distribution, Apple normally focuses on the mainstream consumer market, opening up the box of possibilities and the need for such algorithms even further. One example application for such a consumer-driven market could be the reconstruction of a face after taking a photo. There exist no method to reconstruct a whole 3D model without having stereo images from all angles of the face, but it is possible to trick the user in having captured a 3D photo. Another concrete example for an application regarding depth estimation in stereo videos is to detect moving people in a stereo video and calculate the distance to the camera of each person[4].

Obtaining depth information as additional data to infer intents from human gestures has arrived in mainstream computing with the release of Kinect at November 4th, 2010. Kinect is a hardware add-on for the Xbox video gaming console which enables users to interact visually with the console without actually using a controller or any other peripheral. The Kinect for Xbox one utilizes two cameras, one capturing colored and the other monochrome images. The monochrome sensor is used in combination with an infrared laser projector to obtain depth information via time of flight (TOF). Time of flight is a method to measure the time light needs to reach objects and then to calculate the distance.

[1] http://9to5mac.com/2016/02/03/sony-dual-cameras-iphone-7-plus/, 2016-02-22.
[2] https://en.wikipedia.org/wiki/LG_Optimus_3D
[3] https://en.wikipedia.org/wiki/HTC_Evo_3D
[4] http://de.mathworks.com/help/vision/examples/depth-estimation-from-stereo-video.html

With deducing intents from human gestures a step in the field of artificial intelligence was made as the computer is now able to interpret human body language. As this means processing an enormous stream of data (gathering and processing frame by frame) it represents a dataset of large and complex nature, also known as big data. This also implies the need for new data processing techniques in comparison with traditional ones. As a result one could say that computer vision is linked to both, artificial intelligence and big data. New applications which arose from the combination of those topics are for instance:

- robotic and autonomous driving,
- medical image analysis and automatic surgery,
- 3DTV and video compression.

Besides the technology of time of flight laser sensors - such as the Kinect[5] - there exists also the possibility to obtain depth information from stereo images by analyzing coherent images with so called disparity algorithms. Thus, it is sufficient to have two calibrated aligned cameras (a stereo camera) to acquire disparity information and calculate the depth at each point. But this leads to another fundamental problem of stereo matching: stereo correspondence. Basically, stereo correspondence means the labelling of pixels, i.e. which pixel of the left image belongs to the corresponding pixel on the right image as a projection of the same three-dimensional point from the captured world, projected to the image plane in every image. This problem of stereo correspondence has to be solved in order to actually match those and calculate the disparity. According to **?** , stereo correspondence is one of the most heavily investigated topics in computer vision [**?**]. As there is still a lot of research going on, no algorithm is working without any mistakes and also the runtime is a bit quirky, Microsoft Kinect established itself as a real alternative. This leads us to one of the disadvantages of Kinect sensor: Kinect is sensitive to other infrared sources (like sunlight) due to its nature of utilizing an infrared laser projector to acquire depth information, a stereo camera does not have this issue. Although using two coherent images also have some disadvantages which will be discussed later on, it is an alternative way to receive depth information. Especially thinking about autonomous driving during which at day a lot of sunlight is involved in, other techniques to estimate how far an object is away from one another are necessary to ensure a certain accuracy and fault-tolerance.

[5] Besides the consumer market, for autonomous driving or robotic research Velodyne is a well-known sensor.

1.2 Assignment

The thesis' main goal is to provide an overview of selected disparity algorithms for stereoscopic videos and evaluate those. **?** justified their tiny selection of disparity algorithms with the following: "Compiling a complete survey of existing stereo methods [...] would be a formidable task, as a large number of new methods are published every year." [**?**]. That said the ones with well documented source code and a research paper, also adaptable within the time scope of this thesis, were integrated. The main assignments can be summarized in three research tasks:

- Providing fundamental knowledge of existing stereo matching algorithms to have a basis for an advanced insight into the area of disparity algorithms targeting stereoscopic videos.
- Implementation of a stereo matcher for videos utilizing the OpenCV library. The implemented stereo matcher should be enhanced by using a spatiotemporal context
- Evaluation of the presented algorithms by implementation and presentation of an evaluation suite. Existing datasets, as well as a novel Dataset of the Department of Praktische Informatik IV[6] will thereby be examined using a set of defined quality metrics for assessment of the algorithms together with their runtime. Results are presented on a web-frontend.

1.3 Outline

The main purpose of Chapter 2 is to give an overview of terms and techniques used in this thesis. The following Chapter 3 focuses more on disparity algorithms and related work. To give an overview of state of the art algorithms a small summary of current used disparity algorithms is made. This will create the foundations for the later implementation. Chapter 4 describes the implementation and explains reasons for building an evaluation engine. The details of the implementation are explained afterwards. In addition, the integration of existing algorithms is illustrated. The evaluation engine was fed with datasets which are introduced in Chapter 5. This chapter also explains the used quality metrics and describes the resulting outcome. In the end, the results of this thesis are reflected in the concluding Chapter 6. Besides some future work is pointed out.

[6] `http://ls.fmi.uni-mannheim.de/de/pi4/`

2 Foundations

In this chapter the foundations for related work and the implementation are built. As a first step, computer vision is introduced with a short explanation how image representation works from a computer's perspective. Human visual perception is put in contrast to how computers perceive and interpret their environment. In addition, the labelling problem regarding stereo correspondence and the disparity between stereo images is illustrated. Furthermore, the depth calculation as well as the taxonomy of disparity algorithms is depicted. Finally, optical flow, a technical method that measures direction and movement of every pixel based on dominant movement in the original scene, is introduced to round this chapter up.

2.1 Computer vision

Computer graphics describes the terms and definitions of everything which has to do with basically treating images programmatically on a computer, interpreting and working with them. To give an example, the applications of computer graphics range from image representation, image creation, image transformations to applications of color models. Computer vision shares concepts from the domain of computer graphics, but works in reverse. Instead of modeling a scene and generating an image of it, computer vision optically measures the real world and tries to analyze it by applying models to the captured images. For instance, typical jobs are to get information out of an image, like image segmentation, edge detection, classification, and feature[1] point detection.

A simple example would be to imagine a face of a human being captured by a camera, which may produce errors due to lens distortion, shaky capturing, and sampling of the chip. Image editing would be useful to optimize the image by correction of contrast or brightness, cropping, or further adjustments. The tasks of computer vision are more in analyzing and understanding images for instance (just to name a few):

[1] Geometric shapes or more complex classifiers that are clearly recognizable.

- face localization to know the areas of faces on images,
- feature matching to detect the face on other images,
- feature tracking to track the movement of a person, or
- 3D reconstruction of a facial model.

Image representation

Two different methods exit of handling images on a computer. On the one hand, a vector image describes its content by representing forms like a circle, line, curve, or rectangle. The properties of these forms and shapes are also included, for instance, coloring, size, and origin. So a vector image basically contains those forms and shapes, their properties and a description of how they are all composed together.

On the other hand, it may become pretty complex using vector images to represent the real-world. In contrast to those there also exist raster images. Raster images (sometimes the term bitmap images is used) are a form to represent natural images, e.g. captured by a CCD[2] image sensor from a digicam. Capturing means sampling information on a matrix of light sensitive sensors to transform received signals into a matrix of color values of the same size.

Both types of images use a coordinate system to describe either the placement of elements (like written above with the properties size and origin of each element) or to describe how each point looks like. The coordinate system most widely used working with images starts in the upper left at the point $(0, 0)$, with the x-axis extending to the right and y-axis extending to the bottom. This can be seen as a grid system with the size of the image $width \times height$ representing $columns \times rows$. By describing how each point looks like the exact description of a pixel is meant [**?**].

One pixel in a grayscale image can range from $0 - 255$ describing the intensity of this pixel. 0 means black and 255 is fully white. In colored images a pixel can have more than one intensity value. More concrete, in a typical RGB[3] raster image each pixel contains three color channels, also called the RGB tuple. Thus

$$3 \cdot 1\ bytes = 3 \cdot 8\ bits = 3 \cdot 8 = 24\ bits$$

are stored per pixel utilizing RGB tuples. In C or C++ such pixel values are normally described as unsigned chars. A char represents eight bit and unsigned means

[2] CCD: charge-coupled device
[3] RGB: red-green-blue color channels

that it ranges from $0-255$ instead of -128 to 127. Sometimes RGB is used with an additional alpha channel specifying the degree of opacity, named RGBA[4]. The composition of these color channels orchestrate the final pixel value as it is obtained by, for example, an image sensor. Figure 2.1 depicts an example of a RGB raster image and shows the values of three pixels. The first marked pixel in figure 2.1 describes the RGB tuple with the following values $(237, 237, 237)$. Utilizing three color channels the final raster image then needs up to 24 bits per pixel, meaning an image the size $width = 300\ px$ and $height = 400\ px$ needs

$$300 \cdot 400 \cdot 24bits \cdot \frac{1\ byte}{8\ bits} = 360.000\ bytes$$

in memory. Images can be compressed with, for example, the JPEG algorithm but as the later implementation works only with pure raster images, as the unaltered values are examined, the amount of bytes as explained above is to be held in memory during the execution of the implementation.

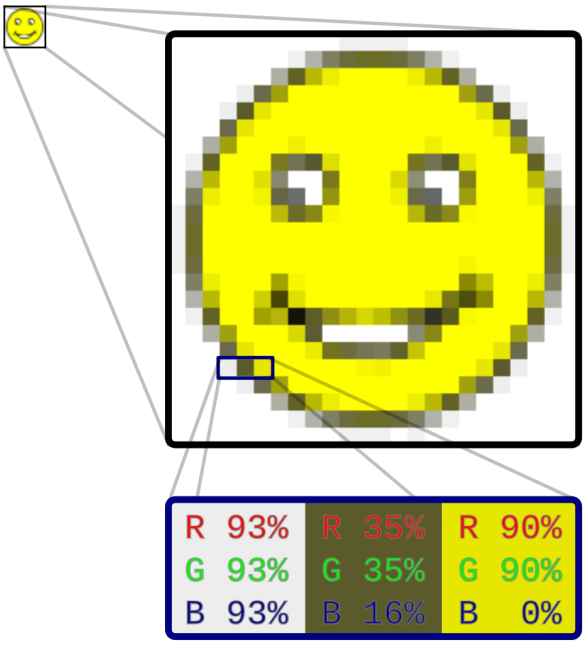


Figure 2.1: Example of a RGB raster image[5]

[4]RGBA: red-green-blue-alpha color channels
[5]Source (accessed 02/2016): `https://en.wikipedia.org`.

Human visual perception

In his manuscript 'Astronomia Pars Optica' from 1604, Kepler explains the use of both eyes for depth perception. He defined the term binocular as the composition of two latin words, 'bini' for double and 'oculus' for eye. With uniocular as 'uni' for one the sight with only one eye is meant. Binocular vision is then the vision creatures having two eyes obtain while using them together according to Kepler. According to **?** [**?**] and **?** [**?**] creatures with binocular vision have several advantages over creatures with only uniocular vision. Not to mention all but three, the most important ones which affect the depth perception:

1. Considering human beings, the second eye increases the field of view [**?**]. About 120 degrees are the binocular field of view (projected on both eyes) and two uniocular fields of view with about 40 degrees.

2. This also leads to another advantage with occluded, half-occluded or non-occluded objects [**?**]. Looking at Figure 2.2, the point P is in focus of the human being. Something directly behind this point may be fully occluded by the object in point P. Most of the things besides are non-occluded. Something behind this point P may be half-occluded if it can be seen by either the left or the right eye.

3. An advantage of having two eyes is the third-dimension human beings perceive, which leads to the binocular disparity or retinal disparity. Both terms are used in the literature and both mean the same: extracting depth information out of two coherent retinal images (obtained by the human eyes) [**?** **?**].

Figure 2.2 depicts the mapping of the three points R, P and Q on the retina of each eye. The letter F stands for *foveae* in which the visual axis ends. The eye is constructed out of photoreceptor cells, mainly rods and cones. The rods are necessary for seeing at night while the cones are responsible for humans being able to see the world sharp. In the foveae is the peak of cones and it contains very few rods. This means that the human visual system works the way that the visual axis joins the point of fixation with the foveae. This can be seen in Figure 2.2 as the lines between F of each eye to the point of fixation P. Both eyes should be brought into convergence that the point of interest is projected onto the foveae of each eye. Everything on the horopter (the circle) is corresponding (e.g. P and Q), all points other than being on the horopter are non-corresponding (R) in terms of retinal disparity.

In the later described disparity algorithms which act like a tool for computers

to be able to see the shift of pixels from the left image to the right image, the human being does somehow the same. Humans experience the depth which is sensed unconsciously by the eyes and calculated by the brain in real-time. With two eyes basically two slightly different images are obtained. The brain acts as the computer which puts both coherent images together and extends the two-dimensional space into a three-dimensional space and calculates the position of the objects in the z-axis.

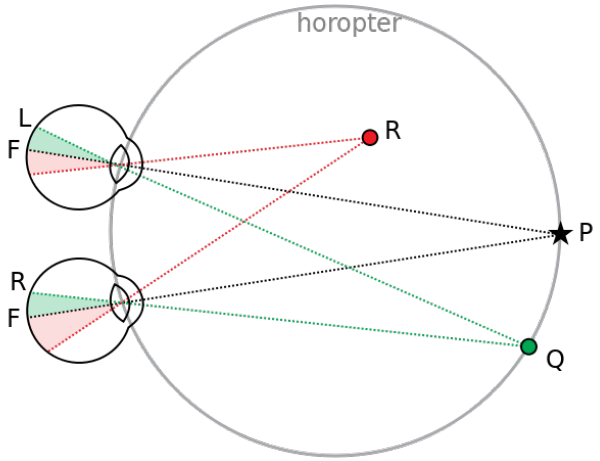


Figure 2.2: Binocular vision with horopter principle[6]

In contrast to the visual perception human beings perceive, computers need to do several steps to obtain disparity and calculate depth information:

- identify objects,
- identify layers,
- match objects / pixels in both images,
- calculate the shift of the pixels from left to right, and
- obtain final depth values.

The human brain enables human beings to see and experience the real-world three-dimensional. A computer has to be programmatically instructed to identify and group objects in both images [**? ?**]. This is not an easy task as can be seen in the next sections.

[6]Source (accessed 02/2016): `https://en.wikipedia.org`.

Stereoscopy

To paint the bigger picture, stereoscopy and the illusion of depth are introduced. Stereoscopy is sometimes linked to the phrase 'the illusion of depth' as it is a technique used to add a third dimension to a flat image to simulate depth [**? ?**]. Specifically, the goal is to show each eye a slightly different image and thus achieve depth perception in our brain. With so called stereoscope or special glasses depth perception can be transferred to the consumer in a cinema or at home via showing each eye a different image which then is composed to the final spatial perception.

There exist several techniques to create the stereoscopic effect. One of these glasses is the shutter system. The concept of a shutter glass is that it cycles a block (meaning only one eye is dispatched to the screen) with a certain frequency (usually about 120 fps[7], resulting in 60 fps per eye) synchronously with the 3DTV. This means that only one specific image is passed to exactly one of the consumers eyes. So each eye is shown about 60 fps which naturally is experienced as flicker-free. The older anaglyph 3D technique uses multiplied images tinted with red/cyan to filter out the respective image by the glasses filter foil, thus only one image is dispatched to one specific eye at a time. Nowadays the anaglyph 3D technique is sometimes used in magazines to show 3D graphics. As all techniques are not representing the real-world and the depth perception can be adjusted with for instance camera positioning (image one would reposition his eyes to perceive the real-world differently) they can be summarized as the illusion of depth.

Epipolar geometry

The geometry of stereo images, called epipolar geometry, plays an important role in understanding the mathematical equations in the upcoming section. The most important terms of epipolar geometry are:

- image plane,
- baseline,
- epipole,
- epipolar line, and
- epipolar plane.

[7] frames per second

The *image planes* in Figure 2.3 and Figure 2.4 are the blue surfaces which represent the captured image through the cameras O_L and O_R. The *baseline* is the line joining both camera centers with the image plane. Focusing on the figures, the baseline is the line going from O_L to O_R, as O reflects the origin (camera center). An *epipole* is the joint of the baseline with the image plane, referring to the symbols e_L and e_R. The *epipolar plane*, visualized as green triangle in Figures 2.3 and 2.4, is determined by point X and both origins O_L and O_R. It is the surface reflecting the z-axis, the depth. An *epipolar line* then is the intersection between the origin to the point of interest, in this particular case X, which lies on the epipolar plane and intersects the image plane.

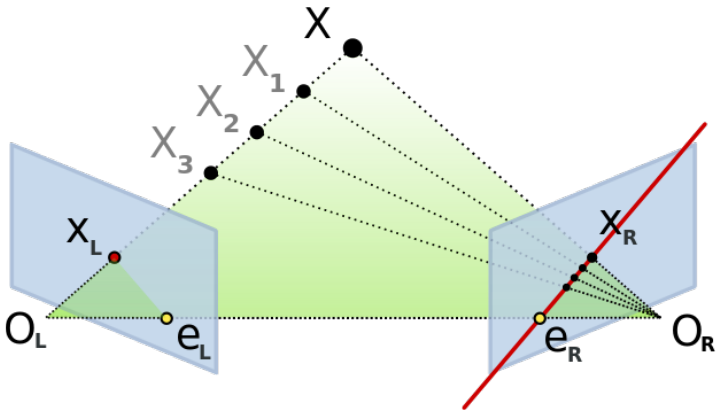


Figure 2.3: Epipolar geometry[8]

This results in an epipolar constraint [**?**]: Each image point X_i of a space point in the image plane, e.g. consider point X in Figure 2.3 must lie on the corresponding epipolar line $O_L\vec{X}$. More concrete focusing on Figure 2.3: this constraint states that the correspondence for a point on the epipolar line $O_L\vec{X}$ must lie on the line $e_r\vec{X}_r$. As seen above Figure 2.3 depicts the left and right view of an object in point X.

The Figures 2.3 and 2.4 both illustrate the epipolar geometry on a pair of unrectified images and the result after the rectification was done. Rectification[9] is necessary to reduce the search-space from two-dimensional to one-dimensional. For determining the exact position of X (possible positions X_i with $i = [1 \dots 3]$)

[8] Source (accessed 02/2016): `https://en.wikipedia.org`.

[9] Affine transformation (rotation and translation) neglecting geometric distortion to rectify the images.

the diagonal has to be scanned in the unrectified image. In the rectified image only the horizontal needs to be investigated. In the further proceeding this line is called the scanline which most of the algorithms operate on [**?** **?**]. After the rectification process the following two statements come true:

- Epipolar lines are parallel to the x-axis (horizontal).
- Corresponding points are on the same y-axis (vertical).

Implicitly the following two assumptions were made:

- the focal length f of both cameras which captured the images are the same,
- the origin of one camera is the so called camera principal point (the joint of the optical axis with the image plane and the fovea counterpart) [**?**].

In conclusion, corresponding points are constrained to be on the same line and thus depth can be inferred by using triangulation and camera parameters. Based on this, the investigations of stereo correspondence and the actual depth calculation using triangulation is discussed in more detail in the next sections.

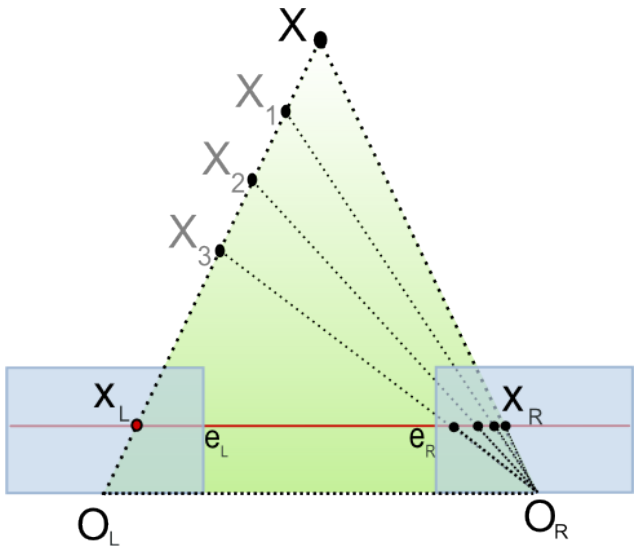


Figure 2.4: Epipolar geometry after image rectification[10]

[10]Source (accessed 02/2016): `https://en.wikipedia.org`.

2.2 Stereo correspondence

Stereo correspondence can be seen as a pixel labelling problem [**?** **?**] between two (or more) stereo images. The essential problem is to find corresponding pixels in images of different cameras and needs to be solved. Without knowing which points belong to each other in two separate stereo images, no conclusions can be drawn for instance calculating the disparity. Henceforth, while talking about images for stereo matching, rectified images are implicitly meant. If images are existing in an unrectified unaligned version then as a preprocessing step the images will be rectified.

Constraints

Stereo matching algorithms rely on several assumptions about the real-world. From the pioneer work of **?** [**?**] the following constraints can be reasoned which are important for the development of such algorithms (also cf. [**?** **?** **?**]):

As a short remark: X_i is a point on a scanline in image i (i can be replaced with either L for left or R for right side) and thus only the x-position is mentioned.

Uniqueness
: As each pixel on a surface has one unique physical position in space, each pixel from each image has at most one disparity value.

Continuity
: The term smoothness constraint is also mentioned in the literature. Disparity can vary but smoothly almost everywhere in an image except at object boundaries which represent a discontinuity in depth, i.e. the difference of adjacent points should be small $||X_{L_1} - X_{R_1}| - |X_{L_2} - X_{R_2}|| < \varepsilon$.

Epipolar
: Recapture of the epipolar constraint from the section before: corresponding image points have to lie on the corresponding epipolar lines. If the epipolar lines are known to be parallel to the x-axis, the search space can be reduced to a 1D search space along the epipolar lines.

Ordering
: Following up the epipolar constraint: if the epipolar lines run in parallel to the x-axis, multiple consecutive image points have to lie on the same corresponding epipolar line in the same ordering.

Limit
: There is a defined disparity maximum (limit) d_{max} holding $|X_L - X_R| <$

d_{max}, defining the maximum disparity value which can be found in a stereo image. Hence, $d(x, y)$ is in the range $[0 \dots d_{max} - 1]$.

Lambertian
Algorithms for stereo matching also rely on the assumption of opaque lambertian surfaces, meaning a surface that reflects light equally into all directions and thus appears equally bright independent from where light is coming and where the camera is placed. Thus the algorithms can expect the intensities and colors of corresponding points to be almost the same.

Besides those constraints there also exist some common pitfalls which can disturb the result of algorithm.

Common pitfalls

Algorithms are using different metrics to analyze similarities in images along scanlines, in whole areas, or at a global view to then estimate the disparity. This can be challenging especially considering the upcoming traps. On the one hand, potential issues from the camera setup can be challenging, such as:

- photometric distortions,
- noise,
- calibration error of the cameras.

On the other hand, the scenery can be tricky:

- specularities and reflections,
- transparent objects,
- matching ambiguity,
- occlusions (missing data) and discontinuities.

These issues also challenge the algorithms to stereo match the pixels correctly. With matching ambiguities, constant or low-contrast regions are meant. A good example for that are textureless regions or repetitive structures. Textureless regions could contain a small set of matching pairs of pixels, other pixels of that region could be erroneously assumed the same. The presented constraints support the algorithms regarding those pitfalls.

Simplified stereo matching

Figure 2.5 depicts a simplified example of how stereo matching works on a one-dimensional search space: there exist two arrays with $length = 5$, one in the left and one in the right image. Assuming the top row $[p \dots t]$ reflects one row in the left image. The bottom row $[u \dots y]$ accordingly the same row in the right image. The pixel p, q, w and y are unmatched, e.g. occluded. Having a function $d(z)$ which returns the disparity for a given element z in those arrays, $d(r) = -2$ means the shift two to the left. Accordingly $d(s) = -2$ and $d(t) = -1$.

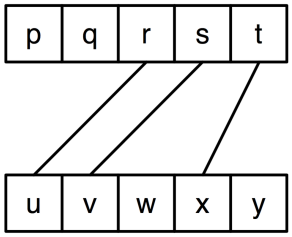


Figure 2.5: Two arrays illustrating stereo matching on a 1D search space [**?**].

Up to this point the epipolar geometry and the challenge with stereo matching were introduced. The upcoming section defines disparity, illustrates the disparity map, and the depth calculation.

2.3 Disparity map between stereo images

In the last two sections epipolar geometry and the problem with stereo correspondence were introduced. In this section the focus is on the term disparity, how disparity can be visualized via disparity maps, and how the depth can be calculated out of those disparity values.

Disparity

The disparity is the shift of a pixel / object (feature) between two or more images. An object may appear at position (x_1, y_1) in the left one and at position (x_2, y_2) in the right one. The disparity is the shift from the left position to the right one. With P_i declaring a point, left or right side, the following represents the disparity for two points in a two-dimensional space utilizing the pythagorean theorem:

$$D(P_L, P_R) = \sqrt{D_X^2(P_L, P_R) + D_Y^2(P_L, P_R)} \tag{2.1}$$

Henceforth, as the assumption of rectified images was made, only the horizontal disparity D_X is meant by the term 'disparity'.

$$D_X = |X_1 - X_2| \tag{2.2}$$

In other words, having a pixel (x_1, y_1) in a reference image (left) l and a pixel (x_2, y_2) in our matching image (right) r the correspondence is given by:

$$x_2 = x + |d(x_1, x_2)| \quad \text{with} \quad y_1 = y_2, \tag{2.3}$$

where $d(x, y)$ is the function which delivers values out of the disparity space (x, y, d) computed by the algorithms.

Resulting in matching pixels from one image to another, the disparity for each pixel-wise combination is calculated as seen in the previous subsection (simplified stereo matching) and presented here. Such disparities can also be seen as the inversed distances to observed objects. As a matter of fact, at the border of each image some pixels can not be calculated caused by the non-existing counterpart for matching. Those pixels with no fellow are called 'occluded' pixels. For example, in some cases pixels are hidden in one image by an object due to the blocking line of sight of this object.

Disparity map

In order to actually analyze the output of algorithms ground-truth data is necessary. An algorithm normally outputs a disparity map reflecting the disparity space (x, y, d). This disparity map can be seen as matrix having the size of the original image ($m \times n$) and containing values ranging from 0 to $d_{max} - 1$ utilizing one color channel (grayscale). The maximum disparity can be set via parameter for most of the algorithms and a feasible value which yields to sound results is 64. For better visual analysis the disparity maps are usually normalized to values ranging from $0 - 255$ [**? ? ?**]. Figure 2.6 c) shows the ground-truth data representing the disparity map. The disparity map depicts grayscale intensities with lighter gray representing pixels / objects closer to the camera.

Tying in with the term ground-truth **?** created the first "highly realistic CG dataset that properly models real-world imperfections, while providing accurate ground truth." [**?**]. Without such datasets bad evaluation of stereo matching algorithms can be made as there would been no reference to evaluate against. Figure 2.6 shows the previous dataset of the University of Tsukuba, the well-known *Head and lamp scene.*

(a) left input image (b) right input image (c) ground-truth data

Figure 2.6: Tsukuba benchmark stereo image pair of the University of Tsukuba [**?**].

The input for a perfect algorithm would be the reference image (a) and the matching image (b). After computation the result would be similar to the ground-truth data (c). With evaluation metrics the computed disparity map is then compared to the ground-truth data. Measuring instruments serve as a quality indicator for an algorithm's performance. The above example is given for basic knowledge and understandability of how a disparity map actually looks like. Details of this process are examined in the evaluation Chapter 5.

Depth calculation

From an obtained disparity map and given camera parameters the depth can be calculated. The mathematical description of the following equations has been introduced by **?** in Chapter 3.4.9 (Depth Resolution in Stereo Setups) of their book "*An introduction to 3D computer vision techniques and algorithms*" [**?**]. Assuming the focal length of the camera's lens and the baseline[11] are known, the following holds:

$$Z = \frac{f \cdot B}{d} \quad \text{and} \quad d = \frac{f \cdot B}{Z} \tag{2.4}$$

$$X = \frac{x \cdot Z}{f} \quad \text{and} \quad Y = \frac{y \cdot Z}{f} \tag{2.5}$$

where:

- Z is the distance along the z-axis (camera axis),
- f is the focal length,
- B is the baseline (in meters),

[11] The distance between both image sources.

- d is the disparity of the point.

After Z is determined, X and Y can be calculated using the usual projective camera equations (2.4-2.5) where the point (x, y) is the pixel location in the 2D reference image and (X, Y, Z) describes the real 3D position [**? ? ? ?**]. Figure 2.7 depicts the depth calculation from disparity with X being the estimated point and $d = x - x'$.

The following subsection describes the more general steps of how disparity algorithms work, known as the taxonomy.

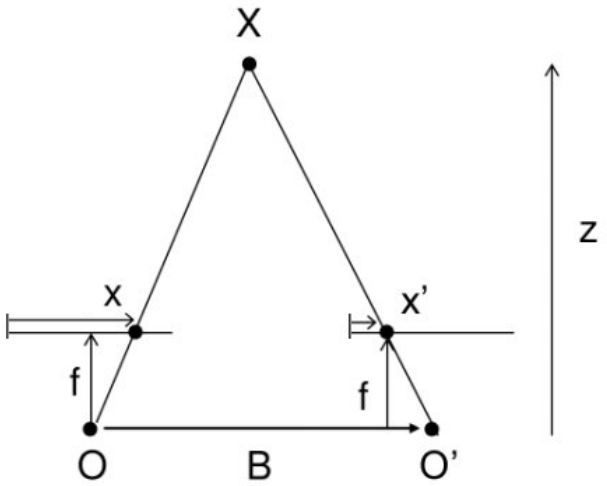


Figure 2.7: Depict of depth calculation from disparity[12].

[12]Source (accessed 02/2016): `http://docs.opencv.org`.

2.4 Disparity algorithms

In the last sections different aspects affecting stereo matching were introduced. To get a better understandability of the algorithm's technique, this section focuses on the diversity and taxonomy of disparity algorithms.

Diversity of disparity algorithms

A lot of different algorithms exist and their workings differ slightly. According to **?** [**?**], **?** [**?**], the following categories to separate disparity algorithms exist. Some of these classifications are discussed in more detail the related work Chapter 3.

First, the output of an algorithm is rated: they can create sparse or dense disparity maps. On the one hand, most of the algorithms produce a dense disparity map meaning that almost every pixel got a corresponding shift value. On the other hand, sparse algorithms only calculate values around, for instance, feature points (cf. feature matching). One advantage of sparse algorithms compared to dense disparity algorithms is that they are normally faster in computation but limited in applications. Approaches to interpolate sparse disparity maps into dense disparity maps exist, but they tend to produce inaccurate results in comparison to dense algorithms.

Second, **?** [**?**] categorize direct and indirect methods. Indirect methods are feature based or operate in the transformed image space (cf. Chapter 6.3.7 in [**?**]). Direct methods use intensity based measures.

Finally, disparity algorithms are classified into local and global methods:

- Local Methods
  - Feature matching
  - Block (area) matching
- Global Methods
  - Belief propagation
  - Graph cuts
  - Dynamic programming
  - Layering (hierarchical scale-space)

Taxonomy of disparity algorithms

Assumptions need to be made before starting to describe the taxonomy of disparity algorithms:

1. The algorithm is fed with a pair of rectified images as input.
2. The algorithm produces a dense integer disparity map, which means that disparity is estimated at each pixel.
3. Most of the current algorithms works according to the following steps (see Figure 2.8)

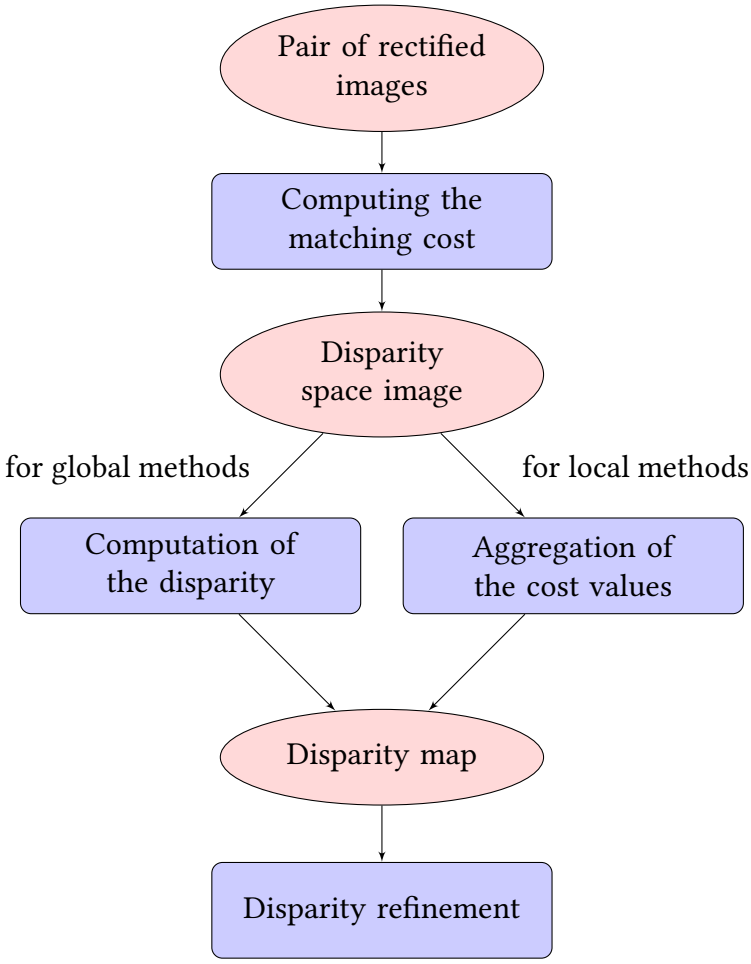


Figure 2.8: Basic processing flow of typical disparity algorithms, cf. [**? ?**].

The upcoming subsections discuss the steps in more detail, especially regarding the computation of the matching cost and the subsequent aggregation.

Matching cost functions

At first, the similarities of pixels in both images are calculated. In general, the literature shows matching cost as the dissimilarities of pixels. The matching cost needs to be computed for the decision which pixel belongs to another. Hence, the cost needs to be low for similar pixels. Some of the matching criteria used for determining the matching cost are described in Table 2.1 cf. [**? ? ? ? ?**].

Method	**Formula**
Sum of absolute differences	$\sum_{i,j \in U} \lvert I_1(x_L + i, y_L + j) - I_2(x_R + i, y_R + j) \rvert$
Sum of squared differences	$\sum_{i,j \in U} (I_1(x_L + i, y_L + j) - I_2(x_R + i, y_R + j))^2$
Normalized cross-correlation	$\frac{1}{n} \sum_{x,y} \frac{(I_1(x_L, y_L) - \overline{I_1})(I_2(x_R, y_R) - \overline{I_2})}{\sigma_{I_1} \sigma_{I_2}}$

Table 2.1: Most common similarity measures

$I_k(x, y)$ stands for an intensity value of the image k at the point with given coordinates (x, y). The set $U = U(i, j)$ describes close-by points located around the point (i, j). The sum of absolute differences (SAD) similarity measure is one of the simplest ones and describes the difference between pixel values. The absolute intensity differences of both images I_1 and I_2 are summed up for all adjacent pixels in the neighborhood (described with U). Zero stands for the equality of both regions. In optimal images nearly every pixel in the left image should have a corresponding pixel in the right image, fulfilling the constraints from the section before, and thus the calculated SAD should sum up to zero. The lower the result, the more similar the pixels and the cheaper the matching cost are.

In the sum of squared differences (SSD) similarity measure the pixel differences are squared and summed up. This measurement needs a bit more computational power and is usually chosen to discriminate high differences. It can yield to better results if outliers need to be excluded and the difference is not strong enough while using SAD.

There also exist the normalized cross-correlation (NCC). Cross-correlation measures the correlation between two intensity values in a point (x, y). The normalized cross-correlation subtracts the mean $\overline{I}$ of the intensities and divides by the standard deviation σ_I to normalize the intensity values. This may be necessary to balance brightness variations. NCC is excluded in most scientific investigations regarding disparity algorithms as it behaves similar to SSD (cf. [**? ? ?**]).

Disparity space image

Related to the disparity space introduced in the section before, the disparity space image (DSI) should be defined. The DSI is an image or a function over a continuous or discretized version of the disparity space (x, y, d) and represents the matching cost (i.e. the dissimilarity) of a given $d(x, y)$. It can be imagined as a three-dimensional matrix with the x-axis meaning the column, the y-axis the disparity and each combination the matching cost for that particular value as the z-axis. The disparity space image $C(x, y, d)$ is the result of the matching cost values over all pixels and all disparities, where the function C that denotes the matching cost for the given input parameter. This leads to the aggregation step, during which the matching cost form the final disparity for local methods.

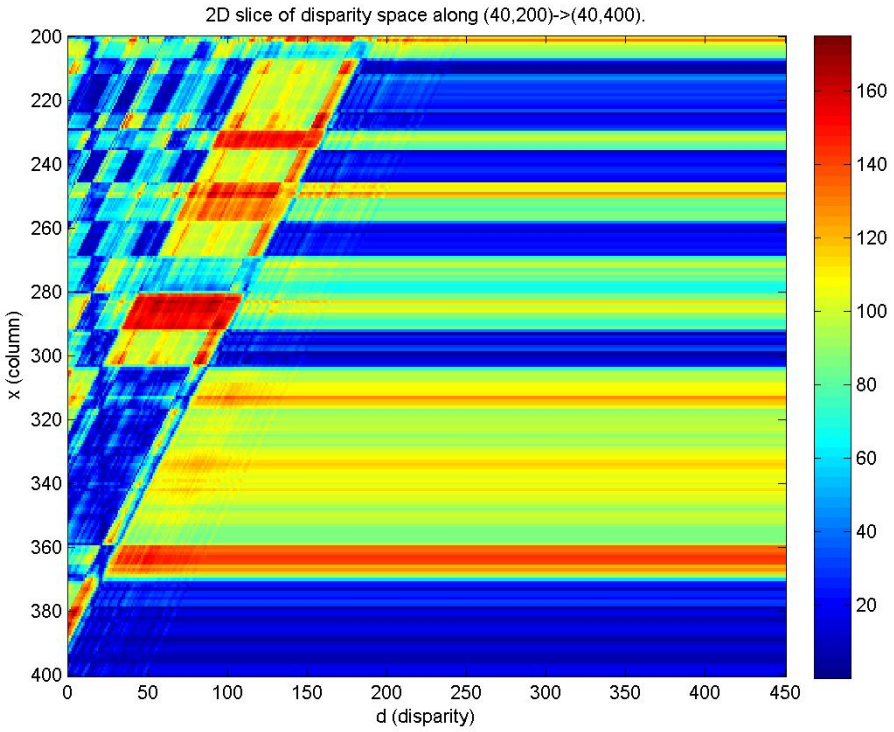


Figure 2.9: Illustration of a disparity space image.[13]

[13] Source (accessed 03/2016): `http://www.cs.virginia.edu/~cab6fh/CV_4/WRITEUP.html`.

Aggregation

In the aggregation step the decision has to be made, which discrete set of disparities represents the scene best [**?**]. As the matching cost values over all pixels and all disparities are stored in the DSI the minimum for each row is chosen as the best matching pixel and thus declared as the corresponding pixel. In other words: for every pixel the disparity with the lowest cost is selected. This strategy is known as the winner takes it all (WTA). [**? ?**]. As the pixel with the lowest cost is chosen, the following holds:

$$d(x, y) = \arg\min_{d'} C(x, y, d'). \tag{2.6}$$

Disparity computation

After the aggregation, the actual disparity is computed in this step. It is split up for the two different methods: local and global.

(i) Local methods. Local methods focus on the matching cost computation and the cost aggregation steps. The final disparity computation is trivial as the minimum cost value (least dissimilarities) over each row is chosen (WTA).

(ii) Global methods. In contrast to local methods, global methods unify the three basic steps into a single one by defining an energy function to be minimized. It ties in with the labelling problem [**?**]. A row in the DSI can be imagined as the different labels (i.e. disparity values) one pixel can receive. The labelling problem describes the search of the disparity as the choice of the correct label. Each pixel should only have one label assigned in the end.

Let P be a set of pixels and D a set of disparities. The energy function aims to find a disparity d which minimizes some energy:

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d). \tag{2.7}$$

The data term $E_{data}(d)$ defines the matching cost for a given disparity function d and expresses how well the disparity function d matches with the input image pair. C is the matching cost DSI:

$$E_{data}(d) = \sum_{(x,y)} C(x, y, d(x, y)). \tag{2.8}$$

As each pixel should be matched to a good find in the other image but simultaneously the adjacent pixels should be normally piecewise smooth, i.e. about the

same value / intensity, the smoothness term $E_{smooth}(d)$ is introduced to reflect that (cf. stereo correspondence constraints). The λ is introduced to control how much the smoothness term should influence the overall data term. To make the smoothness term computationally affordable it is, depending on the algorithm, usually restricted on the differences between adjacent pixel disparities [**?** **?**], i.e. the disparity gradient:

$$E_{smooth}(d) = \sum_{(x,y)} p(d(x,y) - d(x+1,y)) + p(d(x,y) - d(x,y+1)), \quad (2.9)$$

where p is a "monotonically increasing function of disparity difference" [**?**]. Depending on the used algorithm, other smoothness term functions exist. The optimization problem to solve is defined as the minimization of the energy function, i.e.:

$$D = \arg\min_{d} E(d), \quad (2.10)$$

where D is the disparity map containing the final values for every (x, y) and d a set of parameters or a disparity function affecting the energy value.

The search space for finding a solution is large, as an $n \times m$ image with k disparities has about $k^{n \times m}$ possible solutions. According to **?** [**?**], **?** [**?**] finding the global minimum is *NP-hard*. The related work in Chapter 3 gives an introduction into solving those optimization problems.

Disparity refinement

Disparity refinement can be seen as an optional post-processing step some algorithms perform automatically or may be requested manually. Refinement steps can also be implemented independently from the algorithm as they are executed on the final disparity maps. Sometimes the literature mentions those as clean-up steps. Here is a list of some known refinement steps:

- Sub-pixel estimation for higher accuracy.
- Disparity verification with left-to-right and right-to-left disparity map comparison (can also detect occluded areas).
- Filtering of disparity values, for instance using a median filter to remove mismatches.
- Interpolation of missing values: can be necessary when using an algorithm which produces a sparse disparity map.

Simplified block matching

For demonstration purpose of a working local disparity algorithm, block matching, also known as area matching, is sketched in a simplified version. The following algorithm assumes rectified images. Thus, the algorithm is executed along the scanlines.

1. Divide the images in blocks of the size $m \times n$ (e.g. 8×8).
2. Find the corresponding block along the scanlines as shown in Figure 2.10, i.e. the block with the lowest matching cost (e.g. sum of absolute differences).
3. Calculate for this block the displacement (the shift from left to right image) which results in the disparity.
4. This yields in the final, ideally *bijective*[14] disparity map after finding the corresponding block from the left to the right image and vice versa. If a block could not be matched the bijective criteria is not fulfilled.

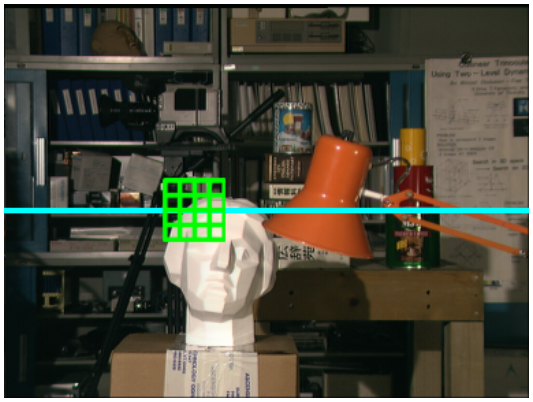


Figure 2.10: Illustration of block matching along a scanline.

In general, block matching leads to more accurate results with smaller window sizes. A bigger window leads to more smoothing which results in lower noise. The window size depends on the image size and its content. Therefore, no general assumption can be made. For each scenery the window size should be adjusted individually.

[14] Considering two sets, for each element of the first set a corresponding element of the second set is found. It also holds that both sets contain the same amount of elements. Thus, it is a one-to-one correspondence which also works inversely.

2.5 Sub-pixel accuracy

As seen in the sections before, the disparity algorithms produce a disparity map consisting of integer values only. For most of the imaginable applications integer values should be enough. However, the world is continuous and there are applications which rely on accurate disparity estimations. For instance, having no sub-pixel values, image-based rendering produces an image for visualizing the disparity map, which can appear to be made up of many thin shearing layers [**?**]. To get an accurate sub-pixel value, the most common technique is to use curve fitting by utilizing an *n-th* polynomial-order function. In this particular case [**?**], a second-order polynomial function, i.e. a parabola is used. The curve is fitted around three or more values of the matching measure. The point of interest lies in the center of the chosen window (as Figure 2.11 depicts). The minimum of this parabola is the searched value [**? ?**].

Curve fitting with a second-order-polynomial in Figure 2.11 works with three data points: (d_{i-1}, m_{i-1}), (d_i, m_i) and (d_{i+1}, m_{i+1}). d_i is the found integer value for disparity and m_i is a match value for the displacement d_i. With the curve fit a new minimum value d_x is found which no longer needs to lie on the integer grid.

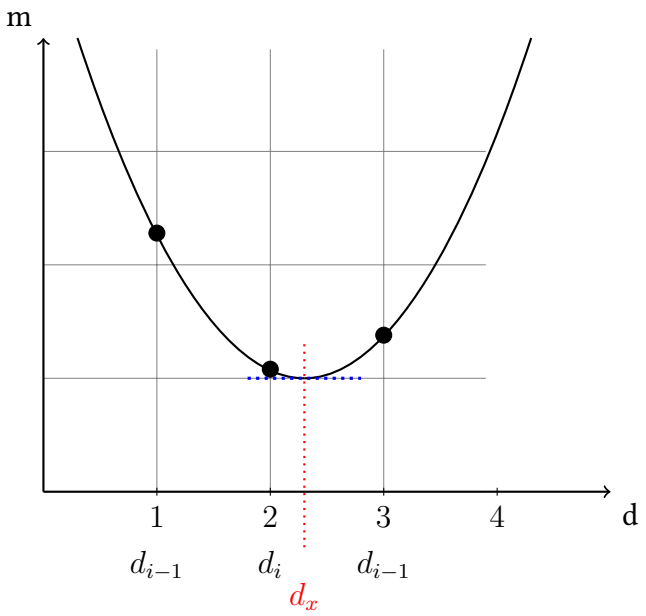


Figure 2.11: Sub-pixel estimation of a disparity value around adjacent pixels.

2.6 Optical flow

Similar to the problems discussed in the section before, optical flow is also an image matching problem. The optical flow is defined as vectors describing small local displacements like moving objects or camera motion between two consecutive frames [**?** **?**]. The principle of the matching problem of images is comparable to disparity algorithms. The main difference is that instead of analyzing left and right image, a scene is investigated and the disparity describes small local vectors. To be more precise, the optical flow relies on the assumption that a certain point (x_1, y_1) in a frame at time t_1 will be matched to a point (x_2, y_2) in a frame at time t_2. Different approaches for estimating the optical flow of pixels exist, like:

- Correlation or block-matching,
- feature tracking,
- energy-based methods, or
- gradient-based methods.

Optical flow is heavily used in autonomous driving, automated traffic surveillance systems and video compression like H.264 [**?** **?** **?** **?**]. Recently a dataset containing ground-truth data of real-world sceneries regarding optical-flow information was released by **?** . Figure 2.12 shows an example of estimating the movement of a vehicle. In the left image of Figure 2.12 most of the vectors are null as no local displacement can be estimated. Only a few vectors (small white dots) near the vehicle illustrate the displacements.

Figure 2.12: Optical flow estimation to obtain motion vectors (left) and pixel velocity magnitudes (right).[15]

[15]Source (accessed 02/2016): `http://de.mathworks.com/discovery/optical-flow.html`.

3 Related work

In this chapter, the related work regarding disparity algorithms is treated. As integration of some disparity algorithms for the later evaluation is part of this thesis, the ones which were actually implemented are examined in more detail. The well-known semi-global matcher by **?** , also implemented in the OpenCV library [**?**], is introduced. OpenCV[1] is an extensive image processing framework, with the main goal towards real-time computer vision. **?** introduce an approach that enables fast matching of high-resolution images, which is also outlined in the upcoming section.

Both approaches utilize local methods for estimating disparity maps. One candidate adopting global methods is the Middlebury MRF library, which is also introduced. It implies solving optimization problems, i.e. the minimization of a global energy cost function. The library's implemented methods to solve such optimization problems are outlined in greater detail. In the end, an outlook on disparity algorithms on stereoscopic videos is given, which includes an approach towards spatiotemporal consistency and remapping of the disparity range.

3.1 Semi-global matching

? combines two different methods, global- and local-matching for determining accurate disparity at a lower runtime as other global algorithms, which are time consuming even on current hardware [**? ?**].

The semi-global matching (SGM) method utilizes pixel-wise matching of so called mutual information (MI) via entropy H. The joint entropy of two images I_1 and I_2 results from the sum of their combined entropy and a global two-dimensional smoothness constraint H_{I_1,I_2} which leads to the following cost:

$$MI_{I_1,I_2} = H_{I_1} + H_{I_2} + H_{I_1,I_2}. \tag{3.1}$$

The discussed one-dimensional constraints from Chapter 2 are applied as well. Calculating the matching cost based on mutual information is insensitive to differ-

[1] `http://opencv.org`

ent video recording conditions and illumination changes [**?** **?**]. The joint entropy H_{I_1,I_2} is low (meaning low information content) for rectified images as one image can be predicted by the other. The MI matching cost is defined as the following:

$$mi_{I_1,I_2}(i,k) = h_{I_1}(i) + h_{I_2}(k) - h_{I_1,I_2}(i,k), \tag{3.2}$$

where h_1 and h_2 are calculated from the probability distribution of corresponding intensities. Thus, $h_{I_1,I_2}(i,k)$ serves as the matching cost for the two intensities i and k. The idea is, that one image needs to be warped[2] such that corresponding pixels are at the same location in both stereo images:

$$I_1 = I_b \quad \text{and} \quad I_2 = f_D(I_m), \tag{3.3}$$

where I_b is the base image, I_m the match image and $f_D(x)$ is a function which outputs the matching corresponding point. As the matching cost represents the information content of two intensities I_1 and I_2, which should be low (i.e. matching parts are as equal as possible), the disparity map D needs to be known *a priori* for warping. Hence, the MI matching cost needs to be calculated either iteratively or hierarchically. On the one hand, an iteratively approach utilizes a random disparity image for calculating the MI matching cost, which serves as the base for the next iterations. On the other hand, the MI matching cost can be calculated hierarchically by recursively using an up-scaled disparity image, which has been calculated at half resolution with a common similarity measurement like SAD. For a deeper explanation of how the mutual information is exactly calculated and used in the SGM method compare [**?** **?** **?** **?**].

OpenCV BM and SGBM

The OpenCV library [**?**], currently at version 3.1.0, offers two implementations for disparity estimation, block matching and semi-global block matching based on the idea of **?** . This latest version also contains a new filter, which was initially introduced with version 3.0.0, named *Disparity WLS Filter*[3]. WLS stands for weighted least squares (in the form of a fast global smoothing algorithm). This disparity filter smoothes the disparity and also performs a left-right-consistency check to refine the results in especially half-occluded and uniform areas [**?**]. This yields to better and more accurate results but has the drawback of loosing negative disparity values. Negative disparity appears if the stereo cameras are verged or

[2] In this context warping can be seen as a function which maps pixels from the destination image to pixels in the original image. Then the pixels are copied at the mapped position to the coordinates in the destination image.

[3] `http://docs.opencv.org/3.1.0/d9/d51/classcv_1_1ximgproc_1_1DisparityWLSFilter.html`

inclined towards each other. The WLS filtering results in disparity ranging from 0 to D_{max}, which is set beforehand as a parameter. Thus, unknown disparity is denoted by -1.

3.2 ELAS: Efficient large-scale stereo matching

? proposed a novel approach for estimating the disparity with so called support points [**? ?**]. ELAS summarizes a command-line interface for the cross-platform library for efficient large-scale stereo matching and the library itself. According to them, "it is robust against moderate changes in illumination and well suited for robotics applications with high resolution images" [**? ?**]. A support point is like a feature, a point which can be robustly matched. For those support points, a sparse disparity map is calculated. For more robustness, only the support points which can be matched left-to-right and right-to-left are retained. To remove ambiguities, the ratio between the best and the second best match of all points is taken into account. If the ratio exceeds a fixed threshold, the points are removed. A support point which has a different disparity value than all its neighbor (adjacent) points is categorized as an outlier and removed as well. As the found support points may not cover the whole image, additional support points in the image corners are added. They adopt the disparity value of their nearest neighbor. Then, image coordinates of the remaining support points are used to create a 2D mesh via Delaunay triangulation. To obtain a dense disparity map, missing disparities are interpolated using the mesh of the Delaunay triangulation by using the nearest-neighbor on the same image line. For more information how the support points are calculated and how the interpolation is done exactly, compare [**? ?**].

3.3 Middlebury MRF library

The Middlebury MRF library [**? ?**] utilizes a global energy function consisting of Markov random fields to formulate an energy minimization problem and offers the following methods to solve this optimization problem:

1. iterated conditional modes (ICM),
2. graph cuts expansion approach (cf. [**? ? ?**]),
3. graph cuts swap approach (cf. [**? ? ?**]),
4. sequential tree-reweighted max-product message passing (TRWS) (cf. [**? ?**]),

5. sequential belief propagation (BPS) (cf. [**?**]),

6. max-product belief propagation (BPM) (cf. [**?**]).

The following subsections give a rough overview on some of those methods. Additionally, a short introduction into MRF-based energy functions is given. Finally, a general outline of the concepts, that are utilized by the above mentioned techniques to solve such optimization problems, is given.

Solving optimization problems

Many problems in computer vision, for instance image smoothing, can be described in terms of energy minimization. Thus, solving of optimization problems is a key part in modern stereo matcher algorithms. They solve the labelling problem as described in Chapter 2. Most of the current disparity algorithms employ global methods to solve an energy minimization problem. Usually, they utilize Markov random fields (MRF) based energy functions. As these are *NP-hard*, approximation algorithms are typically used, like [**? ?**]:

- dynamic programming,
- belief propagation,
- graph cuts.

All of these methods are supposed to solve inference problems, or at least provide approximated solutions. Markov random fields, mentioned before, are also known as Markov network. Bayesian networks as well as Markov networks are so called graphical models. Such graphical models help to understand the reasoning behind those formulations and to actually build algorithms which solve those inference problems. Both networks express the dependencies of nodes as the conditional probability. A chain of nodes is called the joint probability[4]. This then is the product over-all probabilities. The goal of algorithms which solve inference problem is to compute certain marginal probabilities[5], i.e. the probability that some pixel reach a specific label node [**?**]. With inference the computation of these marginal probabilities is meant. Marginal probabilities are defined as the sums over all possible states of all the other nodes in the system. They are also called beliefs [**?**].

[4]The joint probability $P(A \wedge B)$ is the probability of event A and event B occurring. It is the probability of the intersection of two or more events.

[5]The marginal probability is an unconditional probability as it is not conditioned on another event.

Markov random fields

Markov random fields (MRF), also called Markov network, are used to formulate problems in a probabilistic way. The problem thereby is represented as an undirected graph consisting of random variables. For a simple undirected graph compare Figure 3.1.

A MRF is a graph $G = (V, E)$ where $V = 1, 2, ...N$ denotes a set of vertices or nodes. Each node is associated with a random variable u_j for $j = 1...N$. E describes the edges $(i, j) \in E$ between the nodes i and j. The neighborhood of a node i is the set of nodes to which the node i is adjacent, i.e. $j \in N$ if and only if $(i, j) \in E$. The neighborhood of a node i is denoted as N_i.

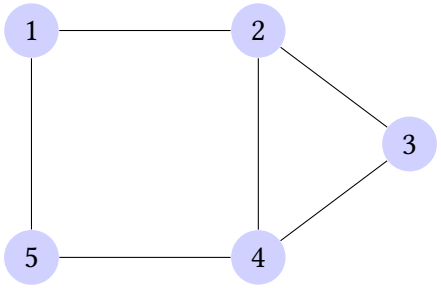


Figure 3.1: Simple undirected unweighted graph

The Markov random field satisfies:

$$P(u_i \mid \{u_j\}_{j \in VN}) = P(u_i \mid \{u_j\}_{j \in N_i}), \tag{3.4}$$

where N_i is the so called Markov blanket of node i. It describes that the graph should be conditionally independent of all of the other variables given its neighbors. A hop from one node to another can be seen as a chain of probabilities which have to occur, also called Markov chain. The main idea behind MRF in combination with computer vision problems is to formulate the labelling problem in such a way, that each pixel has a likelihood to belong to a certain label [**?**]. The core problem is to find exactly one label for each pixel, which is represented as a node in a MRF. This label represents the optimal solution to an underlying problem, in the case of stereo correspondence: the disparity of a pixel regarding a reference pixel [**?**].

Contrary to MRF, also Bayesian networks exist. A Bayesian network is a directed graph whereby MRF is undirected. This implies an important aspect: the direction of a certain probability to hop from one node to another. Whereby MRF can

not represent induced and non-transitive dependencies. Two independent random variables may be connected by an edge because of possible dependencies. Bayesian networks overcome these limitations.

The underlying stereo model of the Middlebury MRF library is based on the research of **?** . They model stereo matching by three coupled MRF [**?**]:

- D as the smooth disparity field,
- L for representing depth-discontinuities,
- O is a spatial binary state for handling occlusions.

Figure 3.2 depicts the relationship between D, L and O. Matched points in I_L and I_R are illustrated with lines, connecting each other. As O denotes pixels which are occluded, b, c, g, h have no counterpart in image I_R. Thus, the two random fields L and D, for obtaining a piecewise smooth surface, are not used. The conditional probability[6] over D, L and O given a pair of stereo images $I = \{I_L, I_R\}$ is defined as:

$$P(D, L, O|I) = \frac{P(I|D, L, O)P(D, L, O)}{P(I)}. \tag{3.5}$$

They then approximate inference via belief propagation over this equation. For a deeper dive into this topic compare [**? ? ? ? ? ? ?**].

Factor graph

As those problems are *NP-hard*, several approximation algorithms exist which are outlined in the following subsections. All of these approximation algorithms work on factor graphs. A factor graph represents a factorized function of several variables. Usually two types of nodes exist in factor graphs, squared and circled ones. Circled ones represent variables of a factor and a squared one represents a factor. Factors define the relationship between variables in the graph as they are obtained by the factorization of the function. Such graphs are bipartite, that means that the nodes of a graph can be divided into two disjoint sets, for instance U and V, such that every edge connects a node U to one in V. These factor graphs help to understand the underlying problem and to imagine the implementation of such algorithms as they are used for breaking down a problem into pieces.

[6] Bayes' theorem: $P(A|B)$, a conditional probability, is the probability of event A occurring, given that event B occurs. $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$ where $P(A)$ and $P(B)$ are the marginal probabilities of event A and B. $P(B|A)$ is the probability of observing event B given that A is true.

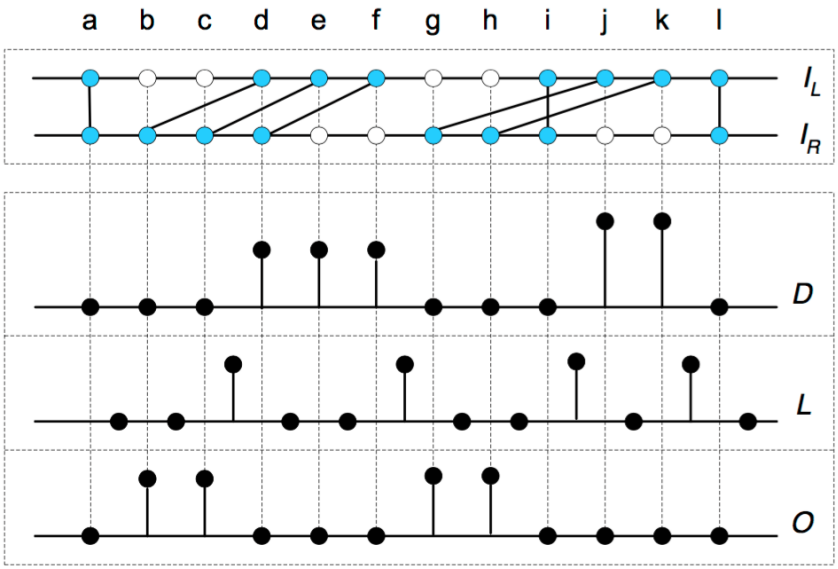


Figure 3.2: Stereo matching model by three coupled MRF's [**?**].

A MRF function is factorized in partial functions and then formulated as a factor graph. The solution to the problem represented by the factor graph is then approximated. An important notion of factor graphs is the message which can be passed from a node to another. So the edges represent communication channels through which messages can be passed. One way to approximate such a factorized function is the use of message passing algorithms (also called belief propagation), which are described later on. One exception exists: if the factor graph contains no cycles, meaning it can be represented as a tree, the solution can be computed exactly.

Dynamic programming

In general, dynamic programming means dividing an optimization problem into smaller chunks. These chunks get solved individually and in the end, they are connected and the optimization problem is minimized [**? ? ?**]. For stereo matching this applies to the partition of a two-dimensional search problem into a series of isolated one-dimensional search problems on each pair of epipolar lines. These problems are then solved independently. With dynamic programming the following energy function (introduced in the foundations Chapter 2) can be solved independently per scanline.

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d) \tag{3.6}$$

Dynamic programming has two benefits, it is solvable efficiently in polynomial time and it enforces the ordering constraint (as it is solved per scanline). But it can lead to streaking effects, meaning that the result image seems to be constructed of many independent layers.

Belief propagation

Belief propagation (BP) in general is a technique to perform inference on a probabilistic model like Bayesian networks or Markov random fields [**? ? ?**]. As mentioned before, **?** presented a stereo model for belief propagation. BP works with messages which are passed from one node to another. This is the reason why BP is also known as the message-passing algorithm. The nodes exchange information about probabilities. In the case of stereo matching, the message contains the probability that the receiver node (a node in MRF) should hold a disparity which is consistent with all information already passed to it by a sender. The nodes are partitioned into low- and high-confidence ones. The messages also carry a property, the entropy. The entropy is high when sending from low- to high-confidence nodes and vice versa (cf. [**? ? ?**]). The nodes calculate a new state after an iteration as they know more about the other node's properties, i.e. marginal probabilities of distant and not directly connected nodes. Also the outcome of past iterations, which yields in joint- and conditional probabilities, influences the overall state of a node. When talking about disparity algorithms, the algorithm ends in a tree if no state is changed anymore and the exact energy can be inferred. In an acyclic graph the algorithm finishes if the overall energy does not improve anymore.

Graph cuts

An additional method to approximate solutions to problems described by Markov random fields, graph cut algorithms can be used [**? ?**]. In general, graph cuts assume a graph G with a set of nodes N and connected by a set of edges E. The goal is to delete enough edges so that each pixel is connected to exactly one label node. Given a weighted graph G with source s and sink t nodes. The graph should be partitioned into two subsets, S and T, where $s \in S$ and $t \in T$. This cut with S and T build a cut-set $C = (S, T)$. Basically, the goal is to find a cut which is minimum, i.e. if the size or weight of this cut is smaller than the size of any other cut. Thus, the cut-set represents a cut such that the sum of edge weights spanning this partition is minimized.

In the case of computer vision, graph cuts are inspired by the combinatorial optimization methods for maximum flow [**? ?**]. Two basic variations of the maximum flow problem exist, called α-β-swap and α-expansion. Initially, three labels exist:

α, β and λ. Normally, one step would be to change the label of a pixel, calculate the energy again and then infer if the change was good or not, depending on the delta. For instance one pixel labelled with λ would then be β. The α-β-swap algorithm interchanges whole areas of α with β whereby areas of λ remain unchanged. In an α-expansion a huge number of pixels labelled β and λ are changed into α. But in each of those methods the outcome is then measured. In Chapter 2 the following equation was introduced:

$$D = \arg\min_{d} E(d). \tag{3.7}$$

If the outcome of such a swap or expansion is better, meaning $E(D_{after}) < E(D_{before})$, the algorithm continues. If not, the algorithm stops. Thus, both algorithms are expected to be stopped after the first unsuccessful run (i.e. energy increases). The difficulty is to find the optimal swap move. As starting point an arbitrary label is chosen. Both is described in [**? ? ? ?**].

3.4 Disparity algorithms applied on videos

Although stereo correspondence is a research field which has been heavily investigated for a few decades, no disparity algorithms that directly target videos yet exist. One reason for that could be the lack of solid ground-truth data as only a few datasets have been introduced lately [**? ?**]. Also, the computational bottleneck of dealing with multi-dimensional data can be an issue, for instance adding a new dimension to the disparity space image which reflects the relationship between multiple frames. As a video is defined by multiple consecutive frames, every disparity algorithm for images can also be applied on videos. The drawback of this trivial approach is the lack of taking the correlation of the frames into account. However, novel approaches were presented by **?** [**?**], **?** [**?**], **?** [**?**], **?** [**?**], **?** [**?**], and are discussed in the following subsections.

3.4.1 Spatiotemporal consistency

The following approaches commonly target the occurrence of noise. On the one hand, noise can occur through estimating disparity. The disparity maps may vary from frame-to-frame which can lead to a flickering effect over time, often perceived as disturbing [**?**]. This happens because each frame is observed separately rather than as a coherent and consecutive signal over time. On the other hand, image or video sensors always produce little noise, although it may not be visible for humans. As a matter of fact, stereo images from real cameras will produce kindly different images due to a variety of reasons, for instance sensor response

differences or luminance [**?** **?**]. Thus, they will not completely match each other which can also yield to noise in disparity maps. Another important factor for the occurrence of noise, which has not been investigated yet, is video compression. This idea is described in more detail in the implementation Chapter 4.

? [**?**] present a two steps approach for dealing with spatiotemporal consistency. First, the disparity maps are computed frame-by-frame. The computed disparity maps are treated as a space-time volume. Then they apply a video restoration algorithm to reduce noise in the space-time volume. This video restoration algorithm is based on the augmented Lagrangian method for total variation (TV) image restoration [**?**]. Basically, it is an algorithm for denoising images by looking at the frames before and after the current frame. Thus, object edges and depth-discontinuity areas are preserved. By applying this algorithm they benefit from three properties of the algorithm: variation regularization, spatial smoothness and temporal consistency, which are established at the same time. This leads to more accurate and spatiotemporal consistent maps. To simulate real scenery they added Gaussian noise to rendered sequences, distributed as $\mathcal{N}(0, 20)$[7]. The outcome produces better results when comparing bad pixels (threshold of 1) and visually clearly better disparity maps as depict in Figure 3.3. As this approach works on computed disparity maps, current image-based disparity algorithms can thereby be easily adapted to the video domain.

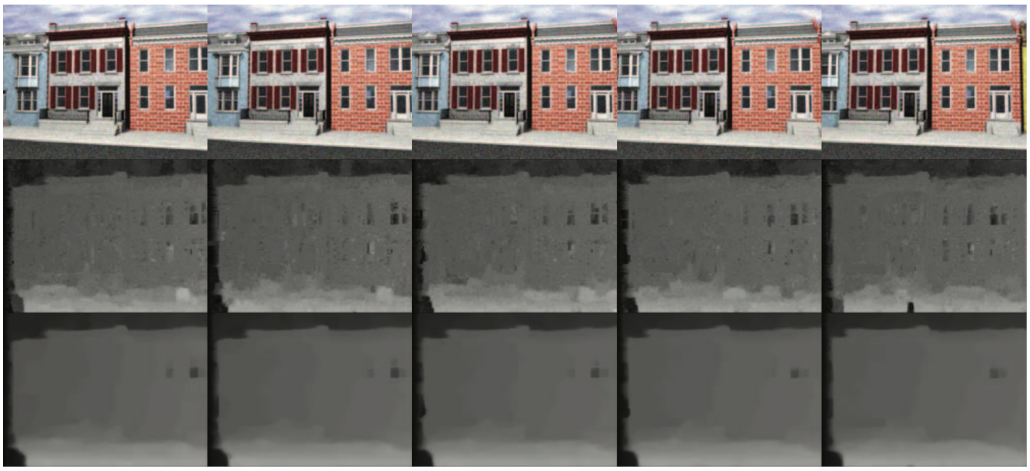


Figure 3.3: Spatiotemporal disparity refinement with the augmented Lagrangian method for TV [**?**]. Top: Original. Middle: Disparity. Bottom: Processed disparity.

[7] Normal (gaussian) distribution is denoted as $\mathcal{N}(\mu, \sigma^2)$, where μ is the mean and σ^2 the variance.

Yet another approach regarding video disparity estimation utilizing the spatiotemporal method above is presented by **?** [**?**]. They focus on salient regions in the video. Motion is an important factor in video processing. Algorithms for estimating salient regions in videos exist. Generally, moving objects lean towards a higher degree of saliency. As typical disparity algorithms tend to have difficulties in estimating the disparity along moving edges and textureless areas, this can help to focus on especially those areas. They utilize motion cues[8] in combination with a modified census transform[9] with a noise buffer to obtain disparity maps. These disparity maps are more accurate and robust towards the edges of moving objects and in textureless areas. Finally, they also apply the previously introduced method to derive a spatiotemporal consistency.

The approaches of **?** [**?**] as well as **?** [**?**] are build on the same principle, which has its origin in the basic approach of **?** [**?**]. They define a matching cost function with an additional property T for the time axis. A space-time cost volume is then generated by stacking the cost maps of input frames. A simple approach could be to smooth the disparity over time by applying a box filter after the disparity map is computed. This would imply that the disparities inside this space-time window are constant. As a result object borders may be blurred up to obliteration and get lost in a non-static scene. They overcome this issue by assuming that the disparity of an object is approximately constant over a small time window and applying weighted box filter. Therefore, they build a 3D filter kernel, which weights the pixels. Pixels which belong to the same object get a high weight and pixels belonging to a different object a lower weight.

Tying in with this approach, **?** [**?**] rewrote the filter as a so called dual-cross-bilateral filter. Instead of using a custom weight model to preserve edges they utilize a bilateral filter which is a common edge-preserving smoothing technique. The cross-bilateral filter preserves those edges while smoothing with respect to a different image. A method for especially stereoscopic images is to use adaptive support weights for correspondence search (cf. [**?**]). This variant smoothes the cost space while preserving edges in both input images. This combined filter is then named the dual-cross-bilateral (DCB) filter. The implementation is called DBC grid. Spatiotemporal consistency is retained with the added temporal dimension T. Observing all frames as a whole is computational complex and difficult. Therefore, they consider five frames as one temporal entity. The DBC grid with this added temporal constraint is called temportal DBC grid. Their approach clearly

[8] Motion cues are responsible for the perception of motion.

[9] Census transform is basically an algorithm, implemented as a filter, for the classification of textures.

reduces errors as illustrated in Figure 3.4. The figure shows a selected frame of a recorded 'skydiving' stereo video [?].

They present a real-time GPU-based implementation for competing with current state-of-the-art disparity algorithms regarding runtime. At this point in time their implementation is the fastest technique in the Middlebury[10] benchmark.

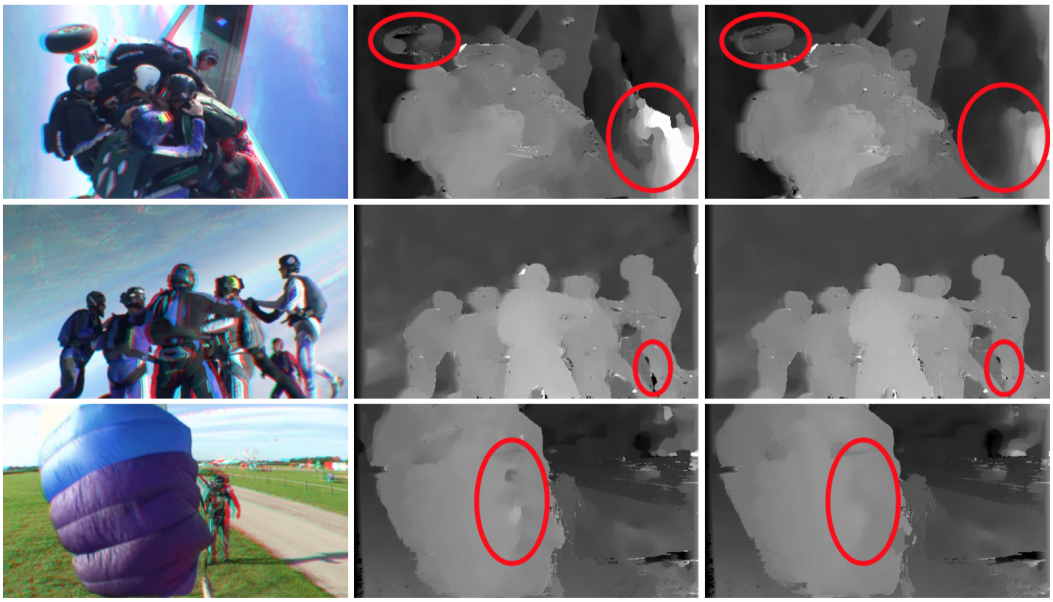


Figure 3.4: Disparity maps of a selected frame of the 'skydiving' stereo video [?]. From left-to-right: video frame (red-cyan anaglyph), DCB Grid, Temporal DBC Grid.

3.4.2 Remapping the disparity range of stereoscopic videos

? examine the problem of remapping the disparity range of stereoscopic images and videos [?]. Remapping of disparity range can be necessary for various reasons. Humans notice the projected stereo content differently, depending on the screen size and the distance to the screen. Another issue is negative disparity. The fundamental underlying problem is the interplay of the human visual perception and restrictions of displays. For instance, displaying a close object on a distant screen may result in a negative disparity and then, humans may experience the viewing as uncomfortable. This can lead to temporary diplopia. Those issues form a real problem in the film industry when producing 3D movies. The

[10] http://vision.middlebury.edu/stereo/eval3/

disparity for the best human experience should be kept in the so called comfort zone, which is the area where eyes feel comfortable. Too high positive disparity can lead to retina rivalry areas, which are muscular intense due to focus issues, whereas negative disparity can even result in painful retina rivalry areas. If 3D content is optimized for a cinema screen it will look differently on a home TV screen or even a tablet device, leading to a distinct viewing experience. This entails the need of changing the disparity after a stereoscopic movie was recorded for the adaption to the current viewing situation of the user. For this purpose, they introduce a set of basic disparity mapping operators for the control and the retargeting of the depth of stereoscopic videos. To actually use those operators stereoscopic warping of video streams is also presented. Basically, those disparity mapping operators define editing operations how the disparity can be modified by formulating a new consistent range of disparity values which respect disparity constraints. The goal is to map the disparity to a new range such that the resulting output view fulfills a stereoscopic, a temporal and a saliency constraint (cf. [**?**]), i.e. provide consistent disparity values according to the new range. These constraints are identical to related work on video retargeting [**?**]. To obtain the new disparity range, a sparse disparity map of feature correspondences is computed. Then, with the use of the disparity operators, a stereoscopic warp of the stereo pair is computed, such that the resulting output views fulfill the desired disparity constraints. Stereoscopic warping use the same basic methods as warping for video retargeting [**?**]. Stereoscopic warping is image warping with the help of the introduced disparity mapping operators. The outcome of the paper are production-oriented rules and guidelines for editing disparity of stereoscopic content. In a survey, user concluded that the applied techniques, i.e. stereoscopic warping with disparity mapping operators, yield in a better viewing experience due to depth structure changes without distracting visual artifacts.

4 Implementation

Chapter 3 points out that no real algorithm for stereoscopic video disparity yet exists. Also, an evaluation suite that assesses disparity algorithms based on stereoscopic videos has not been build yet. Datasets with high-resolution stereo videos are rarely available. Source code for existing disparity algorithms are open-sourced and available for the public domain only in a few cases. Additionally, a lot of different unaligned code for evaluation and comparing disparity is found. Thus, the decision towards an implementation of a novel evaluation suite, built on top of OpenCV, was made. Preexisting source code of disparity algorithms was refactored and integrated. Different masks for fine-grained evaluation were implemented as well. An image diminisher which alters stereo images by adding noise, to simulate real scenery, or artifacts from video compression was created. Additionally, a web front-end was developed to present and visualize the results from any benchmark run within the suite.

The following sections describe the implementation, its components and the subsequent evaluation pipeline for disparity maps. First, the preliminaries are outlined and an overview on the implementation as well as the evaluation engine is given. Second, the integration of existing disparity algorithms is presented. Third, the masks for the evaluation are explained and the implementation is described. Fourth, the image diminisher, created to simulate real use cases is presented. Finally, a simple stereo matcher which respects the spatiotemporal context is described in more detail and the web result viewer of the evaluation engine is introduced.

4.1 Preliminaries

As development platform a MacBookPro was used with the following specifications: i5-4258U CPU @ 2.40GHz (dual-core), 8 GB RAM, a fast SSD. For the later evaluation phase a desktop computer with an i5-2500k @ 3.30GHz (quad-core) was considered. The programming part was done with Atom[1], a modern text editor, and CLion[2] from JetBrains, a cross-platform IDE especially for C++. CMake

[1] `https://atom.io`
[2] `https://www.jetbrains.com/clion/`

as a cross-compiling makefile generator was utilized. Everything except the web result viewer was implemented using C++. To reduce the effort of building an evaluation suite from scratch and for reducing code duplicates OpenCV was used. The final build-chain consists of a set of shell scripts and CMake as makefile generator. With CMake it was possible to cross-compile the app for Linux and use a fast server-instance from DigitalOcean[3] for the generation of the disparity maps and to actually evaluate those.

To not rely on different environments, docker images have been used to serve as basis for containerized virtual environments for the evaluation suite. For this purpose, an image especially for OpenCV[4] was created, open-sourced and used. Docker is a tool for the creation of containers and helps to build, run and ship distributed applications. Those docker containers work in a chroot[5] environment and are isolated from other processes. It is not a complete virtualized machine as it runs on the system's kernel. Finally, the scripting language Python was used to create a set of scripts for combining all components in a chain.

4.2 Overview

Initially, a rapid monolithic prototype was built, featuring the execution of different disparity algorithms for each frame, the creation of masks and the evaluation of a given scene with different quality metrics. But as more datasets were found and various masks as an evaluation method were implemented, the need for a leaner process chain arose. Especially, as disparity algorithms need some time to compute the disparity map for one frame a more split approach was favored. Hence, as videos consist of multiple frames (in our datasets about 100 frames in mean) this is a time consuming task. Sometimes metrics change, the threshold is adjusted or a new metric is found. As a result, the implementation consists of small services, with which computed disparity maps can be computed and evaluated independently. Thus, the monolithic prototype was rewritten and partitioned in smaller microservices shaping three different components:

- disparity algorithm executer,
- mask creator,
- evaluation engine.

[3] `https://www.digitalocean.com`
[4] `https://github.com/benjohnde/dockerbase-opencv`
[5] Chroot stands for "change root" and helps to change the root directory for a current process.

Additionally, a small tool for diminishing images was created. It can simulate real scenery by adding Gaussian noise as recommended by [?]. In addition, FFmpeg [?] was wrapped to simulate artifacts originating from video compression. The output, which each one of those microservices in the chain can generate or operate on, is structured in a simple folder tree.

The Figure 4.1 shows the composition and the evaluation chain of these services. As described in the following subsections, the mask creator needs only the left image, as the disparity map is calculated for the left image as reference. Thus, the masks are only created for the reference image.

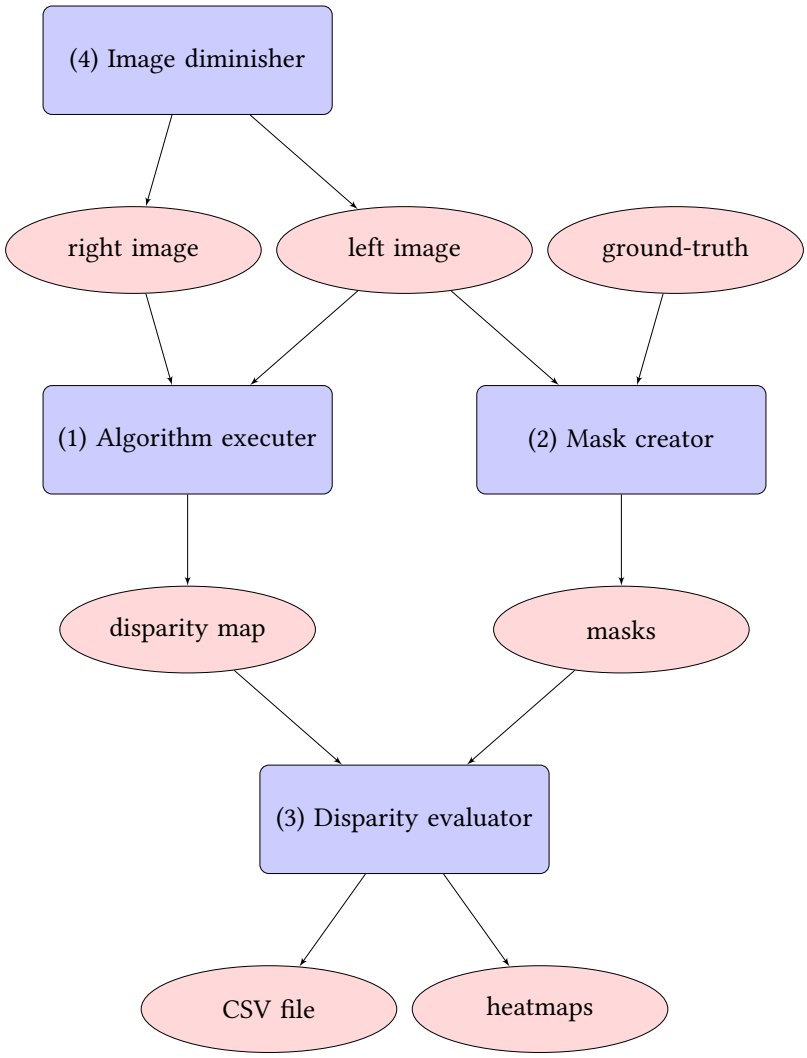


Figure 4.1: Composition and processing pipeline of the implementation.

In contrast to other implementations, input and output are clearly defined and thus different techniques can be adapted easily. Combined frameworks which fulfill two tasks, disparity calculation (as the algorithm is implemented) and the final evaluation step exist. This makes it harder to use the evaluation module separately from the rest. None the less, the open source community around computer vision also lacks of code for a stereo matcher. Due to the diversities of algorithms and evaluation suites, the decision was made to go for an OpenCV implementation of an eval suite for disparity algorithms.

At the current point in time, no disparity algorithms that directly target videos yet exist. As a video is defined by multiple consecutive frames, every disparity algorithm for images can also be applied on videos. The drawback of this trivial approach is the lack of taking the correlation of the frames into account. None the less, it is possible to focus on some other details.

For instance, possible outliers in the sense of frames can occur, which may lead to more erroneous results. The mean performance (error rate) of algorithms on a complete scene can be analyzed. The runtime may vary in a sequence from frame-to-frame. It is also interesting to see the impact of image diminishing effects like compression or noise, simulated as occurring from converting the signal from a real sensor. This is described in greater detail in the upcoming section regarding image diminishing effects.

As middleware between the components OpenEXR[6] is used. OpenEXR is a file format for high dynamic-range (HDR) images. It supports 32-bit floating-point values and is thus good for representing sub-pixel accurate values in a disparity map. The file format is also integrated in OpenCV. For the later evaluation, the comparison of ground-truth data with computed disparity maps, it is sufficient. However, to visualize the images on a default monitor, the values have to be normalized in a range suitable for using sophisticated color ranges like RGB. Hence, heatmaps are created with normalized disparity maps in the range of $0 - 255$.

Basically, the evaluation engine takes a computed disparity map and the ground-truth counterpart as inputs. Only comparing both provides low informative value. Algorithms tend to produce disparity maps with a few unknown fields (e.g. noise or occluded pixels). Crucial are also depth-discontinuities along object borders, textureless regions, and occluded pixels. Thus, masks are used to only focus on these particular areas in an evaluation. The creation of these masks are illustrated accurately in the upcoming section. In Chapter 5, which discusses evaluation

[6] `http://www.openexr.com`

and results, the masks are used in combination with defined quality metrics. The evaluation engine applies these quality metrics in combination with masks and outputs the result in a simple CSV file[7]. As seen in Figure 4.1, the implementation consists of three components. After an algorithm is executed, the masks are created. Finally, the computed disparity maps are compared with their ground-truth companion with each mask applied.

Python scripts in combination with the evaluation engine represent the whole eval-chain. They work basically as a wrapper to iteratively execute each component over each frame of a given input sequence and aggregate the results. The evaluation engine, as well as the other components, are run with command line arguments to pass specific parameters for the execution, for instance which input images should be used and where the computed disparity map should be saved.

4.3 Integration of existing algorithms

As shown in Chapter 3, deciding which algorithms should be implemented was not easy. There is a huge diversity of used technologies amongst disparity algorithms: various programming languages, CPU versus a CUDA[8] implementation, different coding styles and libraries. As a matter of fact, this makes it hard to implement and evaluate every available disparity algorithm. Thus, a more streamlined extendable architecture is presented. The algorithms, which were introduced in the related work Chapter 3, are adopted and integrated. The architecture of the disparity interface is depicted in Figure 4.2.

The abstract class `DisparityAlgorithm` provides both images, left and right one, in form of a `cv::Mat` object and also holds the resulting disparity map. On top of this abstract class further algorithms can be integrated easily. The output of each algorithm is then saved in an OpenEXR file at a given destination path. The following subsections describe how the algorithms were integrated, the pre- and post-processing steps if necessary, and illustrate the parameter.

OpenCV

The OpenCV library [**?**], currently at version 3.1.0, offers two implementations for disparity estimation, block matching and semi-global block matching based on the

[7] CSV stands for comma-separated values. A CSV file usually represents a table, with rows being the lines and in columns the values are separated with commas.

[8] CUDA is a high level programming language that targets highly parallel calculations utilizing Graphic Processing Units (GPUs) by Nvidia.

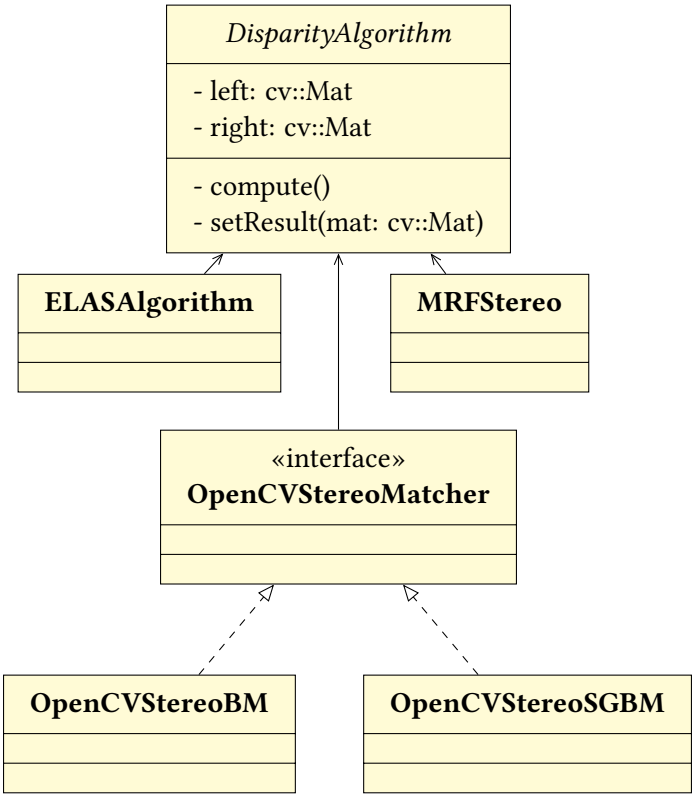


Figure 4.2: Architectural overview on the disparity interface in form of an UML diagram.

idea of **?** [**?**]. Both implementations are integrated and fed with the same input parameters. The matching cost are calculated differently in both implementations. Another difference is, that the simple block matching implementation only operates on grayscale. As block size, an odd number needs to be used as adjacent pixels are observed. Here, 9 was chosen as block size. A pre-filter was not used. The speckle window, which is used to smooth disparities over noisy regions, is also not used. The `disp12MaxDiff` parameter was set to 1, which represents the maximum allowed difference in the left-to-right disparity consistency check. If the difference is higher, the value is set as unknown. The min- and max disparities are set to a value, which should be fit for most of the investigated sequences, -48 and 128.

After an algorithm run, the disparity WLS filter is applied with $\lambda = 8000.0$ and $\sigma_{color} = 1.5$. λ defines the amount of regularization during the filtering process. Larger values lead to over-smoothed edges in the disparity map, so that the disparity map adheres more to the source image edges. σ_{color} specifies how sensitive

the filtering is done towards the source image edges. Both are typical values for smoothing the disparity according to the OpenCV documentation. As a post-processing step, the resulting `cv::Mat` has to be divided through 16, as every input value gets multiplied with 16 according to the source-code[9]. This step is integrated for sub-pixel-accurate disparity maps. In the end, each point contains of 4 fractional bits.

Middlebury MRF library

The Middlebury MRF library [**?**] consist of three parts, which all have to be composed together:

- The imageLib, a small library for 2D multi-band images.
- MRF, an energy minimization software [**?**].
- mrfstereo, which is a stereo matcher front-end for the MRF library.

Some patches need to be applied to the MRF energy minimization software in order to get the graph cuts algorithms to work. The composition is open-sourced[10] with a Makefile. Finally, a small wrapper for the execution was created. The wrapper prepares the input images beforehand, saves them in a temporarily directory and post-processes the output. The output is a disparity map, represented as 8-bit fixed points, where each disparity value has 4 fractional bits. Thus, the result output is converted to a 32-bit floating values matrix, divided through 4 and saved as OpenEXR file.

The MRF library offers the possibility to use the Birchfield-Tomasi method for matching cost calculation instead of the more commonly used sum of absolute differences (SAD). **?** defined matching cost as a global function which penalties occlusions and rewards matches [**?**]. According to [**? ?**] the Birchfield-Tomasi method leads to better results than traditional similarities measurements like SAD. In case of a traditional local similarities measurement, for instance SAD, it is possible to truncate the differences to a certain maximum value. In Chapter 3, the different algorithms, which are implemented in the Middlebury library, were presented. The used algorithm can be selected via command-line parameter. Other parameters are only for the smoothness term of the energy function. It is possible to adjust the smoothness exponent, the maximum value of the smoothness term, and the weight λ of the smoothness term.

[9] `https://github.com/Itseez/opencv/blob/3942b1f36261b196a264e\b35c996222848fe3c93/modules/calib3d/src/stereobm.cpp#L564`

[10] `https://github.com/benjohnde/disparity-algorithms`

ELAS: Efficient large-scaler stereo matching

ELAS summarizes a front-end interface in combination with the library for efficient large-scale stereo matching (LIBELAS) from **?** [**? ?**]. It offers a C++ library with a wrapper for MATLAB applications to compute disparity maps from rectified graylevel stereo pairs. ELAS is integrated with a simple wrapper which executes the ELAS binary. ELAS required the input files as PGM[11] and outputs PFM[12] images. In a preprocessing step the input stereo images are temporarily saved in the PGM format as grayscale images. After the execution of ELAS, as a post-processing step, the PFM files are converted to OpenEXR files, as the subsequent processes expect OpenEXR files as input. For processing the PFM files, a simple file reader was created, according to the official specifications[13]. Specific characteristics of this file format are, that the rows are written bottom-to-top and each column has to be multiplied with a scale factor. The ELAS binary can be started with no parameters but the maximum disparity value.

4.4 Fine-grained evaluation via masks

The trivial possibility for evaluation would be to just compare the computed disparity map with its ground-truth companion. But, this approach would produce erroneous results due to a variety of reasons. To give an example, occluded regions would lead to a higher error rate. As a remedy, masks are introduced to simply focus on interesting pixels. Masks are normal matrices of the size of the input image and they reflect two states: 0 and 1, whereas 1 stands for masked. In the literature they are sometimes also named bitmasks. In this section the following masks are introduced:

- depth-discontinuity,
- textured regions,
- occluded pixels,
- salient regions.

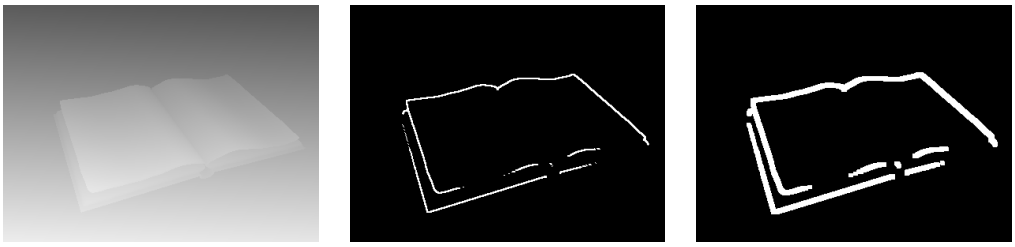
Programmatically they are represented through the OpenCV matrix class `cv::Mat` which can not work as a binary matrix and obtain just two states. Thus, a `cv::Mat` with `CV_8U1` is initialized, which means one color channel utilizing 8-bit unsigned for the values. In this matrix, 1 is mapped to 255.

[11] PGM is a grayscale image format, part of the Netpbm library.
[12] Similar to OpenEXR, PFM is a floating-point image file format, also part of the Netpbm library.
[13] `http://netpbm.sourceforge.net/doc/pfm.html`

Depth-discontinuity

Determining correspondence can fail in textureless or depth-discontinuous regions as mentioned beforehand in Chapter 2. Thus, it is interesting to see how disparity algorithms handle such regions. So, a depth-discontinuity mask was implemented. Depth-discontinuity is observed in the ground-truth disparity map. It is defined as regions, where adjacent disparities differ by more than a certain gap [**? ?**]. For this purpose two basic morphological[14] image processing operations from the OpenCV library are used: `dilate` and `erode`. `dilate` outputs for each position in a `cv::Mat` the maximum value of all the pixels in the neighborhood. `erode` does nearly the same, but returns the minimum value. Both are fed with a kernel, specifying which adjacent pixels should be taken into account.

For getting the disparity gap, the image gets dilated to get the maximum disparity of a pixel's neighbor. Then eroded to estimate the minimum disparity of a pixel's neighbor. Both is done with a $3x3$ kernel, and all adjacent pixels are masked to observe. Finally, the depth-discontinuity areas are the ones with $maximum - truth > gap$. The depth-discontinuity is then dilated by a window with given width to intense the area. Visually, the best results are achieved with a gap of $2.0f$ and a width of 3 for the final dilation. Figure 4.3 depicts this.



(a) ground-truth disparity (b) mask before final dilation (c) final mask for evaluation

Figure 4.3: A depth-discontinuity mask, exampled with the first frame of the book scene from the Middlebury dataset.

Textured regions

Stereo matching algorithms act on the assumption, that disparity is smooth, especially if contrast and color intensity do not change drastically. Therefore, it can be interesting to see how those algorithms treat textured and textureless regions. Basically, regions with little or no texture in an image are defined as areas, where

[14] Morphological operations are techniques to analyze and process geometrical structures.

the squared horizontal intensity gradient, averaged over a square window of a given size, is below a given threshold [**?** **?**].

This is implemented with the OpenCV `Sobel` operator and the `boxfilter`. The sobel operator is a simple edge-detection filter which emphasizes edges in an image by representing the gradient magnitudes from the derivatives calculation. The boxfilter is a blurring filter, which smoothes an image using a normalized kernel with a given size. This is one of the simplest smoothing filters, as it just calculates the mean of its kernel neighbors, with all weighted equally. So with the sobel filter, the edges are highlighted so that a computer can recognize them whereas the boxfilter smoothes the borders of recognized edges a bit. Finally, the matrix of this outcome is checked against a threshold. If a value of the matrix is greater than the threshold, the pixel is marked as a textured pixel.

The values for both, kernel size and threshold are depending of the scenery. With the datasets which are evaluated in the upcoming Chapter 5, visually, the best results are achieved with a kernel size of $(2\ x\ 2)$ for the boxfilter, and a threshold of 16. Figure 4.4 depicts this approach.

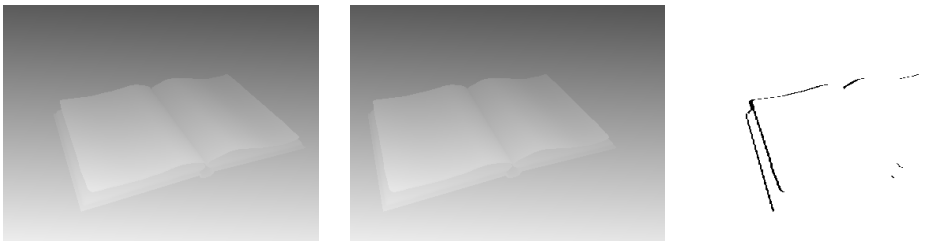
(a) image of left camera (b) final textured mask

Figure 4.4: Textured regions recognition with the first frame of the street scene from the Middlebury dataset.

Occluded pixels

An occluded pixel is defined as a pixel, which is hidden in one of the two images, for instance an object hides it from a different angle. In the case of stereo matching the disparity can not be calculated for such pixels. Thus, occluded pixels have to be handled properly, as they could distort our result. For this purpose, a simple

mask for non-occluded areas is introduced, to indicate which pixels on the scene are visible for both cameras and which are not. Figure 4.5 shows an example of such a non-occluded areas mask.



(a) left ground-truth (b) right ground-truth (c) resulting mask

Figure 4.5: The ground-truth disparity maps of the first frame from the book scene is used to illustrate the non-occluded areas mask.

To obtain the mask, both disparity maps, left and right one, are iterated. Pixels that are visible from both cameras should have the same disparity value in both disparity maps. For occluded pixels, the value is different [**? ? ?**]. Hence, a simple cross-check is sufficient. It is implemented as the absolute value from the subtraction of the left value at a given position $d_L(x, y)$, and the right value at the corresponding position $d_R(x - d_L(x, y), y)$. The result is then checked against a threshold. If the value is greater than this threshold, the pixel is marked as occluded. A suitable value for the threshold is 1, according to [**?**].

Saliency detection

As a novel approach, saliency detection is used as another criteria for the later evaluation. There exist some algorithms for saliency detection in images and videos [**? ? ?**]. OpenCV offers two different saliency detection algorithms:

- one for images, `StaticSaliency`, and
- one for videos `MotionSaliency`.

Here, the static saliency approach was used [**?**] and implemented. The algorithm outputs a saliency map with different saliency values, so a pixel can be more or less salient in the overall context. This approach is also called the spectral residual approach as it analyzes the spectrum of an input image and creates a spectral residual model. However, in the end a binary saliency map is outputted from the spectral residual model, where the pixels are masked as salient if their intensity in this spectral residual model is above a certain threshold. This threshold can

not be set as parameter, but is denoted as three times the average intensity of the saliency map. Basically, the main idea is to extract the pixels which jump out of smooth curves, as they deserve most certainly the humans attention (cf. [**?**]). No parameters can be adjusted. Figure 4.6 shows an example binary saliency map of an image from the SVDDD dataset, which will be presented in the upcoming Chapter 5.

(a) left image (b) salient mask

Figure 4.6: Frame of the tank sequence from the Cambridge dataset and the corresponding salient regions mask from the static saliency approach.

4.5 Image diminisher to simulate real use cases

The presented masks can help to focus on different aspects during the evaluation. Another interesting part of the evaluation is, to see the impact of small errors on the outcome of disparity algorithms. For investigating the robustness of disparity algorithms against image diminishing effects like artifacts from video compression or noise like from a real sensor. As both can be simulated by altering synthetic video sequences, an image diminisher to simulate real use cases is implemented. Basically, this image diminisher consists of two parts: an OpenCV C++ program, which can add noise on top of images, and python script which wraps commands of FFmpeg to alter video sequences.

Gaussian noise

As seen in the related work in Chapter 3, some approaches use restoration algorithms in order to reduce occurring noise. Hence, noise generation was added as a preprocessing step in order to see how noise disrupts disparity algorithms in general. Gaussian noise is used, meaning that the noise is Gaussian distributed

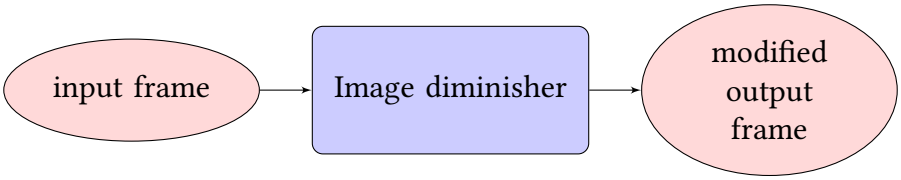


Figure 4.7: Flow of the image diminisher.

(or normal distributed). Normal (Gaussian) distribution is denoted as $\mathcal{N}(\mu, \sigma^2)$, where μ is the mean and σ^2 the variance (squared standard deviation).

As recommended by [**?**], gaussian noise is added to synthetic video sequences with a mean of 0 to simulate real scenery. The variance, σ^2 can be set in our evaluation suite in order to see how this distracts the image. **?** recommended a variance of 20. The distribution is illustrated in Figure 4.8. As can be seen, in comparison to 20, a value of 5 yields to modified pixels by adding values of $[-15; 15]$. Beforehand, to avoid errors, the matrix is converted from 8-bit to 16-bit in order to avoid overflow. Finally, the matrix get reconverted to 8-bit. Values > 255 and < 0 are fixed to 255 and respectively 0.

It is added with the `randn`[15] function of the OpenCV library [**?**]. It generates a `cv::Mat` with the random distribution of values. This matrix is then added on the source matrix, resulting in a noisy image. For each frame, left and right, a new noise matrix is created. Salt and pepper noise was not chosen, as it does not reflect real scenery. Salt and pepper noise is not a common noise, as it contains random occurrences of black and white pixels. It mostly appears by defect image sensors or erroneous analog-to-digital conversion.

Video compression

Most of the widely used video compression techniques belong to the lossy data compression algorithms. As in contrast to lossless compression, lossy tends to produce small artifacts in images or videos as it does not reconstruct the original data as a whole. FFmpeg [**?**] is used to simulate the degradation of stereo videos. FFmpeg is a library for video en- and decoding to create video sequences from images with popular video codecs and also to divide a video sequences into images. Figure 4.9 depicts the integration of FFmpeg in the image diminish process and

[15] `http://docs.opencv.org/master/d2/de8/group__core__array.html#gaeff1f61e972d133a04ce3a5f81cf6808`

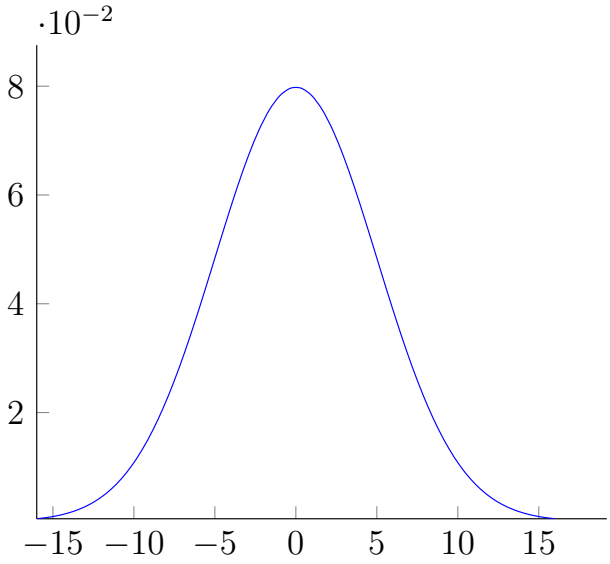


Figure 4.8: Gaussian normal distribution with $\mathcal{N}(0, 5)$.

how the python script invokes FFmpeg. H.265[16] was utilized as codec for video compression. With H.265 only one parameter exists. Here, FFmpeg is used with the libx265[17], an open source HEVC encoder. The only parameter, which can be set is the constant rate factor (CRF). The CRF is a new technique to compress videos. Normally, a quantizer scale factor (QP) exists in video compression. This is a factor, which defines how strong the video should be compressed. In traditional video compression techniques, this factor does not vary throughout a video and thus all frames receive the same compression. H.265 as well as H.264 vary the QP according to the CRF depending on the scenery. Fast moving scenes will receive a stronger QP than slow moving scenes, as the human eye will not focus on too many details in scenes with fast motion. The suitable values vary from $0-51$, with 0 implicating lossless compression and 51 meaning the best possible compression with the worst visual outcome. The default value is 28 for H.265[18].

[16] H.265 is also known as High Efficiency Video Coding (HEVC) and is a standard for encoding/decoding high-resolution video content.
[17] `https://bitbucket.org/multicoreware/x265/overview`
[18] `https://trac.ffmpeg.org/wiki/Encode/H.265`

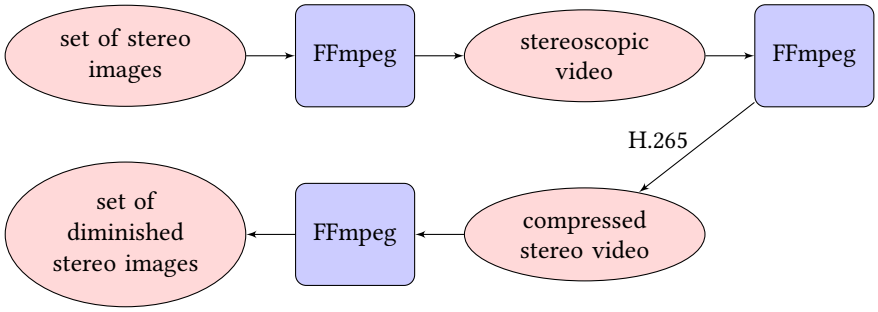


Figure 4.9: Flow of FFmpeg as image diminisher.

4.6 Simple stereo matcher

A simple naive stereo matcher (SNSM)[19] has been implemented for several reasons. First, the SNSM was built to see the limitations of such fast approaches without focusing for instance on arbitrary regions in an image, which can lead to erroneous results. Second, to enhance the SNSM by adding the ability of taking the time axis into account, named spatiotemporal SNSM. Third, to have a skeleton for further research and development. It would be a tremendous task to implement a feature-complete stereo matcher, for instance implementing all available matching cost functions. Hence, only the sum of absolute differences (SAD) was implemented as matching cost function. Moreover, it is a simple area-matching approach with a given block (window) size. Additionally, the previously introduced disparity constraints are not handled properly. The algorithm is implemented with OpenCV as it provides classes and methods to work with multi-dimensional data.

The basic idea is to create a four-dimensional matrix which stores the matching cost for each possible disparity of each frame. Then, it is possible to take the time axis into account by not using the matching cost of one frame but of multiple consecutive frames. Algorithm 1 calculates the matching cost for each possible disparity and stores them in a four-dimensional matrix. As input parameter, the algorithm expects a set of rectified stereo image pairs, $I_L = \{I_{L_1} ... I_{L_n}\}$ and $I_R = \{I_{R_1} ... I_{R_n}\}$, whereas n denotes the amount of images. The maximum disparity to be found is denoted as d_{max} whereas $wSize$ denotes the window size for the blocks to be matched. RECT$\{x, y, width, height\}$ denotes a rectangle cropped out of a given image.

[19] `https://github.com/benjohnde/disparity-evaluation/tree/master/4_NaiveImplementation`

Algorithm 1: CreateDisparitySpaceImage

Input: I_L, I_R, d_{max}, $wSize$
Output: C

```
1  step ← (wSize − 1)/2
2  C ← CreateMatrix(Cols(I_L), Rows(I_L), d_max)
3  for t ← 0 to Images(I_L) do
4      leftImage ← I_L(t)
5      rightImage ← I_L(t)
6      for y ← 0 + step to Rows(I_L(0)) − step do
7          for x ← 0 + step to Cols(I_L(0)) − step − d_max do
8              for d ← 0 to d_max do
9                  rect_L ← Rect{x − step, y − step, wSize, wSize}
10                 rect_R ← Rect{x + d − step, y − step, wSize, wSize}
11                 window_L ← leftImage(rect_L)
12                 window_R ← rightImage(rect_R)
13                 C(x, y, t, d) ← MatchingCost(window_L, window_R)
14 return C
```

CreateMatrix creates a matrix with dimensions according to the input parameters. The size of each dimension is defined with the parameter itself. The output is the matrix $C = C(x, y, t, d)$ whereby (x, y) is the position in the reference image I_{L_t}, t the number of the frame and d the disparity. After the invocation, C returns the matching cost for the given input. Furthermore, the matching cost can be queried with the function $C(x, y, d) \iff C(x, y, 0, d)$, which also represents the disparity space image, as introduced in the foundations Chapter 2, by ignoring the time-dimension.

Algorithm 2: MatchingCost

Input: $window_L$, $window_R$
Output: $cost$

```
1  Δ ← |window_L − window_R|
2  cost ← MatrixSum(Δ)
3  return cost
```

Algorithm 3, then computes the final disparity map by taking the disparity with the minimum matching cost for each field in the reference matrix. Disparities(C) outputs d_{max}, the length of the array holding the disparities, in this matrix. Algorithm 3 needs as parameter the disparity space image, here denoted as $C(x, y, t, d)$

and the number of the frame f, for which the disparity map should be computed. The matching cost is, as mentioned beforehand, the sum of absolute differences in a given window, as can be seen in Algorithm 2. The Δ is estimated as the absolute difference between two windows. Then, the cost is the sum of all fields in this matrix, which is computed by MatrixSum.

Algorithm 3: GetDisparityMap

```
Input: C, t, weighted
Output: DisparityMap
1  DisparityMap ← CreateMatrix(Cols(C), Rows(C))
2  frames ← Frames(C)
3  spatiotemporal ← frames < 3 ? True : False
4  for t ← 0 to frames do
5      for y ← 0 to Rows(C) do
6          for x ← 0 to Cols(C) do
7              if spatiotemporal == True then
8                  if t == 0 then
9                      f_0 ← 0
10                     f_1 ← 1
11                     f_2 ← 2
12                 else if t == frames − 1 then
13                     f_0 ← frames − 3
14                     f_1 ← frames − 2
15                     f_2 ← frames − 1
16                 else
17                     f_0 ← t − 1
18                     f_1 ← t
19                     f_2 ← t + 1
20                 if weighted == True then
21                     Cost ← 1/4 C(x, y, f_0) + 2/4 C(x, y, f_1) + 1/4 C(x, y, f_2)
22                 else
23                     Cost ← C(x, y, f_0) + C(x, y, f_1) + C(x, y, f_2)
24                 DisparityMap(x, y) ← BestMatch(Cost)
25             else
26                 DisparityMap(x, y) ← BestMatch(C(x, y, f))
27 return DisparityMap
```

Algorithm 3 uses the Algorithm 4 which outputs the best match given an amount of disparities $d \in D = \{0 \ldots d_{max}\}$. It iterates over all disparities and finds the minimum. The best match is in this case the minimum matching cost.

Algorithm 4: BESTMATCH

Input: C,
Output: $bestMatch$

```
1 cost ← ∞
2 bestMatch ← 0
3 for d ← 0 to DISPARITIES(C) do
4     if cost > C(d) then
5         cost ← C(d)
6         bestMatch ← d
7 return bestMatch
```

The spatiotemporal approach utilizes a new dimension, the time axis t, and adds it to the disparity space image. $C(x, y, d, t)$ is constructed in Algorithm 1 whereby $t \in T = \{0 \ldots n\}$ denotes the current processed frame. Then, the DSI knows for each position, given a disparity and a frame the matching cost.

As a simple idea, the disparity, chosen for a frame, is the minimum over three frames. Looking at a frame at time f_t, the frames f_{t-1} and f_{t+1} are also taken into account. For the first and the last frame, the two following or respectively the two frames beforehand are used. The underlying assumption is, that there is no much motion between the frames, because this would lead to inaccurate matching cost due to possible object hops.

If the amount of frames is smaller than three ($T < 3$), the spatiotemporal approach is not used. Another possibility is to weight the frames. Here, normal distribution was assumed. Thus, the frame before and after the current frame are taken by $\frac{1}{4}$ into account, whereas the current frame is taken by $\frac{2}{4}$ into account. As a result, a new matching cost matrix is created which reflects the disparity using the weighted approach. The weighted approach can be toggled via the parameter $weighted$ in Algorithm 3, which can be TRUE or FALSE.

For better representation, all approaches are integrated in Algorithm 3. Both, the unweighted and the weighted spatiotemporal approach are evaluated with the other introduced disparity algorithms. The simple stereo matcher without the spatiotemporal dimension is evaluated as well. The prediction is, that all three approaches get outperformed by the other stereo matcher. The simple reason for

that is the lack of handling the disparity constraints as described in the foundations Chapter 2. None the less, it is exciting to see, if the naive spatiotemporal approach achieves better results than without.

4.7 Web result viewer of the evaluation suite

As during the evaluation a lot of masks as well as heatmaps are created and saved, it was helpful to see the visual output of both, especially as a sequence of frames. Thus, a web result viewer for the evaluation suite was implemented. It was created with Node.js[20] and Express[21].

The following Figures 4.10, 4.11 and 4.12 illustrate the implemented web result viewer. The viewer displays all created masks and heatmaps, as well as the input images and their ground-truth disparity map. In addition, an algorithm can be selected and the computed disparity map is shown. The result is then visualized and the possibility to jump between frames as well as playing them automatically with different speed options is given.

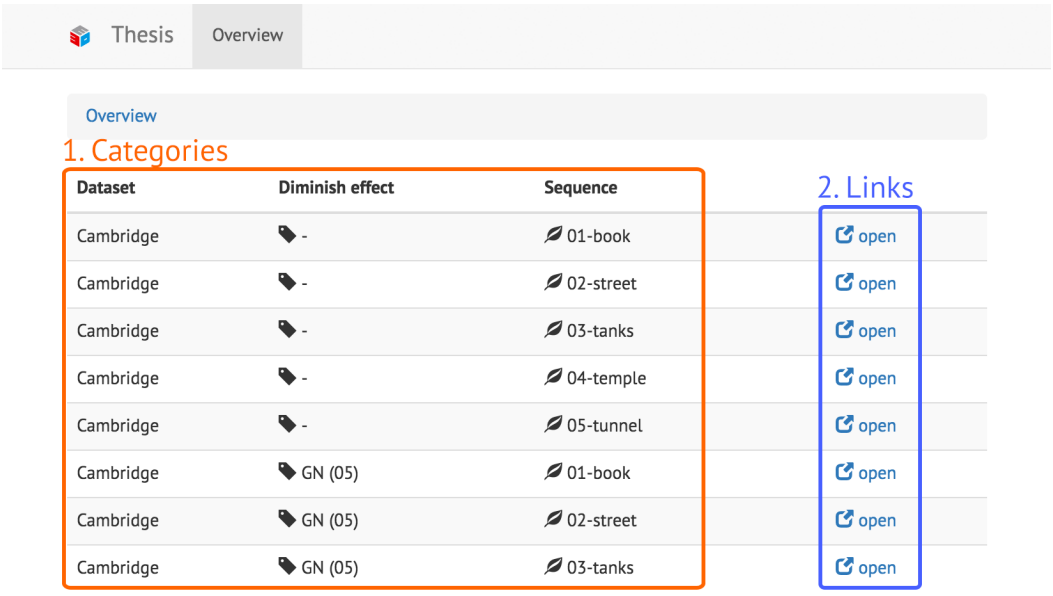


Figure 4.10: Screenshot of overview page of the web result viewer.

[20] Node.js is a framework for developing server-side web applications with JavaScript, cf. `https://nodejs.org/en/`.

[21] Simple model-view-controller (MVC) web-framework, based on Node.js, cf. `http://expressjs.com`.

In Figure 4.10 is the focus on displaying the categories of the results. As various diminishing effects are applied during the evaluation and a dataset can contain different sequences, the results are initially grouped for a deeper dive. With a click on one of the links, the next overview page is displayed. Figure 4.11 displays the mean results of a complete sequence. The algorithm is named as well. With a click on one of the links, the final detailed result view is displayed.

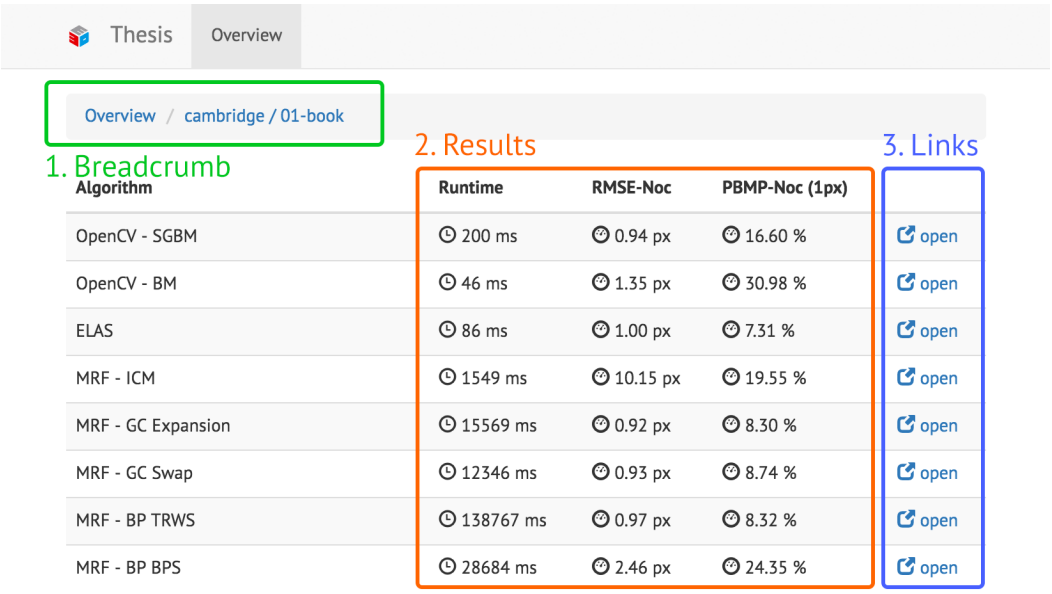


Figure 4.11: Screenshot of sequences with results in the web result viewer.

The final result page, as illustrated in Figure 4.12, shows all created masks, left and right image, the ground-truth disparity and the computed disparity as well as the outliers heatmap. For an easier access to the other pages, a breadcrumb has been installed with which the previous pages can be reached. As mentioned beforehand, the web viewer is capable of playing the whole sequence in the browser. The control bar enables the user to determine the playback and the speed of the playback. A progress bar feedbacks which frame is currently displayed. Additionally, the final result page contains some statistical results of the evaluation, like the current viewed frame, runtime and general performance.

From a technical perspective, the web result viewer maps the filesystem to a web view. The starting point can be configured in a config file. The results are read out of the CSV files, converted to JSON[22] objects and displayed via small scripts.

[22] JSON stands for the JavaScript Object Notation and represents a more human-readable collection of attribute-value pairs.

With jQuery[23], the displayed objects are changed without the need for reloading the whole page. Thus, a sequence can be fluently played and all, the images, the disparity maps, the masks and the result line change at once.

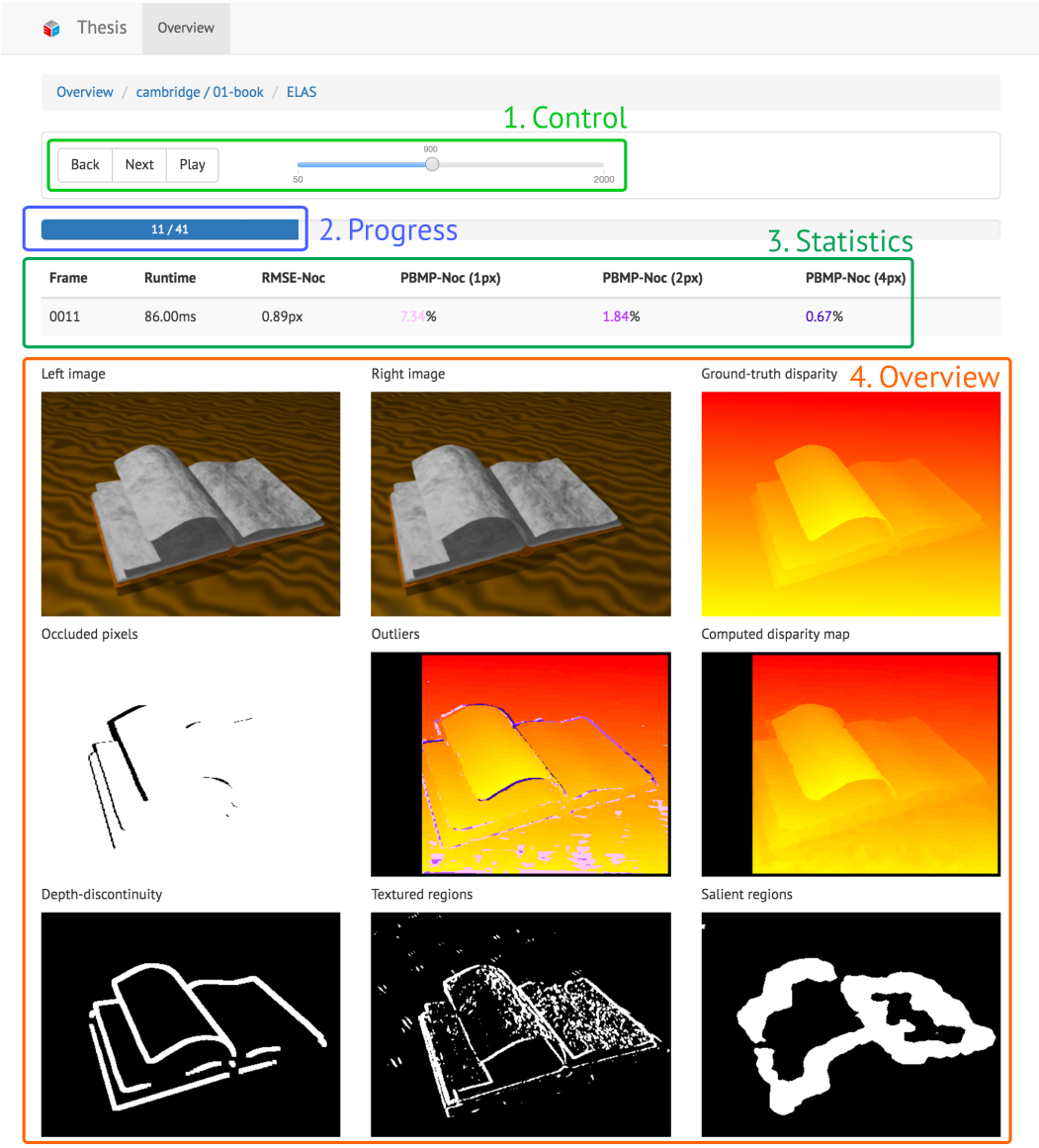


Figure 4.12: Detail of one result in the web result viewer.

[23] jQuery is a small JavaScript library for client-side. It helps to select, manipulate or travers the DOM-tree of a web page, cf. `https://jquery.com`.

5 Evaluation and results

In the previous chapter, the evaluation suite and its features were presented. This chapter is split into several sections. At first, the datasets used for evaluation are introduced. Afterwards, quality metrics are defined to compare disparity algorithms applied to stereoscopic videos quantitatively. Third, the procedure of measuring the introduced metrics is described in greater detail. Additionally, the visualization of measured results is explained. Fourth, the results are presented and discussed.

Overall, the generation of all disparity maps and the process of evaluation took more than 28 days, resulting in about 50 GB of stereoscopic videos. Accumulated, the computed disparity maps, the created masks as well as the heatmaps, allocated about 18 GB.

5.1 Datasets

A dataset basically describes a set of stereo images, which additionally contains ground-truth disparity maps. The aim of such datasets is to provide accurate data, researchers can rely on. This datasets can be used to evaluate the performance of computer vision algorithms, for example the disparity algorithms introduced by this thesis. Without having such datasets it would be crucial to rate the overall quality of stereo matching algorithms. As for today, no high-resolution stereoscopic video dataset yet exists, neither a synthetic nor a captured one. In order to obtain ground-truth depth information, two general options are available. On the one hand, the real-world can be sampled via area scanner, for instance a radar sensor (cf. KITTI vision benchmark suite[1]). On the other hand, a synthetic computer-animated scene is created with rendering the scene using two virtual cameras to generate disparity maps. Of course, the former approach is more error-prone than the latter one. The former one can lack of accuracy due to false measurements whereas the latter one provides real ground-truth information.

? came up with an interesting approach: as area scanners are never 100 percent

[1] `http://www.cvlibs.net/datasets/kitti/`

accurate, they introduced error-bars, which can range from $0 - 10$. An error-bar indicates the certainty of an area scanner, that the measured disparity at a given pixel is valid [**?**]. As it would be a tremendous task to evaluate every existing stereoscopic dataset with every existing disparity algorithms, three different datasets to focus on in the context of this thesis were chosen. Other datasets, which were found but not investigated are the MPI Sintel Stereo Training Data, created for optical flow evaluation [**?**] and the Middlebury stereo dataset, which provides real images with ground-truth information [**?**].

Tsukuba stereo dataset

As reference dataset, the reworked Tsukuba Stereo Dataset was chosen [**?**]. One of the three scenes is called tsukuba to honour the popular *Head and Lamp* scene and was shortly introduced in the foundations in chapter 2. The reference dataset is used to see if the implementation leads to similar results as in other stereo benchmarks. This does not verify that the presented implementation is error-free but can point in the right direction. Of course, the settings (i.e. parameters of an algorithm) depends on the input material (e.g. size of the images, noise occurrences) and on the type of scenery, for instance many regions with arbitrary surfaces (textured vs textureless). Hence, it is possible to have good parameters for one scene and not for another. However, in order to evaluate those algorithms, the reference dataset is used to see how the evaluation engine actually works with the same parameters on the same images.

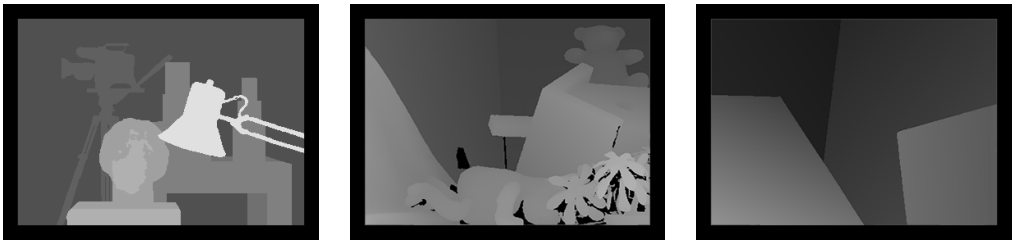
Cambridge stereo dataset

The second dataset from the University of Cambridge[2] is created especially for the evaluation of the DCBGrid. This is also one of the first stereoscopic datasets targeting videos.

The Cambridge stereo dataset consists of five different rendered scenes at a resolution of 400 x 300 pixels, each consisting of about 100 frames [**?**]. The dataset scales disparity in the range from 0 to 64 pixels. The disparity maps are delivered as PNG image files, at 8 bit color depth $(0 - 255)$. Thus, the disparity maps are loaded into a matrix and simply converted to range from $0 - 64$ by dividing the whole matrix through 4. Beforehand, the matrix is converted into CV_32F, meaning that each element is represented by 32 bit floating values.

[2] `http://www.cl.cam.ac.uk/research/rainbow/projects/dcbgrid/datasets/`

(a) tsukuba scene (b) teddy scene (c) venus scene



(d) tsukuba disparity (e) teddy disparity (f) venus disparity

Figure 5.1: Tsukuba stereo dataset [**?**]

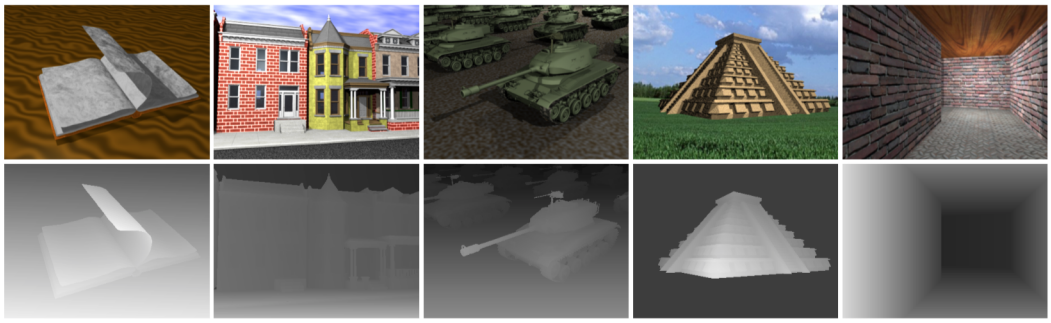


Figure 5.2: Cambridge stereo dataset example

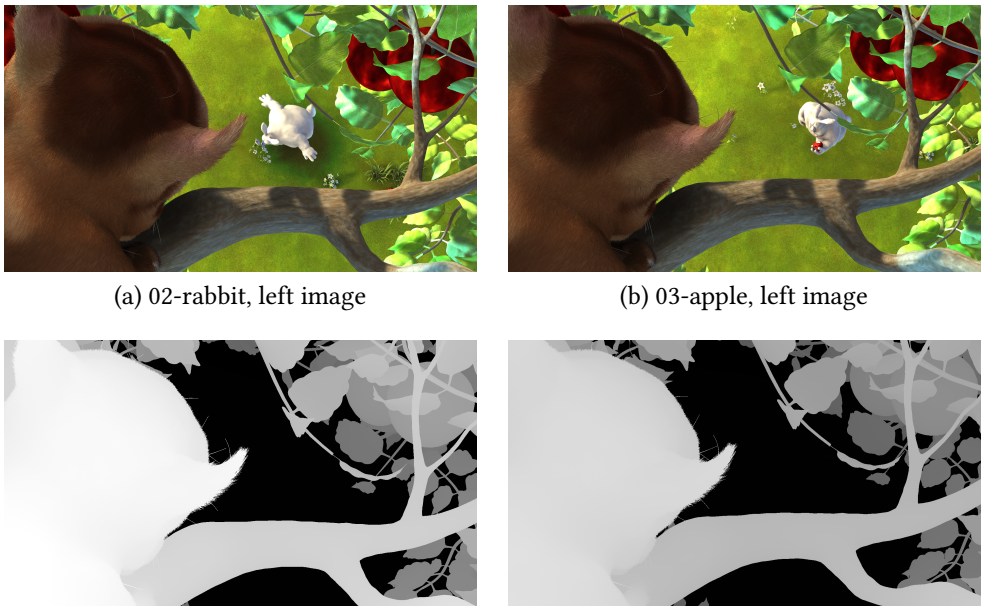
SVDDD - a high-resolution Stereoscopic Video Dataset with precise Depth and Disparity information

As a third dataset, a novel, not yet analyzed dataset was chosen: the SVDDD[3] dataset. The department of Praktische Informatik IV[4] created the dataset with high-resolution video sequences containing accurate depth and disparity information for stereoscopic videos. Figure 5.3 depicts the left image and its ground-truth disparity companion for a small selection of the SVDDD dataset.

[3] SVDDD stands for a high-resolution Stereoscopic Video Dataset with precise Depth and Disparity information.

[4] `http://ls.fmi.uni-mannheim.de/de/pi4/`

The difference here is, that this dataset was not analyzed before. Thus, it is possible that the chosen algorithms do not work properly on this dataset, which may be explained by one of the following reasons: Either the disparity maps are not accurately calculated or the chosen algorithms doesn't apply for the constructed scenery.



(a) 02-rabbit, left image

(b) 03-apple, left image

(c) 02-rabbit, disparity

(d) 03-rabbit, disparity

Figure 5.3: SVDDD high-resolution stereo dataset

Focusing on the latter one, the scenes were created with Blender and utilize open-source scenes from the Big Buck Bunny project[5]. A second camera was added to the scenes to obtain depth information. The camera settings for each scene can be extracted from the Blender files. With those parameters the disparity can be calculated as shown in Chapter 2. The scenes were also adjusted regarding the following points to extract depth-information accurately. Rays of light as well as transparent layers were removed to retain accurate depth-information. Such transparent layers may result in anomalous depth-information as they may bounce back the depth request while obtaining depth information from Blender. Those transparent layers are in between the actual background of the scene, which is visible for the viewer, and the camera. Motion and object blur were reduced as such areas can yield to defective disparity information around the blur. Fain-grained

[5] `https://peach.blender.org`

textures like grass or fine hairs were modified. Without modifying these elements in a scene, a stereo matching algorithm may have problems with those arbitrary textures. The initial scenes of the Big Buck Bunny project contained randomized grass in each image, left and right side. This also yields in the impossibility of stereo matching algorithms to determine the shift of the pixels.

The dataset includes 15 scenes. Each scene consists of high resolution stereo images at 1920 x 1080 pixels with an average bitrate of 94 MBit/s. With an approximate size of 2 MB for each frame, the size of each sequence varies between 0.4 GB and 2.7 GB. For each frame, depth and disparity information are computed with 32-bit floating point precision and saved in OpenEXR files. The above mentioned points apply to the first alpha version of the dataset. During this thesis, the dataset ran through several iterations as more and more problems arose during the evaluation. This is discussed later on.

5.2 Quality metrics

Typical quality measure instruments for comparing disparity maps against their ground-truth reference data are [**?**]:

- percentage of bad matching pixels,
- root-mean-square error,
- parameter-free measures.

As parameter-free measures need modified disparity algorithms [**?**], it was not considered.

Percentage of bad matching pixels

Percentage of bad matching pixels (PBMP) is usually used in comparing the performance of stereo matching algorithms.

$$\mathrm{PBMP} = \frac{1}{n} \sum_{x,y=0} (|d_a(x,y) - d_e(x,y)| > \delta_t) \tag{5.1}$$

d_a stands for the actual disparity whereas d_e symbolizes the expected one. The threshold, denoted with δ_t, can be adjusted and as result, the percentage of outlier pixels, which differ by more than δ_t, is estimated.

Root-mean-squared error

The mean squared error (MSE) as well as the root mean squared error (RMSE) are both the most popular metrics in image and video processing [**? ? ?**]. MSE measures the mean of the squared differences between the intensities of pixels in two pictures at the same position. In conclusion, the average difference per pixel is then the root of the squared error.

$$\text{RMS-Error} = \sqrt{\frac{1}{n}\sum_{x,y=0}(d_a(x,y) - d_e(x,y))^2} \tag{5.2}$$

It represents the sample standard deviation of the differences between predicted values and observed values. Here $d_a(x, y)$ is the actual disparity value for given x and y. $d_e(x, y)$ is our expected disparity value from our ground-truth data. Hence the RMSE is the difference between values on average.

5.3 Measurement

This chapter describes and summarizes the evaluation process. All disparity maps were created on a desktop computer with an i5-2500k @ 3.30GHz (quad-core). First, the idea of a distributed calculation arose, as described in Chapter 4. As this would lack the comparison of runtime, only the desktop computer was used.

The whole evaluation was proceeded in the following manner: Python scripts manage the chained execution of all components. Disparity maps are computed for each frame in a sequence, for each sequence in a dataset, and for each dataset. The Cambridge dataset contains five sequences with each 100 frames. In total, 13 algorithms were executed. This results in a total of 6.000 disparity maps.

In addition, this dataset was duplicated for observing interferences on disparity algorithms. For analyzing the impact of noise, additive gaussian noise with a normal distribution of $\mathcal{N}(\mu, \sigma^2)$ was added. For σ, $10, 20, 30$ were chosen as parameter. Lower values had no noteworthy impact on a few sample disparity maps computations and were also not visually exhibited. Higher values were measured by **?** [**?**], but are clearly disturbing the whole image. As additive gaussian noise should simulate a more real scenery instead of synthetic rendered videos, higher values were not considered.

Due to bandwidth limitations, raw (lossless) high resolution videos are not transmitted without additional compression. That said, H.265, which was introduced in Chapter 4, was chosen as current state of the art video compression codec. For

analyzing the resulting inteferrence on disparity algorithms, video compression artifacts were added by converting the images to a video, applying the codec and unwrapping the video into images. As parameter, the constant rate factor (CRF) has been adjusted. The default parameter value is 28. As 0 means lossless video compression, it was tested with a few sample disparity map computations. 0 had no impact and was not considered for the evaluation. 51 was not considered as well, as this totally disturbs the image visually, even if there is no motion between two consecutive frames. $14, 28, 40$ were chosen as final values to have a feasible amount of different values and with an equal distribution in the available parameter range.

Taking those combinations into account yields to an amount of disparity maps to be computed of 48.000. After the disparity maps were computed, other Python scripts were written and executed for the aggregation of the results. As the results are for each image in a sequence from a dataset and also focusing on one algorithm, they have to be aggregated and written in summarized CSV files. The simple stereo matcher was treated as an exception. The reason for this is simple: all the scripts were written with the intent of calculating single independent disparity maps. As the simple stereo matcher features spatiotemporal consistency, scripts focusing especially on the execution and aggregation of this matcher were written as well.

Parameter tuning

The observed parameters regarding noise and video compression were described before. In addition, most of the available parameters were described accurately in Chapter 4. Adjusting and quantifying also the parameters of each algorithm would be a formidable task. Thus, only the maximum disparity was adjusted according to the dataset. All the other variables were left as default and moreover, some algorithms have no parameter but the maximum disparity to adjust.

Visualization

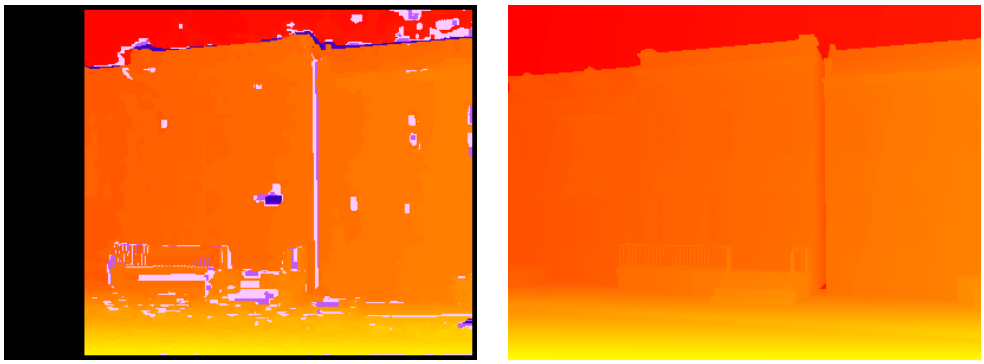
As it may be hard to identify fine-grained changes in grayscale images and also the human eye tends to be more sensitive to observe color changes, heatmaps are used to visualize the results. For visualization, the heatmaps are created with the OpenCV autumn color scheme. The color-scale of this theme is depict in Figure 5.4.

[6]Source (accessed 04/2016): `http://docs.opencv.org/3.1.0/d3/d50/group__imgproc__colormap.html`.

Figure 5.4: OpenCV autumn color scale[6]

Outliers are marked with three different purple color steps in the heatmaps. The darker the purple color is, the more hard the error was. The light blue means $1px$ threshold was exceeded, the color in-between the light blue and the dark purple denotes $2px$ threshold was exceeded and the very dark purple reflects $4px$ threshold was exceeded. The border is excluded from the evaluation (cf. Chapter 2) and marked black. Brighter color means more heat and marks nearer pixels. An example shot can be seen in Figure 5.5.

The maximum disparity in this example shot was 64. This means that due to the baseline separation of the cameras, and the resulting shift of the pixels, some pixels can not be calculated as they have no counterpart. This was excluded from the evaluation. Also, as the area-based disparity algorithms work with a window size and have a small step from the top and the bottom, this was excluded as well. Both is represented with a border mask, which was applied to all evaluation steps beforehand. This can also be seen in Figure 5.5. This guarantees a streamlined and balanced evaluation.



(a) outliers heatmap (b) ground-truth heatmap

Figure 5.5: Example heatmaps for outliers heatmap and disparity ground-truth.

Applying disparity algorithms on videos

As said in the implementation Chapter 4, disparity algorithms for image can be applied to videos. Although the frames of a video can be seen as independent pieces, different anomalies can be further investigated while analyzing videos:

- outliers in the form of a single frames which differs too much from the others,
- impact of additive Gaussian noise to simulate real scenery,
- impact of disturbing effects like artifacts from video compression,
- the benefit from creating a naive spatiotemporal context.

The following sections present only the highlights of the results as it is an tremendous amount of data, which were computed during the evaluation.

5.4 Results

The highlights are illustrated as the amount of information, which were obtained during the measurement is huge. The following Table 5.1 contains the identifier which are used throughout the upcoming subsections. Algorithm (10) OpenCV - simple BM utilizes the same implementation as (2) OpenCV - BM but without performing a left-to-right consistency check and WLS filtering, as described in Chapter 4. The following subsections are rationed in the results against the reference dataset, the general performance of the Cambridge dataset as well as the impact of interferences like noise and video compression. Afterwards, the runtime and the SVDDD dataset are outlined. As of now, the spatiotemporal approach with unweighted matching cost is denoted as *SNSM - STU* whereas the weighted spatiotemporal approach is denoted as *SNSM - STW*.
The general procedure is to describe the high level results of algorithms operate on a whole sequence with PBMP_{1px} utilizing the non-occluded mask. As the focus is on PBMP_{1px}, lower results are better. The cell of the best result in each row is marked with green background color, the worst with red. Then, the other masks are outlined in greater detail. Afterwards, particular outliers regarding the whole sequence are discussed. The same procedure is then applied to impact of noise and impact of video compression.

5.4.1 Against reference dataset

As reference scene the *Head and lamp* scene of the Tsukuba dataset was chosen. For evaluating against the reference dataset, all presented disparity algorithms computed disparity maps. These disparity maps are then compared in combination with different masks. The result can be seen in Table 5.2 and the SNSM is depicted in Figure 5.6. Compared with the Middlebury stereo benchmark[7], similar

[7] `http://vision.middlebury.edu/stereo/eval/`

Id	Sequence
1	book
2	street
3	tank
4	temple
5	tunnel

Id	Algorithm
1	OpenCV - SGBM
2	OpenCV - BM
3	ELAS
4	MRF - ICM
5	MRF - GC Expansion
6	MRF - GC Swap
7	MRF - BP TRWS
8	MRF - BP BPS
9	MRF - BP BPM
10	OpenCV - Simple BM
11	SNSM
12	SNSM - STU
13	SNSM - STW

Table 5.1: Identifier for results

results could be achieved.

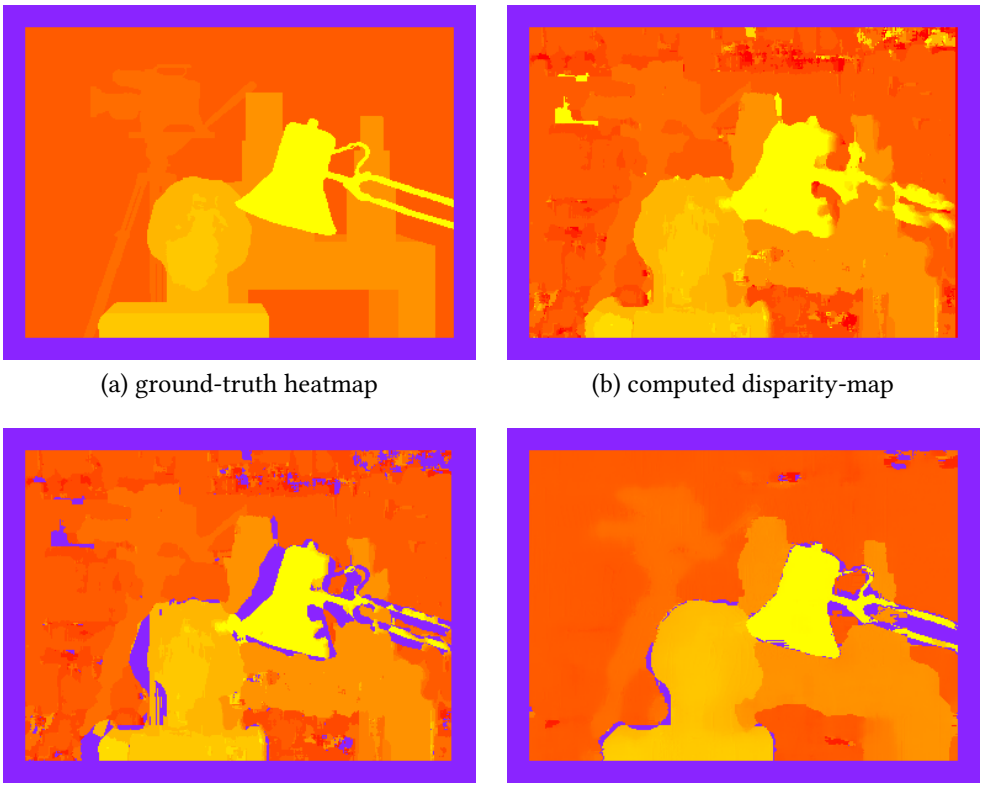
The SNSM utilizes SAD as matching cost function and does not further disparity refinement. The results from SNSM with a threshold of 1 and a block size of 9 are PBMP$_{all,1px}$ = 11.51% and RMSE$_{all}$ $= 1.78px$. According to the Middlebury stereo benchmark, the other SAD implementation (SAD-IGMCT) [**?**] achieved with a threshold of 1: PBMP$_{all,1px}$ = 7.14%. The difference may come from smoothing, which SAD-IGMCT included as a disparity refinement step, which is visible while looking at their outcome. The other results are similar to the results in the Middlebury stereo benchmark.

Some deviations like SAD-IGMCT to SNSM regarding all pixels and a threshold of 2 are huge but can be explained with disparity refinement steps which the SNSM does not perform. The best result is achieved with a variant of a global method, belief propagation and yields to 4.97% of bad matching pixels regarding all pixels and applying a threshold of 2. ELAS performs pretty good but struggles with a very accurate disparity map while comparing the results for PBMP$_{all,1px}$. The simple block matching algorithm of OpenCV (10) is the same as (2), but without the disparity refinement step and a left-right-consistency check. This may explains the very bad results. Surprisingly, the global methods do not outperform the local methods by far.

In the beforehand mentioned Figure 5.6, SNSM struggles with the disparity constraints as they are not forced programmatically. For each point (x, y) the disparity is calculated individually. This leads to a more noisy disparity map than for in-

	$RMSE_{all}$	$PBMP_{all,1px}$	$PBMP_{all,2px}$	$PBMP_{disc,2px}$
(1) OpenCV - SGBM	1.49px	7.00%	4.73%	11.99%
(2) OpenCV - BM	1.45px	10.27%	5.92%	12.40%
(3) ELAS	1.23px	7.07%	4.42%	12.08%
(4) MRF - ICM	5.02px	22.32%	18.83%	19.72%
(5) MRF - GC Expansion	2.56px	5.59%	4.74%	12.86%
(6) MRF - GC Swap	2.79px	5.55%	4.70%	12.74%
(7) MRF - BP TRWS	2.48px	5.41%	4.54%	12.32%
(8) MRF - BP BPS	2.64px	9.49%	9.07%	16.27%
(9) MRF - BP BPM	2.55px	5.34%	4.97%	12.99%
(10) OpenCV - Simple BM	6.10px	28.27%	26.42%	28.99%
(11) SNSM	1.78px	11.51%	9.85%	18.97%
SAD-IGMCT	-	7.14%	5.80%	18.90%

Table 5.2: Result table for reference dataset



(a) ground-truth heatmap

(b) computed disparity-map

(c) outliers in SNSM

(d) outliers in SAD-IGMCT

Figure 5.6: SNSM heatmaps for Tsukuba scene

stance the disparity map of SAD-IGMCT, which respects the disparity constraints, takes the left-right-consistency into account and performs a disparity refinement step. But it also shows that by implementing a simple stereo matcher with just the SAD as matching cost measurement step reasonable results can be achieved. The semi-global stereo matcher implementation of OpenCV achieved the best result in $\text{PBMP}_{disc,2px}$, which focuses on the disparity estimation near depth-discontinuities at object borders.

5.4.2 General performance

From a theoretical perspective, the assumption is that the algorithms struggle with depth-discontinuity regions and arbitrary surfaces, i.e. textureless regions. Also, global methods should perform slightly better than local methods. Salient regions may a bit more random as they take the spectrum of the image into account and not the nature composition of pixel groups like textures. As it would be a tremendous task to deliver all possible plots, only the highlights were included.

The following Tables 5.3 and 5.4 contain the general performance with the focus on $\text{PBMP}_{noc,1px}$.

	1	2	3	4	5	6	7	8	9
1	16.60%	30.98%	7.31%	19.55%	8.30%	8.74%	8.32%	24.35%	6.93%
2	6.88%	13.90%	7.18%	30.75%	10.91%	11.19%	10.90%	13.10%	8.32%
3	9.54%	28.14%	3.99%	11.18%	4.17%	4.25%	4.10%	6.31%	3.35%
4	13.98%	18.86%	9.90%	23.53%	8.01%	10.47%	7.75%	10.20%	11.65%
5	5.96%	16.13%	0.19%	1.37%	3.41%	3.34%	3.42%	5.68%	0.62%
∅	10,59%	21,60%	5,71%	17,28%	6,96%	7,60%	6,90%	11,93%	6,17%

Table 5.3: Result table for general performance, focusing on $\text{PBMP}_{noc,1px}$

Focusing on the general performance of all but SNSM, ELAS outperforms in mean of all sequences the other global and local methods. The worst result is achieved with the OpenCV block matching implementation. It is also impressive that ELAS also outperforms traditional global methods, like graph cuts and belief propagation. None the less, it has to be mentioned, that the performance of global methods may vary intensely depending on the formulation of the underlying energy function [**? ? ?**].
It is interesting to see, that the simplest variant of the block matching implementation in OpenCV is outperformed by just a SAD matching cost implementation, the SNSM. Also, the best results are achieved with the weighted spatiotemporal approach, although the approach was not superior in at least one sequence, but overall.

	10	11	12	13
1	32.61%	8.72%	10.07%	9.65%
2	25.64%	11.79%	8.76%	8.90%
3	13.26%	6.08%	8.71%	7.29%
4	38.96%	12.98%	11.15%	11.26%
5	8.60%	0.93%	4.54%	2.15%
∅	23,81%	8,10%	8,66%	7,85%

Table 5.4: Result table for general performance, focusing on $\text{PBMP}_{noc,1px}$

Depth-discontinuity regions

Figure 5.7 depicts the trend of the percentage of bad matching pixels near depth-discontinuity areas. It is clear to see that global methods perform better than local methods near object borders. The worst results are achieved by simple block matching algorithms. The best result is achieved by belief propagation, which tends to produce feasible results near object borders, also regarding other sequences.

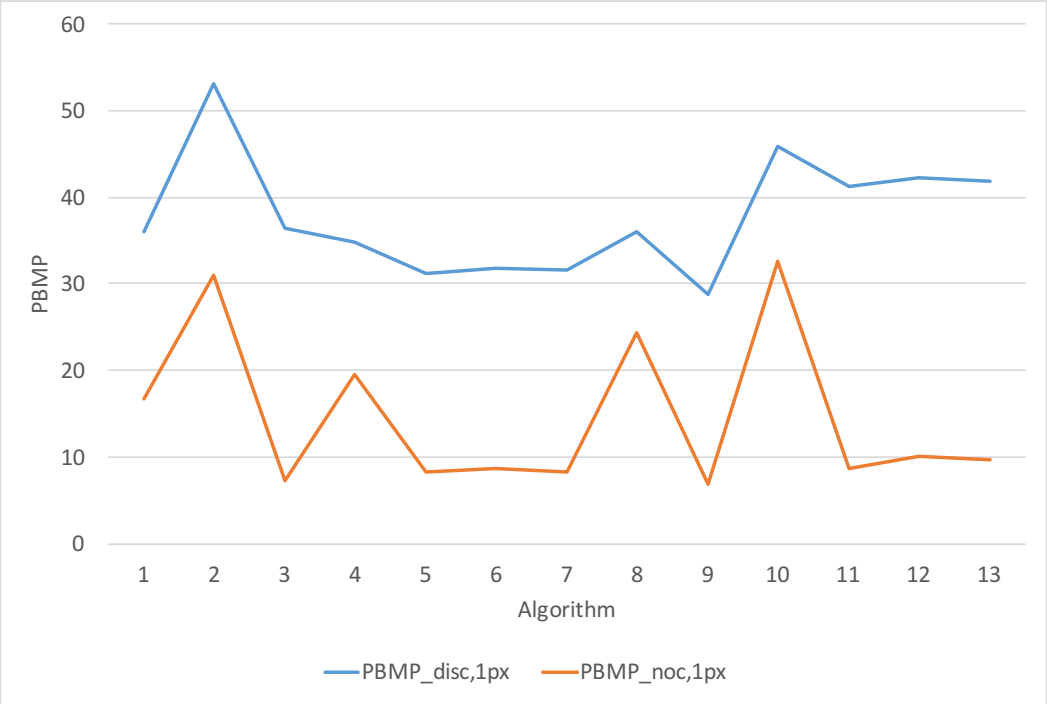


Figure 5.7: Depth-discontinuity mask applied on the book sequence with $\text{PBMP}_{disc,1px}$.

The graph shows also the correlation of the two metrics $\text{PBMP}_{disc,1px}$ and $\text{PBMP}_{noc,1px}$. This means that each algorithm performs approximately the same way, either on

depth-discontinuity areas or on non-occluded pixels with small positive displacement along the x-axis. Concluding, the only algorithms which perform significantly worse than the others are the (2) OpenCV block matching implementation with filtering and (10) the simple OpenCV block matching implementation without filtering. Although algorithm (4) MRF - ICM performs not as well as the others, it handles object borders at which depth-discontinuity occur more accurate than the other two outliers. As this was just an example plot for depth-discontinuity areas, the other sequences behave the same way.

Textureless regions

The next Figure 5.8 compares the amount of bad matching pixels in non-occluded regions with the textureless regions. For this comparison, the tunnel scene of the Cambridge dataset was chosen. It is clear to see that regions without textures, i.e. arbitrary surfaces are worse detectable than compared with textured objects.

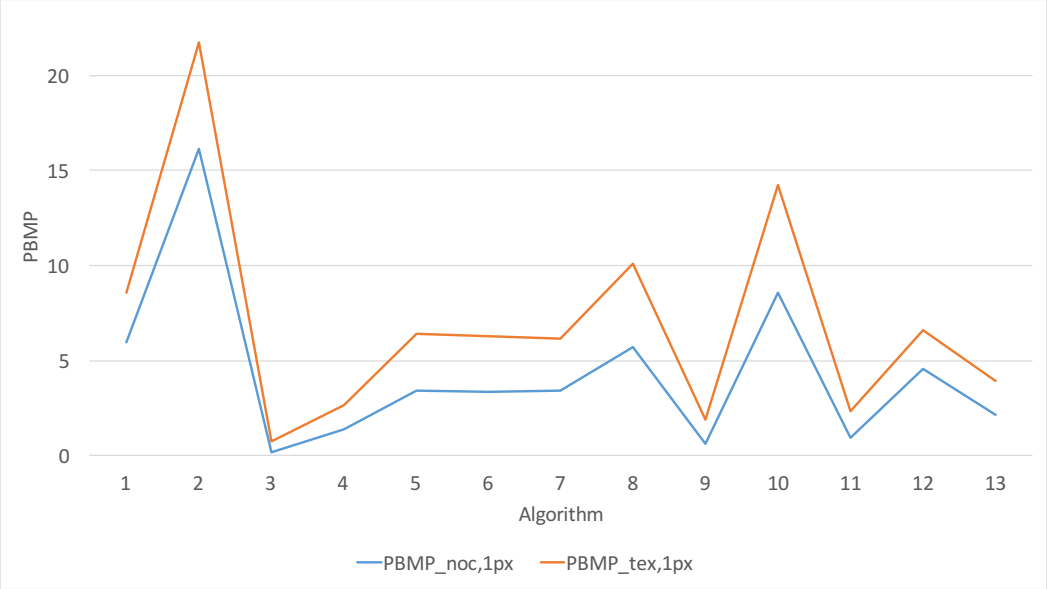


Figure 5.8: Chart of textureless region mask applied on the tunnel sequence.

The percentage of bad matching pixels in textureless regions is nearly the same as overall in non-occluded areas, but with a positive displacement parallel to the x-axis. It is also impressive that overall on average the ELAS algorithm yield to only 0.19% of bad matching pixels in non-occluded regions $\text{PBMP}_{noc,1px}$. In comparison, the worst algorithm, in this particular case the OpenCV block matcher implementation, resulted in $\text{PBMP}_{noc,1px}$ = 16.13%.

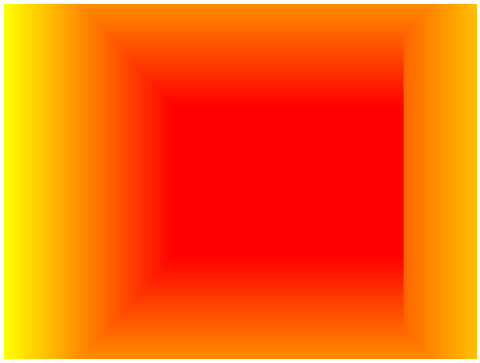
	$RMSE_{noc}$	$PBMP_{noc,1px}$	$PBMP_{noc,2px}$	$PBMP_{noc,4px}$
(3) ELAS	0.36px	0.19%	0.08%	0.04%
(10) OpenCV - Simple BM	5.82px	8.60%	8.58%	8.56%
(11) SNSM	0.96px	0.93%	0.71%	0.54%
(12) SNSM STU	1.02px	4.53%	1.24%	0.73%
(13) SNSM STW	0.92px	2.15%	0.79%	0.59%

Table 5.5: Result table for tunnel scene of Cambridge dataset

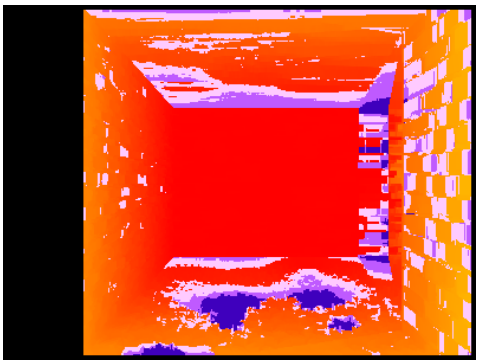
The SNSM also lead to feasible results which can be seen in Table 5.5.

For a direct comparison of the visual outcome of the disparity maps, Figure 5.9 depicts four images. The left image with its ground-truth companion, the worst result of the OpenCV block matcher implementation and the best outcome yielded by ELAS. ELAS seems to determine the disparity map in a nearly perfect manner but seems to struggle with the beginning shadow.

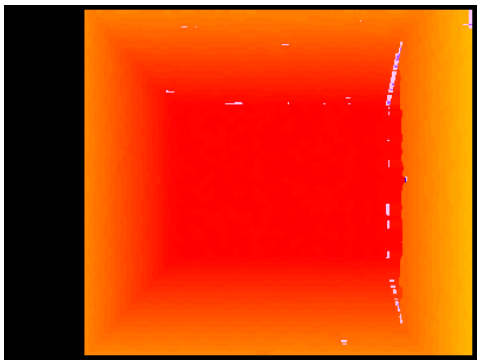
(a) left image



(b) ground-truth heatmap



(c) outliers in OpenCV BM



(d) outliers in ELAS

Figure 5.9: Heatmaps for the tunnel scene

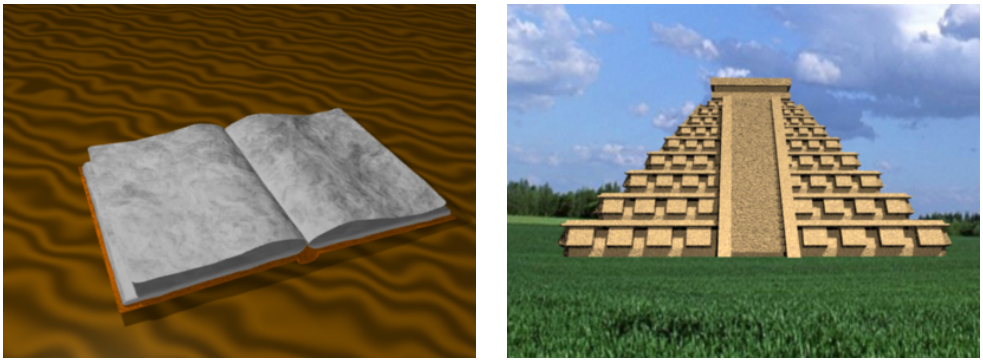
The SNSM led to feasible results in the tunnel scene, as can be seen in Table 5.5. The unweighted approach led to slightly worse overall results. The interesting part here is the weighted approach of the SNSM implementation. The RMSE_{noc} improved whereas the $\text{PBMP}_{noc,1px}$ yield to shoddy results. Considering both metrics, it shows that the naive spatiotemporal approach does not respect motion at all. The slightly better RMSE_{noc} feels a bit random in the overall evaluation.

Salient regions

Focusing on salient regions is a novel approach. The distribution of salient regions does not follow any visual attracting regions. According to **?** [**?**], the main idea is to extract the pixels which jump out of smooth curves, as they deserve most certainly the humans attention. Their approach is implemented in the OpenCV library and no parameter can be set. It is named static saliency, as only images are considered. Thus, it is a pretty basic novel approach to create feasible masks with the static saliency. The overall visual results are a bit confusing. Figure 5.10 shows two sample images and their corresponding computed saliency mask. As can be seen in the figure, the static saliency approach in OpenCV yields to confusing results. Although in both images, same parts, that are classified as salient regions, are comprehensible and understandable, some are not.
The book as a whole is not recognized as salient as well as the stairs of the temple are left out completely. The chart in Figure 5.11 depicts the application of the saliency mask to the book sequence.
The saliency mask has no real informative value. There is no clear correlation identifiable. In some cases, the saliency mask covers also depth-discontinuity areas and occluded areas. Thus, it can yield to worse results than $\text{PBMP}_{noc,1px}$.
In other scenes of the Cambridge dataset, the results were similar to the book sequence. Within high-resolution images, like the SVDDD dataset, the saliency mask looked even more unpredictable. No clear salient regions were identifiable. The assumption is that the algorithm, which is implemented in the OpenCV library is may be not capable of high-resolution images. None the less, it could be shown, that the evaluation does not benefit from such a saliency mask.

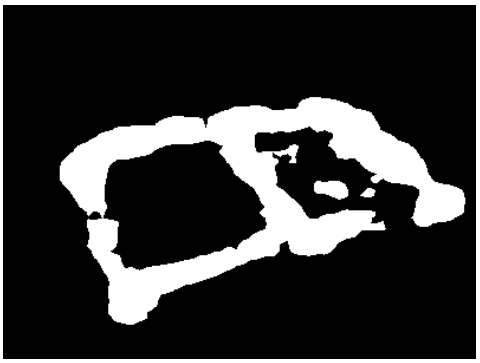
Outliers

Beforehand, the investigation of outliers in a whole sequence of consecutive frames was mentioned. During the evaluation, each sequence tended to have outliers. To give an example, the book scene in combination with the (3) ELAS algorithm was chosen. In this scene, two frames have significant more bad matched pixels than all the others, 23 and 26 although the margin is low. The scenes shows a book. During 41 frames one page will be turned over.



(a) image of book scene

(b) image of temple scene



(c) corresponding saliency mask

(d) corresponding saliency mask

Figure 5.10: Examples for salient masks

The Figure 5.12 shows that in the beginning and in the end of the scene, exactly the point, in which the page is turned over and forms a flat surface with the book. During the movement of the page, more object borders are created as the thin page forms a new object. In frame 23 and 26 the turn over from one side to the other is made. Thus, in exactly this two frames the amount of depth-discontinuity borders is the highest. Figure 5.13 depicts these frames. The increasing error at those object borders can be identified. It is also clearly visible that Frame 23 has the highest percentage of bad matching pixels.

5.4.3 Impact of noise

The impact of noise is an interesting and not often evaluated approach regarding disparity algorithms. For this observation five different datasets were created with the image diminisher. The Cambridge dataset was cloned and each image in this dataset was diminished with a newly created noise mask. The noise mask was added on top of the image matrix. Different values were chosen as mentioned

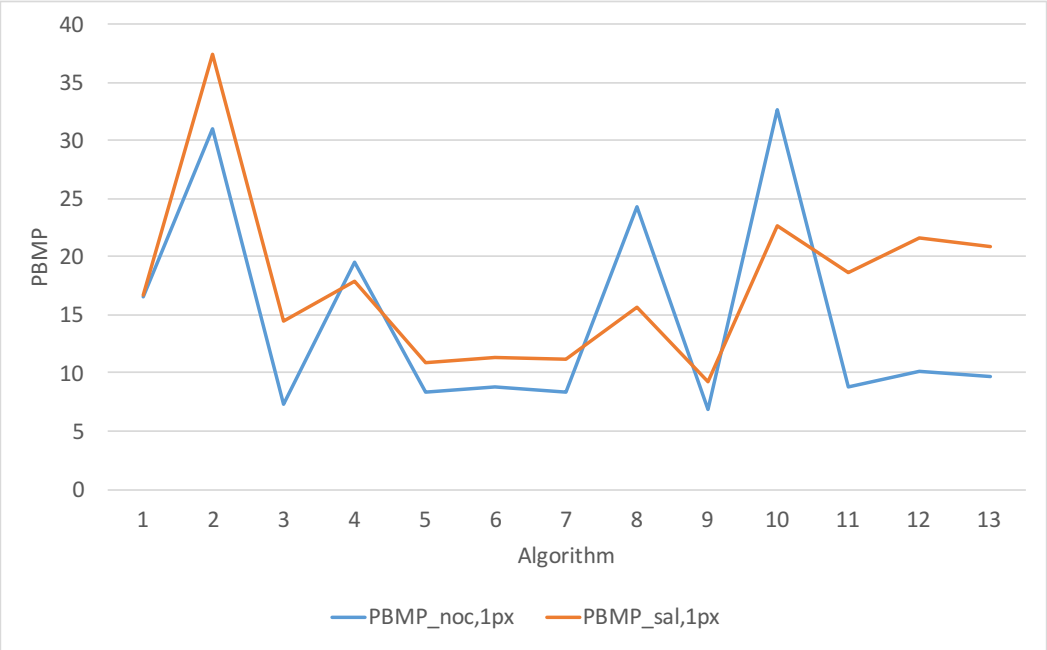


Figure 5.11: Chart of salient region mask applied on the book sequence.

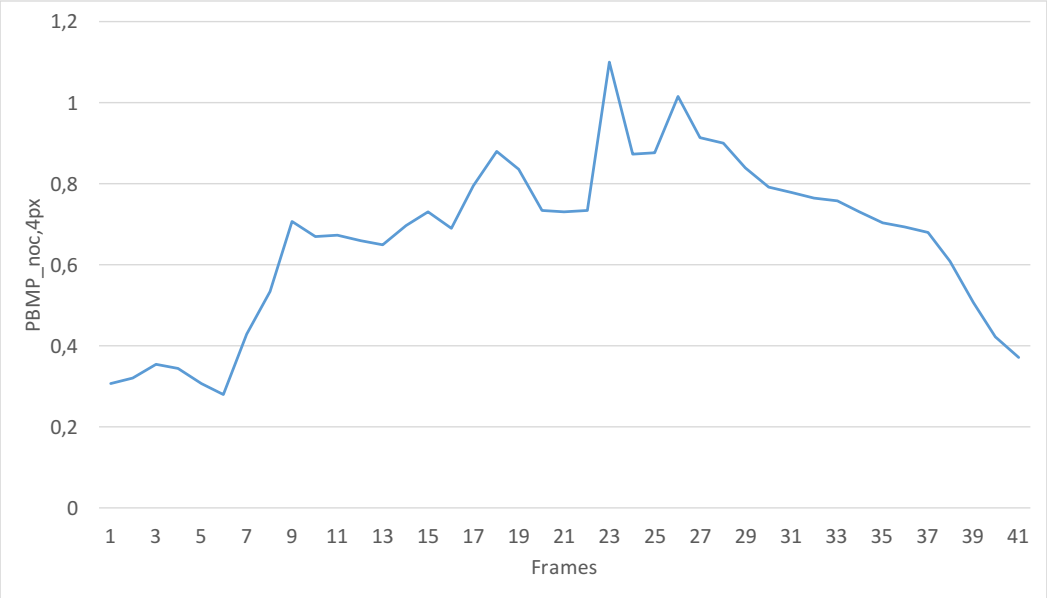
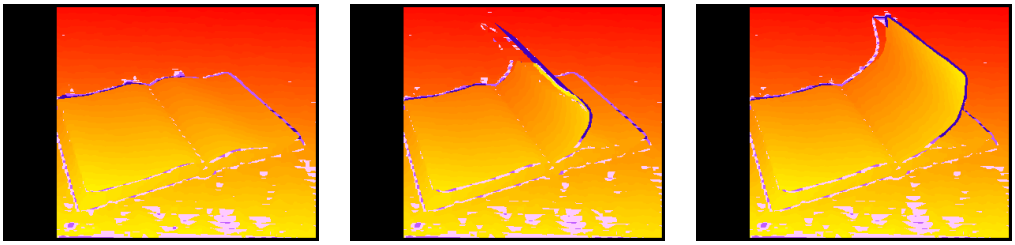


Figure 5.12: Chart of general outliers in a sequence.

in the measurement section. For demonstration purpose, a plot, which contains



(a) Frame 1 (b) Frame 23 (c) Frame 26

Figure 5.13: Examples for general outliers in the book sequence. The disparity maps are computed with the (3) ELAS algorithm.

all three $sigma^2$ values was chosen. Figure 5.14 shows the impact of different σ^2 values for additive Gaussian noise on the computation of disparity algorithms.

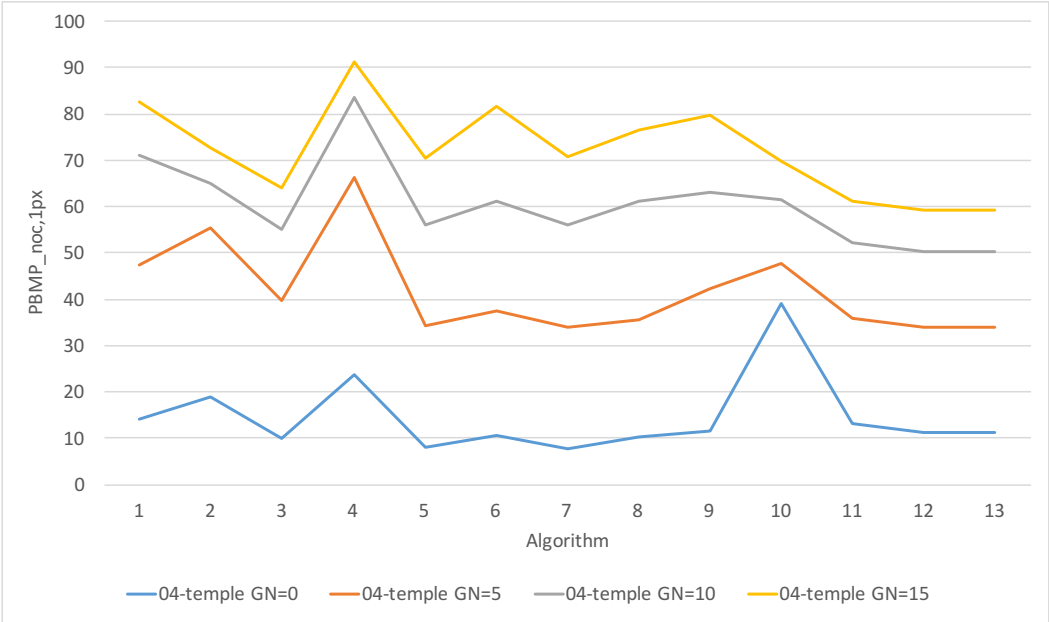
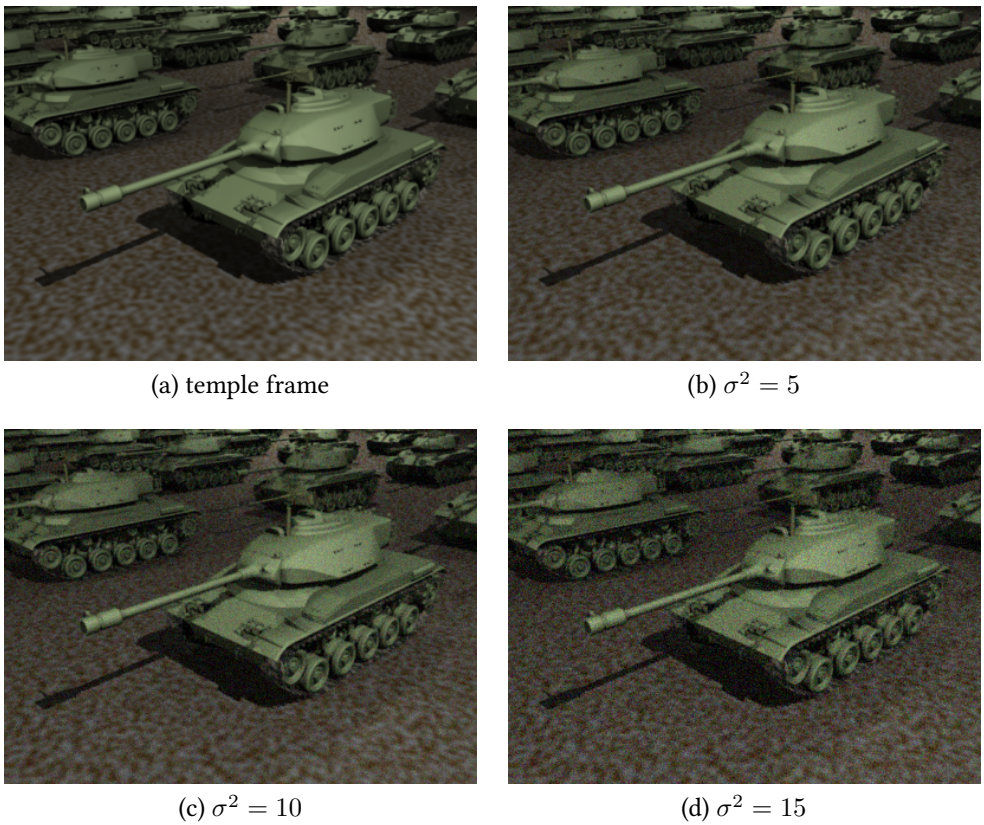


Figure 5.14: Chart of the impact of different σ^2 values for additive Gaussian noise on the result of disparity algorithms focusing on $P_{noc,1px}$.

Here, the tunnel scene was picked to demonstrate the impact of normal distributed noise. Although, as mentioned beforehand in Chapter 4, the idea follows the methodology of **?** [**?**], it is not clear if additive Gaussian noise is an adequate image diminishing effect to simulate real scenery. None the less, it is nice to see that all picked metrics correlate with each other. The extent of the displacement along the x-axis depends on the amount of disturbing noise was added. Figure

5.15 depicts the first frame of the tunnel sequence with different noise degrees. Added noise with a $sigma^2$ of 5 does not seem to distract the visual appearance of the image. None the less, it has a direct impact on the performance of disparity algorithms. The degree of noise with a σ^2 of 15 is visually noticeable. One reason for that might be, that this type of noise may not appear in real-life while capturing a scene with a CCD sensor. The approach follows the methodology of **?** [**?**] but may not be suitable for simulating real scenery. But, for the purpose of simulating an image, captured via CCD sensor, camera noise models exist [**?**]. The approach with additive Gaussian noise changes every pixel in the image, by a different amount. But an image, captured via CCD sensor, does not seem to have this pattern (cf. [**?**]). Changing every pixel, even by a small amount means that the calculated matching cost may differ largely. This is of course depending on other factors, such as window size and which matching cost formula is applied. The assumption remains, that additive Gaussian noise may not be a good choice to simulate real scenery and to measure the in uence on disparity algorithms.



(a) temple frame (b) $\sigma^2 = 5$

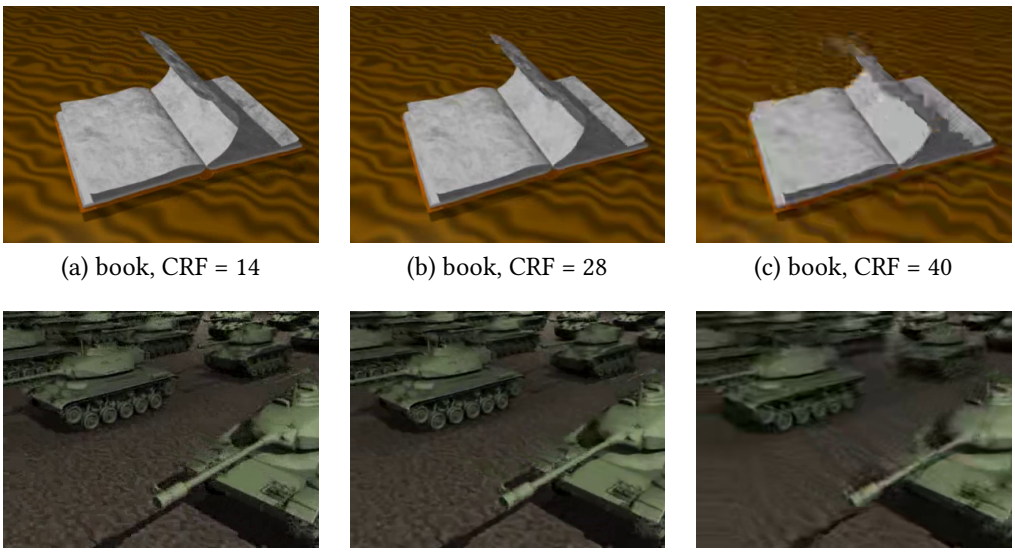
(c) $\sigma^2 = 10$ (d) $\sigma^2 = 15$

Figure 5.15: Examples for image diminishing effects with Gaussian noise

The upcoming subsection also deals with image diminishing effects, in this case video compression.

5.4.4 Impact of video compression

The impact of video compression is an interesting and novel approach. High-resolution videos, due to its nature, are not shipped for television as uncompressed raw data or even lossless compressed. As seen in the Chapter 3, there are different applications for disparity algorithms. One outpointed need is the remapping of disparity for different screen sizes. If this can not happen on raw data, for instance the data is missing and only the compressed version is available, the influence of video compression on the outcome of disparity algorithms is observed. For this observation five different datasets were created with the image diminisher. As underlying video codec H.265 was chosen because it is currently the most efficient video compression technique. The recommended default parameter for H.265, the constant rate factor (CRF) is 28. It ranges from $0-51$, whereas 0 represents lossless and 51 the worst possible compression. $0, 14, 28, 40, 51$ were chosen with 14 and 40 as additionally in-between parameters. 0 and 51 were left out, as the result is too visually distracting for the viewer with 51 and 0 is lossless. Thus, the observed values are $14, 28, 40$.



(a) book, CRF = 14 (b) book, CRF = 28 (c) book, CRF = 40

(d) tanks, CRF = 14 (e) tanks, CRF = 28 (f) tanks, CRF = 40

Figure 5.16: Examples for image diminishing effects caused by video compression artifacts of the H.265 codec in combination with different values for CRF.

Figure 5.16 depicts the effect of different CRF values using H.265. It is clearly visible that a CRF with 14 leads to feasible results regarding the experience for viewers. A CRF with 28 and 40 lead to more distracting artifacts. Figure 5.17 illustrates the correlation of the H.265 CRF and PBMP$_{noc,1px}$. Even the lowest value in test series, 14 leads to a huge amount of bad matching pixels. A CRF with 28 seems to as hard as 14 for disparity algorithm, as the amount of bad matching pixels is nearly identical with the amount of 14.

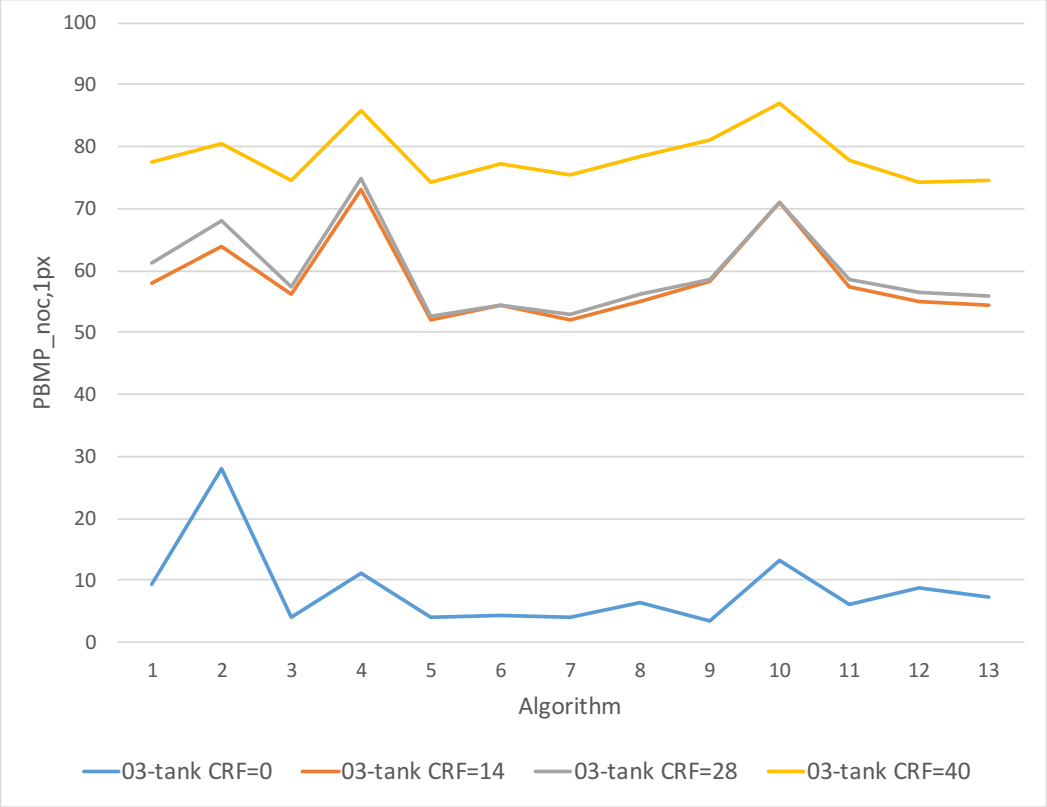
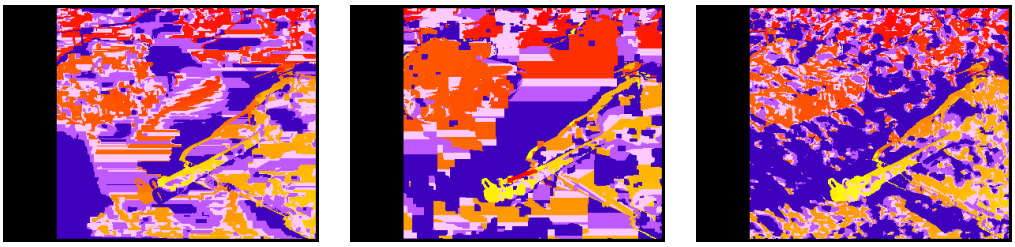


Figure 5.17: Chart of the impact of different CRF values for H.265 video compression on the result of disparity algorithms focusing on PBMP$_{noc,1px}$.

Figure 5.18 depicts the computed disparity maps after diminishing the whole scene with a CRF value of 40. Many disparity information seem to be lost. The structure of the fire pipe of the right tank is still visible and the disparity information are valid within a threshold of $1px$. The rest feels a bit random.

Talking about SNSM, it lead to better results than most of the other algorithms. In the tank sequence, (11) SNSM achieved PBMP$_{noc,1px}$ = 80.72% where as (8) MRF - BP BPM achieved 82.45%, which led to superior results beforehand. The unweighted

and weighted approach were able to improve this result to 75.04% and respectively 75.58%.



(a) (3) ELAS outliers (b) (5) MRF GC Swap outliers (c) (13) SNSM STW

Figure 5.18: Example of computed disparity maps with video compression. CRF is set to 40. Frame 23 of the tanks scene.

5.4.5 Runtime

Talking about runtime, the results differ vastly. Figure 5.19 illustrates the different runtimes in mean over all sequences of the Cambridge dataset. This should provide a solid overview. It is clear to see that four variants run fast, i.e. have a low runtime. This is true for the local methods, as well as ELAS. The global methods need much more time to approximate the energy function.

Especially the algorithm variants of belief propagation needed much time. Throughout the whole evaluation, (8) MRF - BP BPM led to the best results, compared with other global methods. The runtime is according to algorithm (7) MRF - BP TRWS and (9) MRF - BP BPM the lowest. Comparing with other results, which were outlined in the sections before, the runtime is not justified as the longer an algorithm runs does not automatically yield to better results.

The (11) SNSM is the worst local method focusing on the runtime in comparison with (1), (2) and (10), which are all OpenCV implementations of block matching algorithms. This may result as the whole disparity space image is calculated beforehand the disparity maps are computed. So the runtime of the SNSM is basically the runtime for the whole scene divided through the amount of frames. Both variants, the (12) SNSM-STU and (13) SNSM-STW are dealing with the same information in the same way besides the assembly of the matching cost. But this should not affect the runtime at all, so it is also clearly to see, that a certain deviation of the runtime has to be kept in mind.

ELAS outperformed in many times all the other algorithms according to $PBMP_{noc}$. The runtime is feasible for real-time applications but slightly worse than simple local methods, like the OpenCV block matcher implementation. Although, it has to be kept in mind, as outlined in Chapter 4, that ELAS is a library which comes with a front-end and an internal image library. Both can be the origin for the runtime being worse than simple block-matching algorithm. For an accurate comparison which approach is faster, both should be implemented utilizing the same underlying image library.

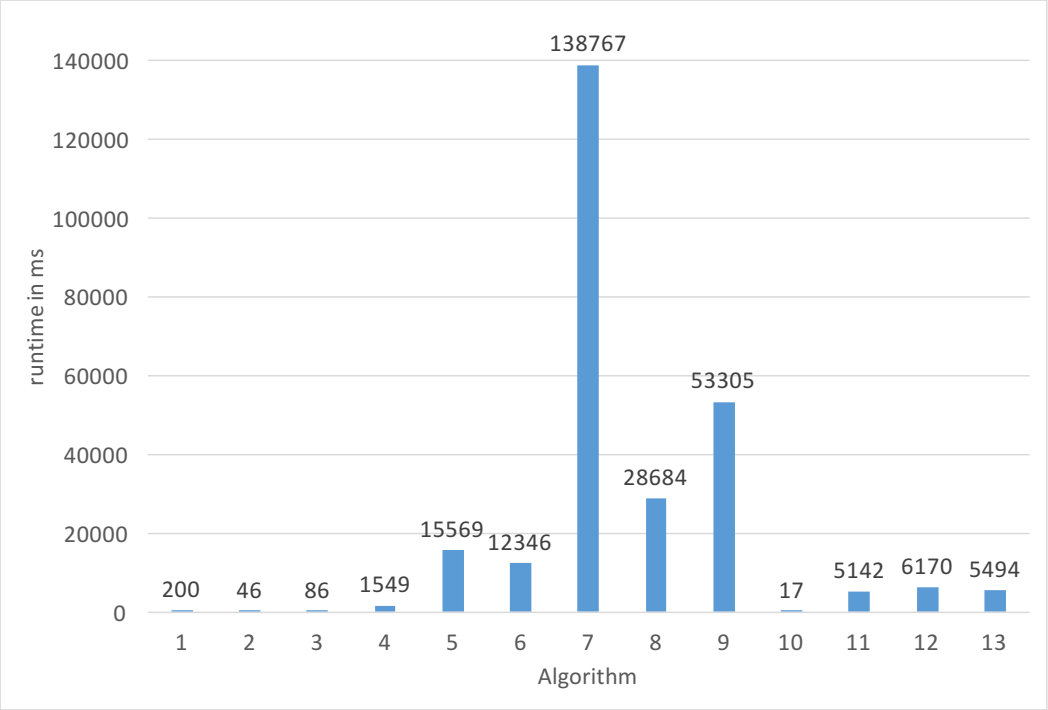


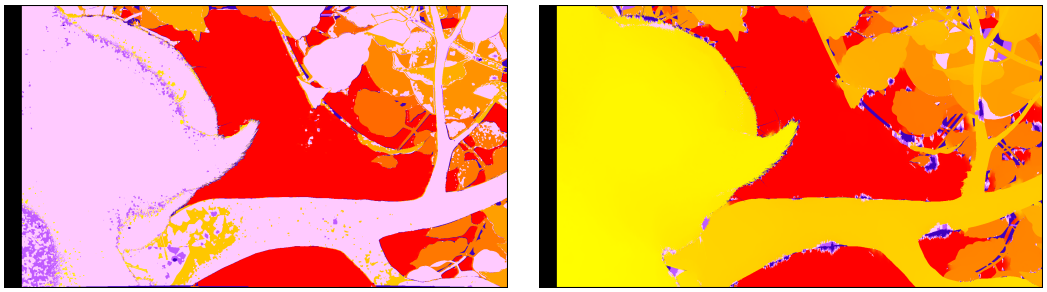
Figure 5.19: Comparison of the runtime of different disparity algorithms

5.4.6 SVDDD

The SVDDD dataset from the department of Praktische Informatik IV[8] was evaluated as well. During this thesis, the dataset ran through several iterations as more and more problems arose during the evaluation. In addition, some algorithms were not able to process high-resolution stereo images. Thus, it is separated from the other results.

[8] `http://ls.fmi.uni-mannheim.de/de/pi4/`

The initially evaluated scenes led to strange results. Some areas were computed correctly but some areas were completely off. One major point, which can lead to such strange, defective results, is negative disparity. Most of the algorithms expect that disparity values range from $0 - d_{max}$. Even a small amount of negative disparity can conduct to a result, which is slightly off, i.e. three or four pixels. Looking at PBMP_{1px} , which can yield to nearly 50%. An example can be seen in Figure 5.20, illustrating two images. In the left image about 10% pixels overall contains slightly negative disparity whereas in the right image, the disparity ranges from $0 - 59$.



(a) Negative disparity (b) Only positive disparity

Figure 5.20: Comparison of computed disparity maps regarding negative disparity.

Another challenge of high-resolution datasets is the huge computational power, which is necessary to compute disparity maps for them. The BP algorithms would have needed about $52GB$ of memory to calculate the scenes for $d_{max} = 64$ as the labels for all possible states have to be created and stored in memory. Also, the SNSM wanted to allocate about $35GB$ of memory as the disparity space image is computed *a priori* before the actual calculation of the disparity maps takes place. The runtime also gives a feeling how computational powerful and complex it is, to perform a disparity estimation on high-resolution stereo images. Summing up, the MRF BP algorithms as well as the SNSM were excluded from the evaluation of the SVDDD dataset.

	1	2	3	4	5	6
02-rabbit-neg	58.62%	61.51%	59.99%	60.58%	57.12%	57.13%
02-rabbit	1.68%	4.31%	2.98%	3.82%	0.65%	0.68%
03-apple	1.69%	4.10%	3.11%	3.44%	0.63%	0.65%
∅ (w/o neg)	1.69%	4.21%	3.05%	3.63%	0.64%	0.67%

Table 5.6: Result table for general performance of SVDDD, focusing on $\text{PBMP}_{noc,1px}$

Table 5.6 contains the mean results of the SVDDD dataset. The best result is achieved with the MRF GC Expansion algorithm, which only led to 0.64% of bad matching pixels in all scenes which is quite impressive for both, the scenes and the algorithms. OpenCV SGBM and ELAS also lead to good results. Comparing the runtime of such high-resolution scenes, the (2) OpenCV BM clearly outperforms the others. Taking the general performance into account, OpenCV SGBM and ELAS both yield to feasible results. For real-time applications ELAS stands out against the other algorithms. Taking both, the overall performance and the runtime into account, ELAS performs best. In Figures 5.21 and 5.22, it is clearly to see that ELAS also performs better than the other local methods regarding $\text{PBMP}_{noc,4px}$.

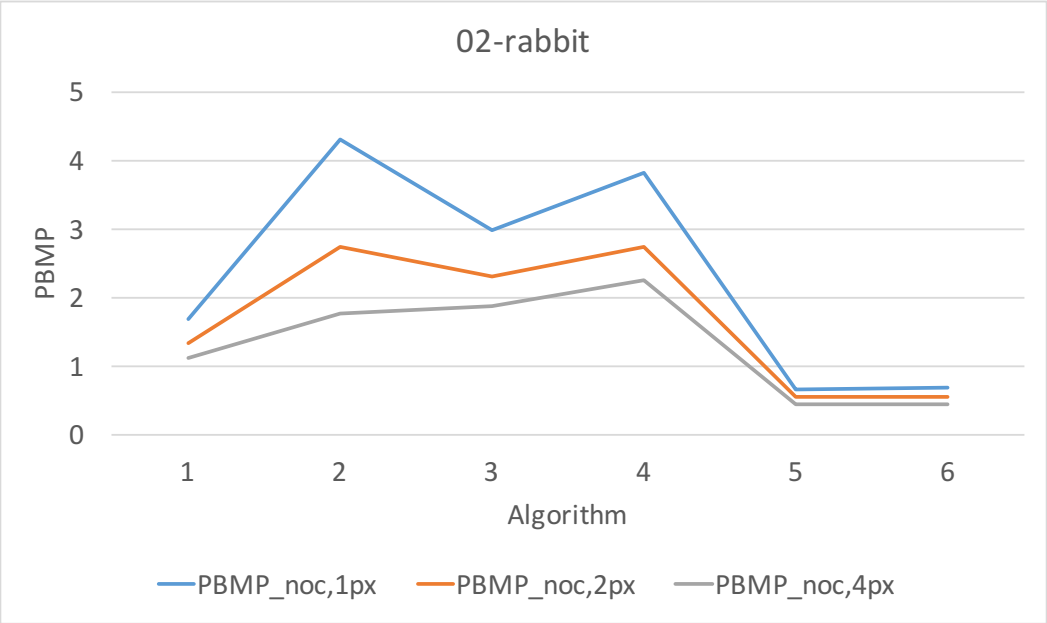


Figure 5.21: Performance of SVDDD rabbit scene

Both scenes, rabbit and apple performed pretty reasonable. In direct comparison with the Cambridge datasets, the results are better. The Cambridge scenes are unrealistic constructed in comparison to the Big Buck Bunny project as they contain a lot of repeating objects and arbitrary textures. Even the tunnel scenery contains many identical looking bricks. The literature states, that those textures are error-prone areas, this could be a reason for the better performing SVDDD. The coloring of the scenes in the Cambridge dataset is very even distributed whereas the SVDDD dataset contains different colors in each scene. Some scenes of the Cambridge dataset led to worse result than the others, especially the street scene performed best, which supports the assumption, that the result rely on the com-

position of the scenery. In the beginning, the assumption arose, that the fine hair textures may be problematic. This is not the case. The computed disparity map lack of accuracy near depth-discontinuity areas, but fine-grained structures were not problematic at all.

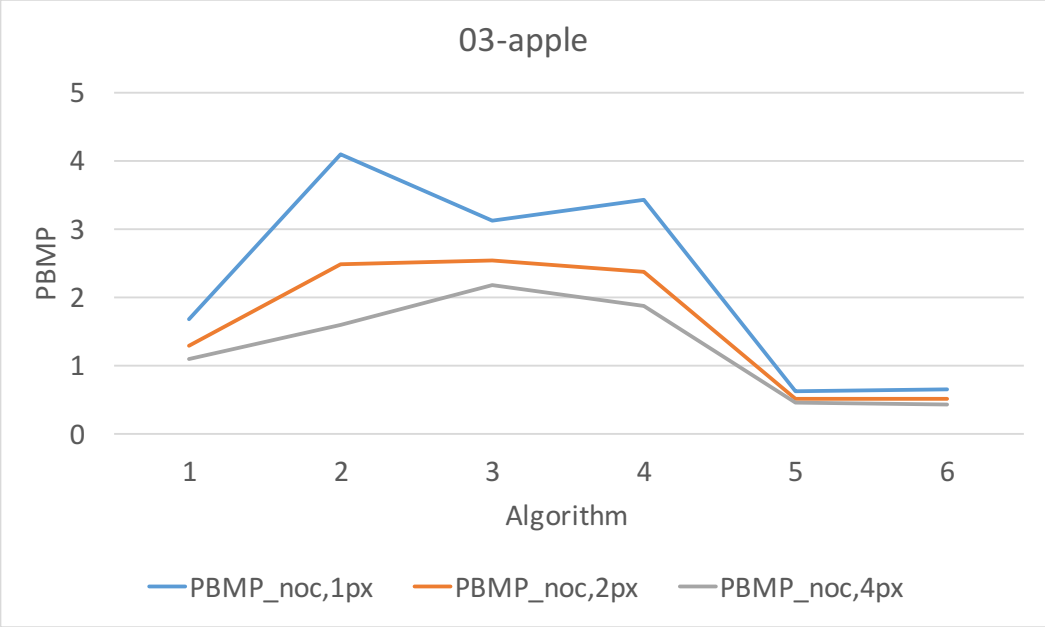


Figure 5.22: Performance of SVDDD apple scene

Focusing on the runtime, which is illustrated in Table 5.7, the fastest algorithm is again the OpenCV BM implementation. But the huge gap between (2) ELAS, (3) OpenCV BM and the others is noticeable. ELAS seems to be a pretty solid approach regarding runtime and overall performance of disparity estimation. The best results were achieved by the global methods, which pretty astonishing results of $\text{PBMP}_{noc,1px} < 1\%$. The runtime was very bad with nearly 10 minutes to compute one single disparity map. It has to be kept in mind, that the computation depends heavily on the underlying computing engine (CPU or GPU) and also the implementation. Also, in difference to the Cambridge dataset, the SVDDD dataset led to endless iterations in the Middlebury library. The Middlebury library is kept to a maximum of 500 iterations, which is not adjustable. The SVDDD dataset always needed the whole iterations, which may be the reason for such a long runtime. A reason for that can be the high resolution of the dataset.

	1	2	3	4	5	6
02-rabbit-neg	6550 ms	817 ms	1923 ms	11796 ms	300939 ms	544833 ms
02-rabbit	4402 ms	465 ms	1000 ms	20141 ms	420835 ms	617998 ms
03-apple	4541 ms	467 ms	984 ms	16063 ms	384221 ms	575137 ms
∅	5164 ms	583 ms	1302 ms	16000 ms	368665 ms	579323 ms

Table 5.7: Result table for runtime of SVDDD

5.5 Discussion

In this section, the result seen before are summarized and discussed. The general performance of the integrated algorithms in comparison with the simple naive stereo matcher (SNSM) is interesting. The outcome of the comparison is, that in general local methods are faster than global methods but the quality benefit of the disparity map may not be high. Although, this depends heavily on the use-case, only one global method, the (9) MRF - BP BPM was really superior. SNSM is on a level playing field with other global methods, and even outperforms some, like (3) MRF ICM and (8) MRF - BP BPS. Taking the spatiotemporal context into account yields overall to better results, but still lacks of respecting the motion of objects. Thus, the result may be a bit random regarding this particular case, as mentioned beforehand in Chapter 4. The masks are granting another perspective on the performance of disparity algorithms, but only the depth-discontinuity as well as the non-occluded pixel mask seem to give a real overall benefit. The saliency mask seems to bit more random as most of the salient regions are not marked correctly whereupon the room for interpretation when it comes to saliency has to be kept in mind.

The results regarding Gaussian noise differ from the results, which **?** [**?**] achieved. Here, a much lower σ^2 led to defective results. As every pixel gets changed according to the normal distribution and additionally, for each image a different noise matrix was calculated to avoid pattern matching, the stereo image gets disturbed in an unnatural way. **?** added additive Gaussian noise to simulate a more natural, real scenery. Gaussian noise is a part of sensor noise, but a small one. To simulate such real scenery, a more realistic model should be created. For the purpose of simulating an image, captured via CCD sensor, camera noise models exist [**?**]. Such complex noise models are difficult to create, but maybe lead to much more feasible results. However, it could be shown that a small amount of normal distributed noise lead to false disparity information. The small amount of informative value is may be at random computed.

With testing the interference of video compression, a novel approach was shown.

The outcome of the approach is, that even a small amount of video compression artifacts lead to a lot of mismatched pixels, although the artifacts are not clearly visible. As two parameters are correlating directly and the change of the CRF from 14 to 28 did not change the overall result, it may be the case, that disparity algorithms struggle with a small amount of noise until a certain level. The jump from about 10% of bad matching pixels to nearly 60% at the first tested step is huge. In case of disparity remapping, as discussed in Chapter 3, it does not make any sense to count on compressed video material. The resulting disparity maps seem to be too much disturbed regarding the overall performance.

Taking the runtime into account, the information taken out of the literature were confirmed. Global methods tend to run pretty long whereas local methods are more for real-time applications. Here, ELAS established itself as a surprise candidate. The overall performance combined with the runtime is good. The results towards object borders are accurate. Depending on the image resolution, global methods struggle with high-resolution images whereas ELAS performed pretty well. Global methods tend to run pretty long, nearly 1000 times the runtime of local methods.

The last part of the evaluation was to benchmark the SVDDD dataset of the department of Praktische Informatik IV[9]. After initial problems with the range of disparity values as well as some ground-truth disparity maps contained unfeasible values, two final scenes were constructed and could be evaluated. Both scenes performed very good along all algorithms. Worth mentioning is also the achieved performance in comparison with the Cambridge dataset. SVDDD led to notable better results. This may come from the composition of both scenes, as much more different textures and colors are included. Focusing on the runtime of the SVDDD dataset and feasibility of applying disparity algorithms on it, the computational limitations became noticeable. The belief propagations as well as the SNSM implementation were in need of too much memory. This may come from bad implementation as well as inappropriate parameters. The parameters of the Middlebury MRF library were not adjustable. Thus, this could not be investigated further more. None the less, the other algorithms drew the bigger picture with the main outcome, that the OpenCV SGBM implementation as well as ELAS performed best.

[9] `http://ls.fmi.uni-mannheim.de/de/pi4/`

6 Conclusion

The work described in this thesis has been concerned with the comparison of disparity algorithms for stereoscopic videos. For this purpose, evaluation methods were introduced and integrated. As stereoscopic datasets focusing on videos are rarely spread, the ones, which were used in the evaluation, were also introduced. Additionally, a simple stereo matcher focusing on videos was implemented described. The following section recaps this in more detail and furthermore a outlook and future work are given.

6.1 Thesis summary

The thesis draws a bow from computer vision basics to the foundations of disparity algorithms. For a better understanding of stereoscopy, the human visual perception was introduced. For formulating the stereo correspondence problem, how disparity algorithms work and which constraints exist, epipolar geometry was recapped. Furthermore, a simplified stereo matcher and common pitfalls were discussed.

Concerning disparity algorithms, the diversity, the taxonomy and specialities were presented. To draw the bigger picture, the estimation of sub-pixel accuracy and Optical flow were described. During the initial literature research, a lot of related work was found. As a feasible selection, the well-known semi-global matcher from **?** [**?**] and ELAS from **?** [**?**] were described in more detail. Furthermore, a general overview on solving optimization problems for the description of the Middlebury MRF library was incorporated. As the thesis has been concerned with the comparison of disparity algorithms, targeting stereoscopic videos, spatiotemporal consistency as a novel approach to take the correlation of consecutive frames into account was discussed. Additionally, remapping of the disparity range in stereoscopic videos was presented.

The implementation of the thesis consists of several parts. First, an evaluation suite composed of four components was presented. An image diminisher was implemented to alter sequences by adding Gaussian noise to fake a more real scenery or artifacts of video compression to simulate real use-cases. An algorithm executer

with a generic disparity interface for the integration and prospective expandability was built. The OpenCV implementation of disparity algorithms, the Middlebury MRF library and ELAS, the efficient large-scale stereo matcher, were integrated. A mask creator was presented to create simple bitmasks for focusing on interesting areas during the evaluation of such algorithms. A non-occluded mask, a mask for focusing on textureless regions, a depth-discontinuity mask and a mask for determining salient regions were established. Finally, a disparity evaluator with traditional metrics for comparing video processing algorithms were implemented. Second, for visualizing the outcome of the evaluation suite, a web result viewer was presented. Third, as a lot of related work regarding spatiotemporal consistency was read and processed, a simple naive stereo matcher which respects the spatiotemporal context was implemented and evaluated. Worth mentioning is reader for the PFM file format, which was implemented utilizing the OpenCV file format interface. A few scripts for the evaluation as well as a docker image for the distributive computation of disparity maps were created.

Finally, datasets were examined and chosen. A set of defined quality metrics for assessment of the algorithms together with their runtime were introduced. The whole measurement process is depict and the results are discussed. Additionally, the dataset of the department of Praktische Informatik IV[1] was iteratively improved and in the end evaluated.

As concluding result, it could be proven that, as known from the literature, local methods are fast and global methods are slow. Both lack of accuracy near depth-discontinuity areas as well as on arbitrary textures. The runtime differences between local and global methods is immense. A novel approach towards disparity estimation was presented with ELAS. It yielded to good results overall with a decent runtime. Focusing on the SVDDD dataset of the chair, which provides high-resolution stereoscopic videos, ELAS performed outstanding in comparison with other local methods. The dataset performed pretty good in comparison with the Cambridge dataset. Best results over all categories were achieved with global methods, but with the drawback of a huge runtime. It has to be kept in mind, the the drawback of the huge runtime depends on the application. The simple naive stereo matcher (SNSM) led to good results in some sequences and with some diminishing effects. Overall, the presented skeleton for further research and development seemed to be a bit random in some cases. As the spatiotemporal context of the SNSM takes image as a whole into account, object motion may be the reason for some random results. Additionally, it could be shown, that Gaussian noise and video compression lead to defective disparity estimations, even

[1] `http://ls.fmi.uni-mannheim.de/de/pi4/`

with a small amount of disturbing effects. To sum it up, the thesis brought a good fundamental knowledge affecting numerous topics of disparity algorithms, plenty of implementational work was done and a versatile evaluation was presented.

6.2 Outlook and future work

With the web viewer, a tool for sharing benchmark results was presented. This tool could be enhanced even more to visualize the results with graphs. Another possibility is to let researcher submit their dataset and the whole eval-chain is then executed on a server with the results being shown afterwards. The simplest idea to enhance the evaluation engine is, to add more metrics and more algorithms.

With the simple stereo matcher, a skeleton for creating a four-dimensional disparity space image was presented. This could be a starting point for further research to optimize stereoscopic videos respecting the spatiotemporal context. Concluding, the SNSM performed reasonable but seemed to be a bit random in some cases. Thus, the next steps could be to try to focus on non-moving objects or to track the movement of objects via Optical flow to take only static parts into account.

Future work regarding enhancement of disparity algorithms in general are for instance to implement other matching cost calculation methods and evaluate those [**?**]. Another approach could be to focus more on how humans experience depth [**?**] and to combine those learnings with neuronal networks [**?**] which has not been done yet.

In general, the available datasets lack of high resolutions. Also multi-view datasets are not available with high-resolution images at all. Although some real-world ground-truth disparity maps exist, they normally lack of accuracy and tend to be available in only low resolutions [**?**]. One approach towards the direction of high resolution real-world stereoscopic videos was made by **?** [**?**]. They provide a error-bar how accurate the sensed disparity is. As this is currently the only dataset of that nature, there is still room for more.

As a more general outlook, algorithms like ELAS [**?**] are more demanded due to low runtime and the accuracy in high-resolution images for real-time applications like autonomous driving.

Declaration of Honour

Hiermit versichere ich, dass diese Arbeit von mir persönlich verfasst wurde und dass ich keinerlei fremde Hilfe in Anspruch genommen habe. Ebenso versichere ich, dass diese Arbeit oder Teile daraus weder von mir selbst noch von anderen als Leistungsnachweise andernorts eingereicht wurden. Wörtliche oder sinngemäße Übernahmen aus anderen Schriften und Veröffentlichungen in gedruckter oder elektronischer Form sind gekennzeichnet. Sämtliche Sekundärliteratur und sonstige Quellen sind nachgewiesen und in der Bibliographie aufgeführt. Das Gleiche gilt für graphische Darstellungen und Bilder sowie für alle Internet-Quellen. Ich bin ferner damit einverstanden, dass meine Arbeit zum Zwecke eines Plagiatsabgleichs in elektronischer Form anonymisiert versendet und gespeichert werden kann.

Mannheim, 3. Mai 2016

Ben John

Abtretungserklärung

Hinsichtlich meiner Abschlussarbeit mit dem Titel *„Comparison of Disparity Algorithms for Stereoscopic Video“* räume ich der Universität Mannheim/Lehrstuhl für Praktische Informatik IV, Prof. Dr. Wolfgang Effelsberg, umfassende, ausschließliche unbefristete und unbeschränkte Nutzungsrechte an den entstandenen Arbeitsergebnissen ein. Die Abtretung umfasst das Recht auf Nutzung der Arbeitsergebnisse in Forschung und Lehre, das Recht der Vervielfältigung, Verbreitung und Übersetzung sowie das Recht zur Bearbeitung und Änderung inklusive Nutzung der dabei entstehenden Ergebnisse, sowie das Recht zur Weiterübertragung auf Dritte.

Solange von mir erstellte Ergebnisse in der ursprünglichen oder in überarbeiteter Form verwendet werden, werde ich nach Maßgabe des Urheberrechts als Co-Autor namentlich genannt. Eine gewerbliche Nutzung ist von dieser Abtretung nicht mit umfasst.

Mannheim, 3. Mai 2016

Ben John