Master Thesis

Enhanced depth estimation for "freehand stereo" using PatchMatch Stereo

Dominik Wetzel

born on August 29, 1989 in Greiz

Course Computer Sciences

University of Applied Sciences Zwickau (UASZ)
Faculty of Physical Engineering/Computer Sciences
Professional group Computer Sciences

Supervising institution:	University of Technology Dresden Faculty of Computer Science Institute of Artificial Intelligence Computer Vision Lab Dresden (CVLD)
Supervisor at the institution:	Prof. Ph.D. Carsten Rother
Supervisor at the UASZ:	Prof. Dr. Frank Grimm
Date of delivery:	March 12, 2015

Autorenreferat

Algorithmen, welche Stereokorrespondenzen berechnen, nutzen normalerweise rektifizierte Bilder um Disparitäten zu berechnen. Falls keine Rektifizierung vorhanden oder diese inakkurat ist produzieren die meisten Stereo Algorithmen verzerrte Bilder. In dieser Arbeit wird eine Methode vorgestellt mit der solche Probleme für den PatchMatch Stereo Algorithmus reduziert werden.

Es werden vier Rektifizierungsalgorithmen vorgestellt und untersucht. Diese sind der Algorithmus von Hartley, der Algorithmus von Bouget, Quasi-Euclidean epipolar rectification (QER) und Polar Rektifizierung. Es wurde eine Evaluierung entwickelt und durchgeführt, um einen Algorithmus zu finden, welcher später als Rektifizierungsalgorithmus für PatchMatch Stereo dienen soll. Das Ergebnis ist der Algorithmus von Hartley zusammen mit der Fundamental Matrix, welche mittels QER ermittelt wurde.

Anschließend wird der PatchMatch Stereo Algorithmus vorgestellt und für freihand Stereo erweitert. Dazu wird der Suchraum vergrößert indem eine y-Disparität in das Datenmodell von PatchMatch Stereo integriert wird. Das erlaubt es Korrespondenzen zu finden, welche nicht auf einer horizontalen Linie liegen. Ein neuer Term wurde vorgestellt der Kosten je nach y-Disparitäten hinzufügt. Zusätzlich wurde eine Implementierung auf Basis der Open Computing Language (OpenCL) geschrieben, um die Geschwindigkeit des Algorithmus zu erhöhen.

Ein Evaluierung für die Änderungen wurde definiert und durchgeführt. Die Ergebnisse werden in der Arbeit präsentiert und analysiert.

Die Evaluierung zeigt, dass die Erweiterungen bessere Ergebnisse erzielen, besonders im unrektifizierten Fall. Es wird gezeigt dass der neu eingeführte Kosten Term nicht gut funktioniert und überarbeitet werden sollte. Außerdem konnte herausgefunden werden, dass die OpenCL Implementierung nicht die erwarteten Verbesserungen bringt.

Abstract

Stereo matching methods are usually relying on rectified images to calculate disparities. If the rectification is inaccurate or not present at all most stereo matching algorithms will produce distorted disparity maps. In this thesis an approach is described that overcomes this issue for the PatchMatch Stereo algorithm.

As groundwork for stereo matching four rectification procedures namely Bouget's algorithm, Hartley's algorithm, Quasi-Euclidean epipolar rectification (QER) and Polar rectification are introduced and examined. An evaluation approach for these algorithms is developed and used in order to find an algorithm that can be further used as rectification algorithm for a stereo matching algorithm, namely PatchMatch Stereo. As result the Hartley's algorithm in conjunction with the fundamental matrix calculated by QER is used.

Further the PatchMatch Stereo algorithm is introduced and improved to work with free-hand stereo. Therefore an extended search space is introduced. Basically it adds a disparity in y direction to the data model of PatchMatch Stereo. This allows to find matches that lie not on a horizontal scan-line. A new cost term is introduced to penalize too high y-disparities. Additionally an approach is defined to improve the speed of the algorithm by applying an Open Computing Language (OpenCL) implementation.

Further an evaluation procedure was defined and used with the proposed adjustments. The results are presented and analyzed.

After evaluating the improved PatchMatch Stereo in different scenarios, it turned out that the algorithm performs better with the adjustments especially in an unrectified case. Additionally it was found out that the proposed cost extension does not work well and should be thus revised. It was found out that the OpenCL implementation does not bring the expected improvements.

Information about the supervising institution

The University of Technology Dresden was founded in 1828 and has currently over 36.000 students with over 4.500 international students from 125 nations [Ode15].

The faculty of computer science is one of the fourteen faculties the university has. The Computer Vision Lab Dresden (CVLD) is located in that faculty and has currently seventeen members. "The mission of the Computer Vision Lab Dresden is to develop novel theoretical concepts which are practically relevant. [They] work on a broad range of application areas, from image matching, via semantic scene understanding, to Bio Imaging. On the theoretical side [they] focus on optimization and learning in probabilistic graphical models" [Com15].

Acknowledgements

Special thanks goes to Tim Gubner who read my thesis many times and gave valuable advice on how to write and extend different parts. I want to thank my supervisors Frank Grimm, Carsten Rother and Holger Heidrich who gave advice for the thesis and encouraged me in writing. Furthermore thanks goes to Andreas Dinter for proof reading. Special thanks goes to my wife Isabell Wetzel who encouraged me to write further and not give up on hard times. Additionally I want to thank my God who helped me a lot through the time of writing.

Table of Contents

List of used Abbreviations

AELD	average epipolar line distance
API	application programming interface
ARE	average rectification error
ASIFT	affine-scale-invariant feature transform
CoV	coefficient of variation
parConf	parameter configuration
CPU	central processing unit
CV	Computer Vision
CVLD	Computer Vision Lab Dresden
GPU	graphics processing unit
invPt	invalidated point
OpenCL	Open Computing Language
OpenCV	Open Source Computer Vision Library
PCL	Point Cloud Library
PR	point ratio
QER	Quasi-Euclidean epipolar rectification
RAM	random access memory
RANSAC	random sample consensus
UASZ	University of Applied Sciences Zwickau

List of Figures

List of Tables

1 Introduction

Three-dimensional (3D) models are useful in many areas. In movies they can be used to replace human characters or insert objects that do not need to be build in the real world. They are used as assets in video games. Architects can use them to visualize their building plans. Even in medicine 3D models can be used to visualize inner organs and so on. Nowadays such models can be printed with 3D printing devices which allow even more use cases. 3D models are created from 3D artists or can be extracted from image sequences which lays in the domain of Computer Vision (CV).

CV "is the transformation of data from a still or video camera into either a decision or a new representation" [BK08, p. 2]. Decision means that the computer can determine whether the data represents for example a car or a person. "A new representation might mean turning a color image into a grayscale [one] or removing camera motion from an image sequence" [BK08, p. 2] or might mean reconstructing 3D models out of an image sequence. Another possible definition is: CV is the area of "[d]eveloping computational models and algorithms to interpret digital images and visual data in order to understand the visual world we live in" [Rot14, p. 9].

CV has many fields of research, for example 3D scene understanding which is concerned with the recognition of objects within a scene, their position and pose and the relation towards each other [Wes14]. Another research area is 3D reconstruction which allows to create 3D models out of images or image sequences. This thesis is ranged in the latter area because it tackles the problem of estimating the depth values in an image pair of the same scene. Such an image pair is called a stereo image.

1.1 Goals and challanges

Several algorithms exists for estimating the depth of stereo images (for example [MSK14], [Jia+14], [Ric+10] and [BRR11]). This thesis focuses on one particular algorithm, namely PatchMatch Stereo [BRR11]. with the goal of refining its performance.

The algorithm shall be tested with images that are taken with one camera from different angles. These are called freehand stereo images. Thirty of such images shall be photographed and shall contain high depth changes, fine structures and few texture.

The relationship between those images (the epipolar geometry) is unknown and must be calculated. Therefore different algorithms for this shall be compared.

The PatchMatch Stereo algorithm must be understood and improved in a way that it works with stereo images where the epipolar geometry is not accurately estimated. The changes to the algorithm shall be evaluated afterwards.

1.2 Structure

This thesis is structured into six chapters whereas the first chapter (chapter 1) introduces the fundamental goals and purpose of this document. This is followed by an introduction of notations for the thesis as well as fundamentals about the proposed part of CV in chapter 2. Chapter 3 introduces related research that has been done in the area of stereo matching. In chapter 4 the preprocessing step for stereo algorithms is tackled. This preprocessing is called rectification. Therefore different approaches for rectification are explained and evaluated. Chapter 5 describes the stereo matching algorithm PatchMatch Stereo and proposes performance improvements for it. This is followed by an evaluation of these improvements. The last chapter (chapter 6) summarizes the results, gives a conclusion and sketches different ideas to further improve the algorithm.

2 Fundamentals

This chapter will introduce naming conventions and fundamental concepts.

In order to make a 3D scene out of two images different steps are performed. First two pictures from the same scene are made with a camera which is modeled by the pinhole model (section 2.2.1). These images define an epipolar geometry (section 2.2.2). Given some point correspondences it is possible to calculate a matrix that contains the relation between the two images, the so-called fundamental matrix (section 2.2.3). Using the fundamental matrix it is possible to rectify (section 2.3) the images as rectified images allow easier disparity calculation. Such disparities can be calculated using the PatchMatch Stereo algorithm (chapter 5). Afterwards these disparities can be used to extract the depth information which can be used to reconstruct a 3D model (known as 3D reconstruction) (section 2.4). The whole procedure is visualized in figure 2.1.

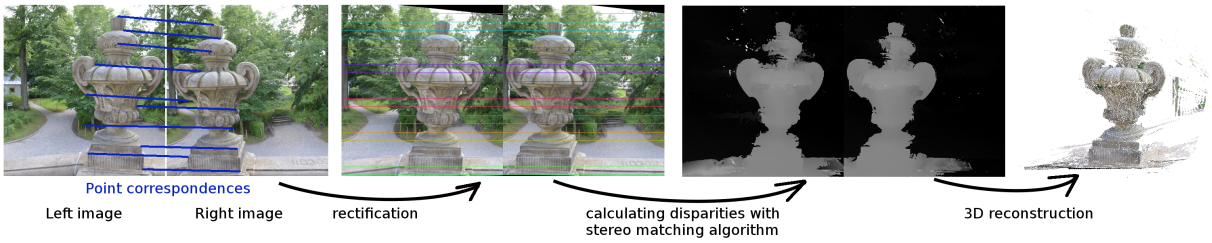


Figure 2.1: The procedure from two images to a 3D reconstruction.

2.1 Notations

In order to not get confused by potentially differing notations of other publications this section will introduce notations that are used throughout the thesis.

Points In the frame of this thesis 3D-points are referred to as points in the real world and are named with upper case letters (e.g. Q). 2D-points are 3D-points that are projected onto an image plane. These are named with lower case letters (e.g. q). Points using inhomogeneous coordinates are marked with a tilde (e.g. $\tilde{q}$). Subscript on points refer to their respective element (e.g q_x = x-coordinate of q)

Stereo image A stereo image is a pair of images, that show the same scene from slightly different views. It consists of two roughly horizontally offset images.

Point correspondence If two 2D-points in different images refer to the same 3D-point these two 2D-points are called a point correspondence.

Freehand stereo A way of taking stereo images is to take two pictures from different angles with the same camera. This is called freehand stereo. Other approaches are also possible but the focus of this thesis is on freehand stereo.

Apostrophe Stereo images consist of two images, but calculations (or relations) are mostly done on one image, the *reference image*. In order to distinguish matrices, points etc. from the reference image with the matrices, points etc. of the other image an apostrophe is used to mark the matrices, points etc. of the other image. For example x refers to a point on the reference image and x' refers to a point on the other image.

Vector cross product The cross product of two vectors $(a \times b)$ in $\mathbb{R}^3$ can be written as a matrix multiplication of a skew-symmetric matrix $[a]_\times$ and the vector b and is called vector cross product. This is shown in equation (2.1). [cf. Tre98]

$$a \times b = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = [a]_\times \cdot b \tag{2.1}$$

2.2 Stereo image geometry

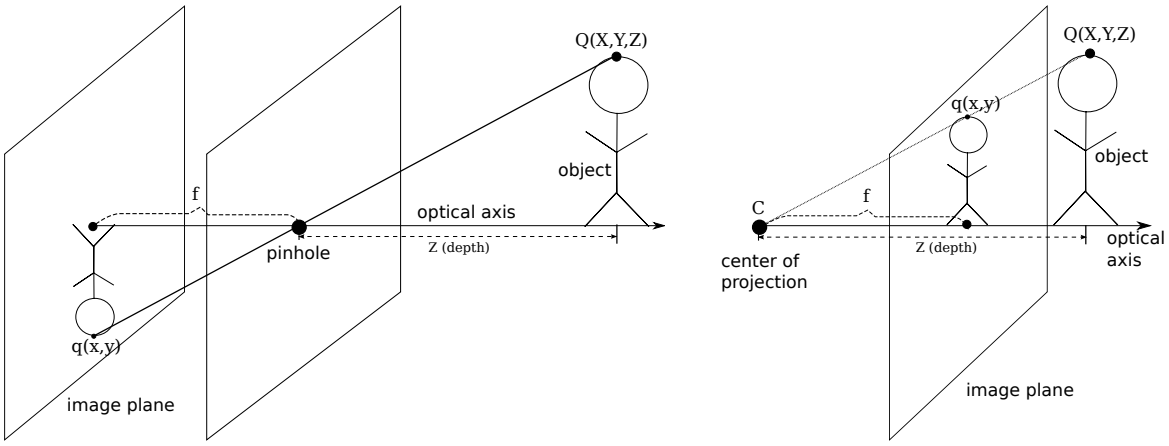
This section will describe the relationship between the two images of a stereo image and the therefore used camera model.

2.2.1 Camera model

A camera mostly consists of a housing, a sensor and a lens. The photosensitive sensor, which is also referred to as the *image plane*, senses the light that falls upon it. The housing prevents stray light from hitting the sensor, while the lens lets some light pass onto it [cf. Daw14, p. 9].

A lens can be rather complex and is hard to model correctly. In order to simplify and abstract the real camera into a model the pinhole camera model[1] is used. In the pinhole model the lens is replaced by a pinhole. This model is extended with some distortions which are introduced through various other parts of the camera e.g. lenses [cf. Daw14, p. 9].

In this model light rays go from points of an object through the pinhole and are projected onto the image plane upside down. This is visualized in figure 2.2a. The size of the projection is only dependent on the distance from the pinhole to the image plane, the *focal length*(f) [cf. BK08, p. 371].



(a) The original pinhole model **(b)** The arranged pinhole model

Figure 2.2: The pinhole camera model

In order to simplify the model (and to avoid projections that are upside down) the image plane and the pinhole are swapped (see figure 2.2b). Furthermore the pinhole is reinterpreted as the *center of projection* (C). In this case the projection q is created by the intersection of the light ray from point Q to C with the image plane. The intersection of the optical axis with the image plane is called *principle point*. [cf. BK08, 371f.]

Intrinsics matrix

The projected points can be calculated as in equation (2.2) [cf. BK08, p. 374]. This equation uses homogeneous coordinates in order to simplify the calculation into a matrix multiplication.

$$q = K \cdot \tilde{Q}, \text{ where } q = \begin{bmatrix} x \\ y \\ w \end{bmatrix}, \; K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \; \tilde{Q} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.2}$$

[1] "The pinhole camera model goes back at least 987 years to al-Hytham [1021] and is the classic way of introducing the geometric aspects of vision" [BK08, p. 370].

In this equation is q the 2D-point and $\tilde{Q}$ the corresponding 3D-point. The focal length is f. The equation states two different focal lengths, due to low-cost sensor-elements are typically rectangular instead of square sized [cf. BK08, p. 373]. For the sake of simplicity only squared sized sensor-element were considered. The translation from the principle point to the upper left corner of the image plane is described by c_x and c_y. This is used because images are usually described with the upper left corner as the origin. These values are in most cases set to half of the image size in the corresponding direction. Other values occur if the cameras sensor is not exactly centered to the optical axis. In special occasions skew can occur, described in [HZ04, p. 164]. The skew factor is referred as s and is set to zero among this thesis. The matrix K contains the intrinsic parameters of a camera and is therefore called *camera intrinsics matrix*. With this transformation an inhomogeneous 3D-point can be projected to a 2D-point in homogeneous coordinates on the image plane.

Extrinsic matrix

"The camera's *extrinsic matrix* describes the camera's location in the world, and what direction it's pointing" [Sim12]. Therefore a matrix is used that relates the world coordinates to the the camera. The matrix consist of a rotation R, which is shown in equation (2.3), and a translation $t = (t_x, t_y, t_z)$.

$$R = R_z(\theta) \cdot R_y(\varphi) \cdot R_x(\psi), \text{ where} \tag{2.3}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, R_y(\varphi) = \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{bmatrix}, R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix}$$

Both are combined to one 3×4 matrix shown in equation (2.4).

$$M = [R|t] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_z \end{array}\right] \tag{2.4}$$

"The vector t can be interpreted as the position of the world origin in camera coordinates, and the columns of R represent [...] the directions of the world-axis in camera coordinates" [Sim12]. With this it is possible to position the world coordinate system within the camera. The choice of the world coordinate system is arbitrary, this means that for every 3D world coordinate system there exists an unique extrinsic matrix. This is especially useful for relating multiple cameras to each other.

Projection matrix

Combining the intrinsics and extrinsic matrix via matrix multiplication results in the 3×4 projection matrix P (see equation (2.5) [cf. HZ04, 154ff.]). It transforms 3D world coordinates into 2D camera coordinates.

$$q = P \cdot Q, \text{ where } P = K \cdot [R|t] \tag{2.5}$$

Distortions and calibration

It is possible that some aspects of a camera can introduce distortions, as stated in section 2.2.1. There are two main distortions, that can be properly modeled and removed. These are "*/r/adial* distortions [which] arise as a result of the shape of [the] lens [and] *tangential distortions* [which] arise from the assembly process of the camera as a whole" [BK08, p. 375]. These two distortions add some new parameters to the model, which allows an algorithm to eliminate these from the image.

A camera is called *calibrated* if its K matrix and its tangential and radial distortion parameters are known. This can be achieved through a calibration process, which is described in [BK08, 381ff.]. For more information about the camera model see [chapter 11 in BK08, 370ff.].

2.2.2 Epipolar geometry

Consider two cameras, or one camera that takes two shots, which is mathematically equivalent, that photograph a scene from different angles, like in figure 2.3. The camera centers C and C' are related by a rotation and a translation R, t. The line connecting C and C' is called the *baseline* and the intersections of the baseline with the image planes are called *epipoles* (e and e'). Lets pick a 3D-point Q of the scene. If the point is not occluded in one of the images the 3D-point is projected onto both image planes (as q and q'). The three 3D-points Q, C and C' span a plane, the *epipolar plane*. The intersection of that plane with the image planes are called *epipolar lines*.

Figure 2.3 also shows that every point on the ray from C to Q is projected to the same location (q) on I, but in I' these points are wandering on the epipolar line. This means a point in one image defines an epipolar line in the other image. All 3D-points that are not on this epipolar plane do span new planes and therefore new epipolar lines which are all intersect at the epipoles. [cf. Sch05, 68f.]

The epipolar geometry reduces the search space for one point correspondence to a line instead of the whole image.

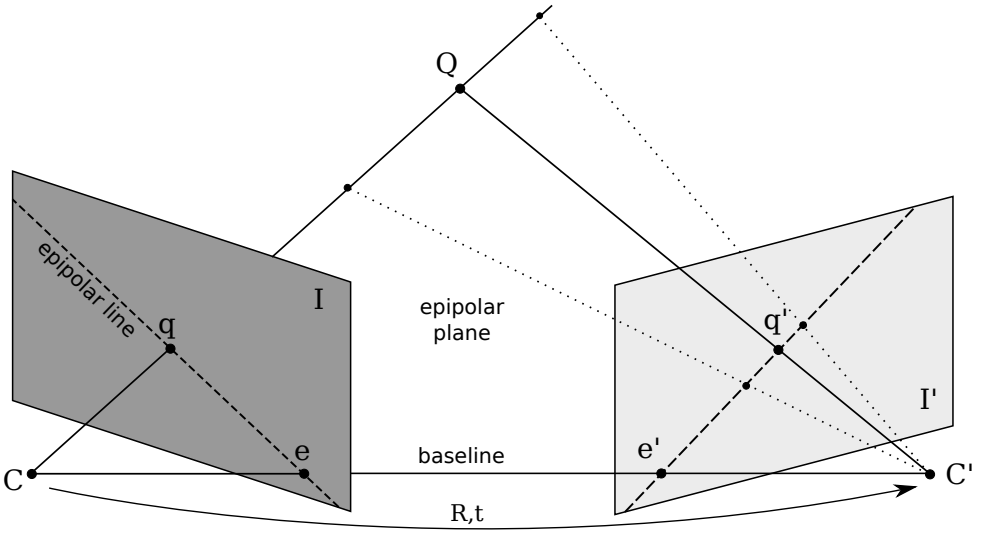


Figure 2.3: The epipolar geometry. I and I' are the respective image planes.

2.2.3 Essential and fundamental matrix

This section introduces two new matrices that are used to relate the two cameras towards each other, the *essential matrix* (E) and the *fundamental matrix* (F).

Essential matrix

The essential matrix "contains information about the translation and rotation that relate the two cameras in [world coordinates]" [BK08, p. 421]. The matrix can be defined by the cross product of t with R. This is shown in equation (2.6) [cf. HZ04, p. 257]. Note that this equation uses the matrix representation of the cross product.

$$E = [t]_{\times} \cdot R \tag{2.6}$$

R and t "describe the location of the second camera relative to the first in [world] coordinates." [BK08, p. 421]

Usually (and especially in the freehand stereo case) the rotation and translation between the cameras is not given and cannot be measured easily. Hence this matrix is mostly used to retrieve these information from the stereo image itself. It is possible to calculate this matrix via the fundamental matrix if the cameras are calibrated with equation (2.7) [cf. HZ04, p. 257].

$$E = K'^{T} \cdot F \cdot K \tag{2.7}$$

Fundamental matrix

The relation of two cameras in pixel coordinates is described by the fundamental matrix. Therefore it contains the intrinsics information of the two cameras and can be defined by equation (2.8) [cf. HZ04, p. 244].

$$F = K'^{-T} \cdot E \cdot K^{-1} \tag{2.8}$$

This matrix can be calculated via at least seven point correspondences. It can be computed with uncalibrated cameras. For more information about the calculation of the fundamental matrix see [chapter 11 in HZ04, 279ff.]

Properties This matrix has some important properties which are given in the following. This overview is taken from [table 9.1 in HZ04, p. 246]

- F is a rank 2 homogeneous matrix with 7 degrees of freedom where rank is the amount of "linearly independent rows or columns of the matrix" [Wei02] .
- **Point correspondence:** if x and x' are corresponding image points, then $x'^T F x = 0$
- **Epipolar lines:**
  - $l' = Fx$ is the epipolar line corresponding to x.
  - $l = F^T x'$ is the epipolar line corresponding to x'.
- **Epipoles:**
  - $Fe = 0$.
  - $F^T e' = 0$.

As can be seen it is possible to directly retrieve the corresponding epipolar line for a given point with the fundamental matrix.

2.3 Rectification

The image planes can be reprojected onto one plane, in a way that the corresponding epipolar lines are horizontally[2] aligned (also called a frontal parallel configuration) [cf. BK08, p. 430]. This method is called rectification and is done on image coordinates.

[2]It is also possible to align the epipolar lines vertically but for better understanding and reading only the horizontal case is used.

Therefore a transformation is performed on both pictures, such that the images are in a frontal parallel configuration afterwards. The images are then called *rectified*. A mapping can be defined that maps each point on the original image towards the rectified image which is visualized in figure 2.4.

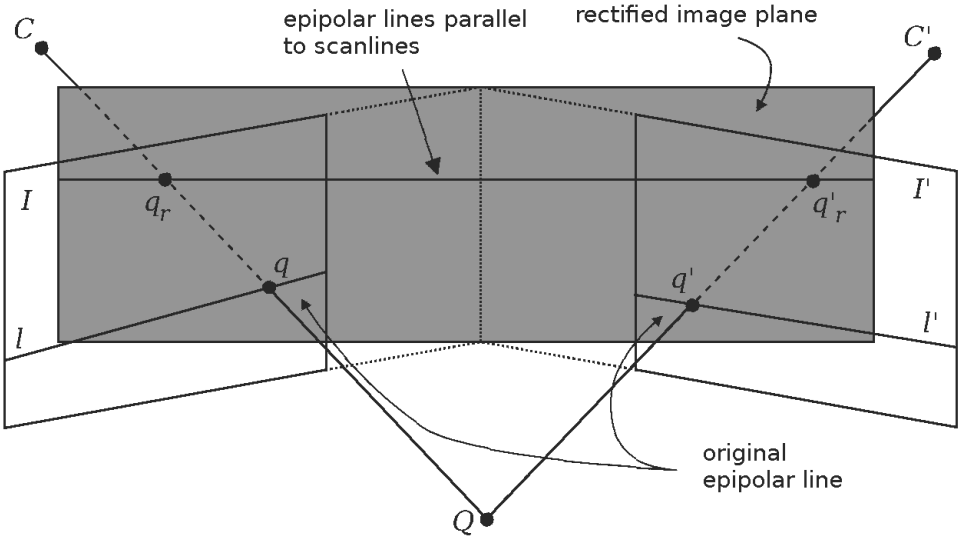


Figure 2.4: Original and rectified views

Source: [cf. figure 7.1 in Sch05, p. 106]

In this figure q_r and q'_r are the rectified points of their corresponding points q and q'. Also note that the epipoles are mapped to infinity. With this done the point correspondences lay on scan lines instead of slanted lines and the fundamental matrix has a special form shown in equation (2.9) [HZ04, p. 249].

$$F_{rect} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.9}$$

There are many different algorithms for calculating the right rectification transformations, four of them will be discussed in section 4.1.

Homography A "projective mapping from one plane to another" [BK08, p. 384] is called homography (H). Some rectification transformations can be described via a 3×3 homography for each image.

2.4 Depth estimation

Depth is estimated over two steps, first *disparity maps* are computed and afterwards these disparities are used for the triangulation procedure. The quality of the estimated

depth depends (besides other factors) on the known information about the camera. If the camera is calibrated it is possible to get the depth up to a scale factor. With some additional information (e.g. the size of one meter in the image) the depth can even be calculated absolute which allows precise metric measurements. Otherwise the depth can be estimated up to a projective transformation [cf. HZ04, 264f.].

2.4.1 Disparity

Disparities "are the differences in x-coordinates on the image planes of the same feature viewed in the left and right cameras: $[x - x']$" [BK08, p. 415]. These disparities are calculated for all corresponding points in the images, resulting in a disparity map. Figure 2.5 shows such a disparity map.

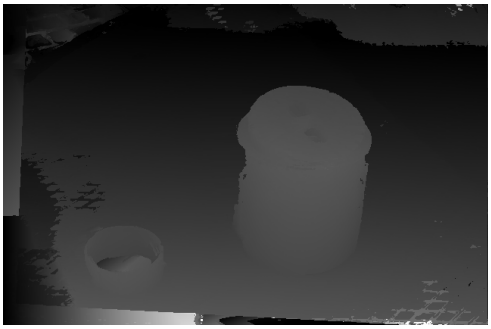


Figure 2.5: A disparity map of the sewerCover test set.

2.4.2 Triangulation

The depth Z can be calculated from the focal length f, the translation t and the disparity d by similar triangles with equation (2.10) [cf. BK08, p. 417].

$$Z = \frac{f \cdot t}{d} \tag{2.10}$$

This is visualized in figure 2.6. In the case when f and t are not present these parameters can be estimated. Equation (2.10) shows that the depth is inversely proportional to the disparity.

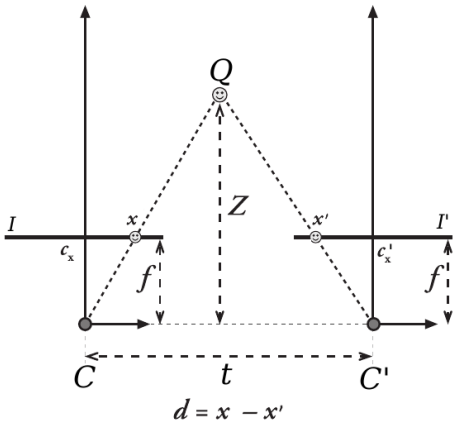


Figure 2.6: Triangulation process

Source: [cf. figure 12-4 in BK08, p. 416]

As a consequence nearby objects have high disparities while far away objects have low ones. Furthermore this implies that a small disparity change on far away objects has an huge impact on its calculated depth, whereas with nearby objects this is only a small change. Due to this depth information of good quality can only be achieved with objects relatively near to the camera [cf. BK08, p. 417]. For more information about triangulation and 3D reconstruction see [BK08, 415ff.] and [chapter 10 in HZ04, 262ff.]

3 Related work

This chapter will show related research in the area of stereo matching algorithms. Stereo matching algorithms are usually defined as optimization problems. Based on the method of solving the optimization problem they can be divided into two categories: global and local methods. In global methods "[a]n energy [...] function [(also called cost function)] is given at first [and] then an optimization algorithm is used to obtain the disparity value which makes the energy function minimum" [Zha+12]. Such an energy function is usually divided into a data term, which represents the data, and a smoothness term which "penalize[s] disparity solutions that are not smooth" [cf. Cap12]. Global methods are usually more accurate but also more time-consuming.

As opposed to global algorithms, local algorithms calculate the disparity of each pixel depending on a window of neighbor pixels (called support window). This is usually square sized. These algorithms are usually faster but less accurate than global ones especially on occluded regions [cf. Zha+12]. In recent years this problem has been overcome by an adaptive support weight strategy [YK06] as seen for example in [Hos+09], [Ric+10] and [Rhe+11]. PatchMatch Stereo itself is a local algorithm, but as stated in [BRR11] it can also be used as a global one. In the following a selection of local methods which use different techniques are introduced.

Cost filtering [Jia+14] proposes a cost function that contains the "truncated absolute difference of color and gradients [in x and y direction]" [Jia+14] and additionally a value which is derived from the image in the Gaussian color model [Geu+01] space. Afterwards they aggregate the costs for each pixel at each disparity level, filter them using a symmetric guided filter [Rhe+11] and finally "[select] the disparity label with the lowest cost" [Jia+14]. This results in an initial disparity map which is further refined with an initial post-processing and a secondary refinement scheme they call "Remaining Artifacts Detection and Refinement" (RADAR) [cf. Jia+14]. According to [Jia+14] the RADAR detects small holes in the disparity map as well as inconsistent regions throughout the images. While this approach works well in the Middlebury benchmark [SS02] its cost function has the downside of only allowing integer valued disparities. PatchMatch Stereo overcomes this constraint by over-parameterizing the disparities with 3D planes.

Segmentation Many other recent algorithms also use the idea of 3D planes in order to model the data. In [MSK14] for example an initial disparity map is segmented using a mean-shift segmentation [CM02] first. Afterwards a set of planes that have consistent disparity values in both images is calculated for each segment. These planes are pixel-wise labeled onto the pixels with an aggregated cost. Afterwards those planes are filtered and reduced to only dominant planes. The planes are relabeled and the resulting disparity map is used as initial disparity map in the next iteration of the algorithm [cf. MSK14]. While the algorithm can be used as a standalone stereo matching algorithm it is more useful in refining disparity maps [cf. MSK14]. In the end only one plane remains for each segment [cf. MSK14]. This might be problematic for fine grained structures and huge disparity jumps on small areas.

Related to PatchMatch Stereo [Hei+13] uses the PatchMatch Stereo data term and extends it by a smoothness term which uses the Huber norm [Hub73] that contains the disparity values and normal vectors of the planes. Furthermore the energy optimization problem is transformed into two subproblems. One is solved with the "use of a primal-dual formulation of the Huber-ROF model as described" [Hei+13] in [CP11]. The other is solved by a random sampling of information from the previous iteration [cf. Hei+13]. Their experiments show that this algorithm performs superior to PatchMatch Stereo but it also requires proper rectified images [cf. Hei+13].

The approach stated in [TMN14] also uses the PatchMatch Stereo data term for a global method but in this thesis only the local approach is used.

Misaligned images In [RTV13] a method is proposed that allows vertical offset (also called y-disparity). A y-disparity is added into the cost function of the data model. For each horizontal offset (also called x-disparity) the vertical offset which produces the lowest cost is stored. This results in a lookup-table. This allows the use of standard optimization methods. In [RTV13] semi-global matching [Hir08] is used as optimization algorithm, but they state that other algorithms are also possible [cf. RTV13]. The downside of this approach is that continues disparity values cannot be used due to it creating a lookup table for disparities which makes it not suited for PatchMatch Stereo.

Conclusion The stated algorithms base on the assumption that the images are correctly rectified (besides [RTV13]). This may not be the case in "freehand stereo".

4 Rectification

Calculating the rectification transformation is a well known problem. Hence many different algorithms exist that try to solve that problem. This chapter shall give an overview about different techniques and will also describe an evaluation approach as well as the evaluation of the proposed algorithms.

4.1 Algorithms

This section will give a brief overview over four rectification algorithms and the performed calculations in them. A complete understanding of these algorithms is not necessary, but can be obtained by studying the respective literature.

Rectification can be computed using two kinds of algorithms: One kind needs calibrated cameras and the others do not but most need some point correspondences. In the following the Bouget's algorithm as a representation of an algorithm that needs calibrated cameras is described. This is followed by a description of the Hartley's algorithm, the Quasi-Euclidean epipolar rectification (QER) and the polar rectification as representations for algorithms that use uncalibrated cameras.

4.1.1 Bouget's algorithm

This algorithm uses calibrated cameras, meaning that, besides the point correspondences, it needs the intrinsics matrices of the cameras as well as the rotation and translation between them.

The algorithm is described in [BK08, 433ff.][3]. It "attempts to minimize the amount of change reprojection produces for each of the two images [...] while maximizing common viewing area" [BK08, p. 433].

In order to arrange the cameras in a coplanar alignment the rotation matrix R (see equation (2.3)) is split in half into a left and right rotation (r and r'). Further the rows

[3] Jean-Yves Bouget refined and completed a method first introduced in [Tsa87]. He never published his algorithm besides his Matlab toolbox described in [Bou13]. [cf. BK08, p. 431]

need to be aligned to. This is done by mapping the epipole of the reference image to infinity. Hence a rotation matrix (R_{rect}) is created which can be seen in equation (4.1) [cf. BK08, p. 434].

$$R_{rect} = \begin{bmatrix} (e_1)^T \\ (e_2)^T \\ (e_3)^T \end{bmatrix}, \text{ where } e_1 = \frac{t}{||t||},\ e_2 = \frac{[-t_y\ t_x\ 0]^T}{\sqrt{t_x^2 + t_y^2}} \text{ and } e_3 = e_1 \times e_2 \tag{4.1}$$

The first row of R_{rect} is the direction of the image's epipole e_1. By assuming the image origin as the principle point it is possible to determine the direction of e_1 directly from the translation vector t. The vector e_2 must be orthogonal to e_1 and is otherwise unconstrained. Therefore the direction of the optical axis is used. It can also be calculated with t. Last but not least the vector e_3 is the cross product between e_1 and e_2.

Multiplying the rotations results in a rectification transformation (R_r and R'_r) shown in equation (4.2) [cf. BK08, p. 434].

$$\begin{aligned} R_r &= R_{rect} \cdot r \\ R'_r &= R_{rect} \cdot r' \end{aligned} \tag{4.2}$$

Afterwards the new rectified projection matrices are calculated. With this it is possible to calculate the rectification mapping.

4.1.2 Hartley's algorithm

Another approach to the rectification problem is "to find homographies that map the epipoles to infinity while minimizing the computed disparities between the two stereo images" [BK08, p. 431]. This is known as the Hartley's algorithm. It needs the fundamental matrix F and some point correspondences as input.

The right and the left epipoles are determined via F and its properties $F \cdot e = 0$ and $(e')^T \cdot F = 0$. Afterwards the homography H' which maps the the epipole e' to infinity is calculated. This is done by

1. a translation T which maps an arbitrary point (e.g. the image center) to the image origin (to avoid distortions around this point),
2. a rotation R which rotates the epipole to $e_{toInf} = [f, 0, 1]^T$ and
3. a transformation G which maps the point e_{toInf} to infinity (e.g. to $[1, 0, 0]^T$).

Equation (4.3) [cf. BK08, p. 432] illustrates this procedure.

$$H' = G \cdot R \cdot T, \text{ where } G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/f & 0 & 1 \end{bmatrix} \tag{4.3}$$

In the next step the matching homography H is calculated. H must map the epipole e to infinity and also row align the images. In order to map e to infinity equation (4.3) is used. Further H is chosen such that it minimizes the total disparity of the point correspondences which can be seen in equation (4.4) [cf. Har99]. This achieves row alignment.

$$\sum_i (H \cdot p_i - H' \cdot p_i')^2 \tag{4.4}$$

These two homographies rectify the images. More information about this algorithm can be found in [BK08, 431ff.] and in the original paper [Har99].

4.1.3 Quasi-Euclidean epipolar rectification

[FI11] propose a third approach for the rectification problem. The main idea of their algorithm is to "seek the collineations that make the original points [(the given point correspondences)] satisfy the epipolar geometry of a rectified image pair" [FI11].

This is known as the QER algorithm. Note that it only needs the point correspondences as input. It bases on different assumptions: It assumes that

1. the intrinsics matrix for both images is the same,
2. the aspect ratio of the focal lengths is one,
3. there is no skew and
4. the principle point is in the middle of the image.

Hence the intrinsics matrix can be parameterized with a single parameter for the focal length. Furthermore the rectified intrinsics matrix (K_{rect}) is set to the original (K) modulo a shift of the principle point (which might become necessary for centering the rectified images into a custom frame [cf. FI11]). The rectification homographies can be calculated with equation (4.5) [cf. FI11]. Each one of the rotation matrices R and R' contains three parameters (θ, φ, ψ) as seen in equation (2.3).

$$H = K_{rect} \cdot R \cdot K^{-1} \text{ and } H' = K_{rect} \cdot R' \cdot K^{-1} \tag{4.5}$$

One parameter, one rotation around the x-axis, is set to zero which results in absolute five parameters for the rotations. Therefore six degrees of freedom are used in total.

The parameters are calculated with the Levenberg-Marquardt-Algorithm [Mar63] which minimizes the Sampson error [HZ04, 98ff.] (E_S). The Sampson error for the i-th point correspondence (E_S^i) is shown in equation (4.6) [cf. FI11].

$$E_S^i = \frac{(q_i^T \cdot F \cdot q_i')^2}{||[u_3]_\times \cdot F \cdot q_i'||^2 + ||q_i^T \cdot F \cdot [u_3]_\times||^2} \text{ with } u_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.6}$$
$$\text{and } F = K^{-T} \cdot R^T \cdot F_{rect} \cdot R' \cdot K^{-1} \text{ (with } F_{rect} \text{ as in equation (2.9))}$$

With these parameters the homographies are calculated with equation (4.5). Furthermore this algorithm estimates the intrinsics matrix and the fundamental matrix. More information about QER can be found in [FI11] and [Mon11].

4.1.4 Polar rectification

It is also possible to rectify two images by "reparameteriz[es] the image[s] with polar coordinates (around the epipoles)" [PKV99]. This approach is known as Polar rectification. It reduces the matching ambiguity to half epipolar lines by orienting them. The orientation is needed if the epipole lies within the image, so that it makes a difference if the epipolar line is for example above or below the epipole. This is done by determining the sign of the homographies which transfer one epipolar line (l) to the other one (l'). This is called a line transfer and can be seen in equation (4.7) [cf. PKV99].

$$\begin{gathered} l' = H^{-T} \cdot l \text{ or } l = H^T \cdot l', \text{ where } H = [e']_\times \cdot F + e' \cdot a^T \\ \text{with } a \text{ as a freely chosen vector such that } det(H) \neq 0. \end{gathered} \tag{4.7}$$

The sign of H is the sign of l^T multiplied with an arbitrary point from the given point correspondences.

The epipoles and the epipolar lines that go through the outer edges of the images are calculated. Those lines are transferred to the other image with equation (4.7). This allows to determine the maximum angle of the image which can be seen in figure 4.1 as Θ_{max}. The rectification is done row by row in the image, where each row represents an epipolar line with a certain angle from the epipole. The minimum distance between the epipolar lines must be at least greater or equal to one pixel at the opposite border of the image to not lose any image information during the transformation.

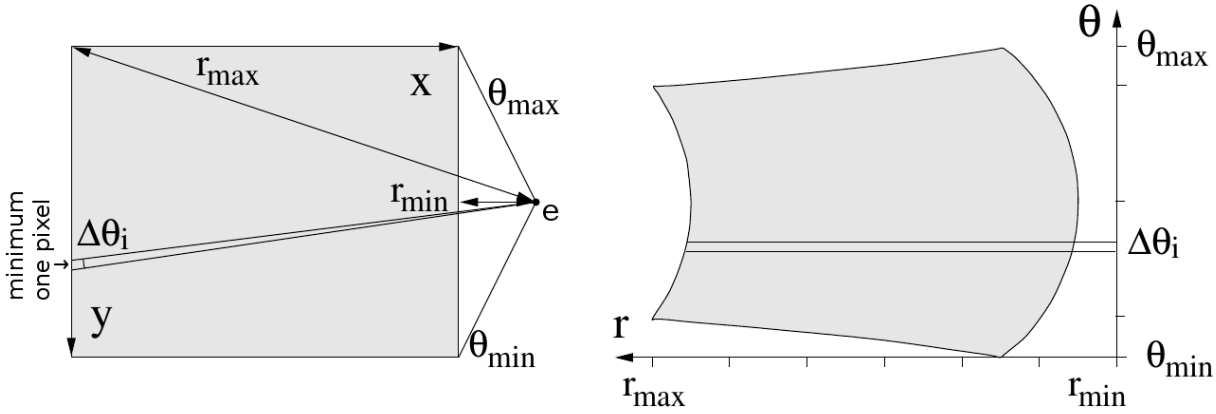


Figure 4.1: The mapping from (x, y)-space to (r, θ)-space

Source: [cf. figure 6 in PKV99]

This procedure represents a lossless mapping from a Cartesian (x, y) coordinate system to an angular (r, θ) coordinate system shown in figure 4.1.

Note that this description only captures the coarse steps of the algorithm. For more detailed information see [PKV99].

4.1.5 Special properties

The algorithms have a few special properties that are described in the following.

Epipole near image If the epipole is in or nearby the image all algorithms except polar rectification do fail, because polar rectification does a transformation into polar coordinates instead of a planar mapping. On the downside this causes the images to be enlarged.

Shearing and zooming Homographies calculated by QER sometimes zoom into the images, which results in a loss of information. Using Hartley's rectification it is possible that the images are sheared in x-direction. An additional shearing transformation (S), introduced by [LZ99], can be used to overcome the shearing and reduce the zooming problem. It can be seen in equation (4.8) [cf. San12]. In the equation w refers to the image width, h to the image height and H to the rectifying homography.

$$S = \begin{bmatrix} a & b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{ with } a = \frac{h^2 \cdot p_y^2 + w^2 + q_y^2}{h \cdot w \cdot (p_y \cdot q_x - p_x \cdot q_y)}, \; b = \frac{h^2 \cdot p_x \cdot p_y + h^2 + q_x \cdot q_y}{h \cdot w \cdot (p_x \cdot q_y - p_y \cdot q_x)},$$
$$p = H \cdot \begin{bmatrix} w-1 \\ \frac{h-1}{2} \\ 1 \end{bmatrix} - H \cdot \begin{bmatrix} 0 \\ \frac{h-1}{2} \\ 1 \end{bmatrix} \text{ and } q = H \cdot \begin{bmatrix} \frac{w-1}{2} \\ h-1 \\ 1 \end{bmatrix} - H \cdot \begin{bmatrix} \frac{w-1}{2} \\ 0 \\ 1 \end{bmatrix} \tag{4.8}$$

4.2 Evaluation

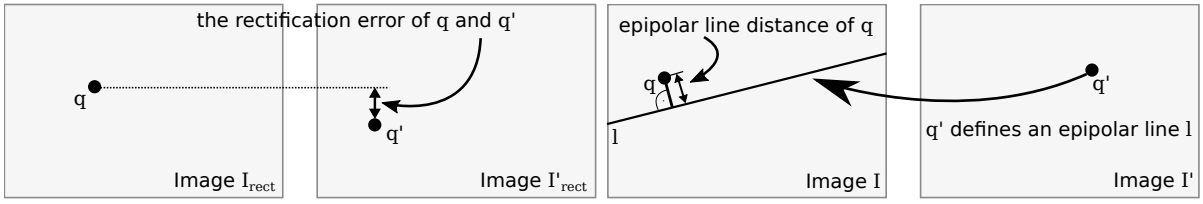
In order to decide, which algorithm should be used for rectification for chapter 5, an evaluation of the described algorithms will be done. This section is divided into four parts, the first depicts the evaluation criteria that are calculated, the second shows the procedure of the evaluation, the third describes the used test scenarios and the fourth presents the results of the evaluation.

4.2.1 Evaluation criteria

In order to evaluate the algorithms metrics need to be introduced. In the frame of this thesis the *average rectification error (ARE)*, the *average epipolar line distance (AELD)*, the *point ratio (PR)*, the failed rectifications and the runtime is used to evaluate the performance of the rectification algorihtms.

The ARE is defined as the average distance between the ordinate of the rectified point correspondences. It is shown in equation (4.9). N refers to the amount of given point correspondences and q_{iy} is the ordinate of point q_i. A geometrical representation of this metric is visualized in figure 4.2a where I_{rect} refers to a rectified image.

$$ARE = \frac{1}{N} \cdot \sum_{i}^{N} |q_{iy} - q'_{iy}| \tag{4.9}$$



(a) ARE **(b)** AELD

Figure 4.2: Geometrical representation of one summand of the ARE and the AELD.

The AELD is the average distance over all point correspondences between the epipolar line of a point and its matching point. Equation (4.10) shows its calculation where a, b, c are the coefficients of the epipolar line l calculated with $l = F^T \cdot q'$. Geometrically this is shown in figure 4.2b.

$$\begin{aligned} AELD &= \frac{1}{N} \cdot \sum_{i}^{N} (||ep_{ix} - q_{ix}|| + ||ep_{iy} - q_{iy}||), \text{ with} \\ ep_x &= \frac{b \cdot (b \cdot q_x - a \cdot q_y) - a \cdot c}{a^2 + b^2}, \; ep_y = -\frac{c + a \cdot ep_x}{b} \end{aligned} \tag{4.10}$$

The PR instead is the ratio between the amount of rectified point correspondences (N_r) which can still be found in the image after the rectification and the amount of original ones (N_o). It is shown in equation (4.11). This can be lower than one if the rectification transformation maps some points outside the viewable area.

$$PR = \frac{N_r}{N_o} \tag{4.11}$$

A rectification is considered a failure if $PR < 0.5$. In such a case the rectification is most likely very distorted due to the fact that it maps at least half of the point correspondences outside a viewable area.

4.2.2 Procedure

The experimental approach used goes through the following steps:

1. Calculating point correspondences
2. Removing outliers of calculated points
3. Defining inner and outer points
4. Calculate the fundamental matrix with the inner points
5. Calculate rectification transformation and rectify the images
6. Calculate the metrics

Note that the runtime is only measured in step 4 and 5. In the following these steps are described in detail.

1. Calculating point correspondences

In the first step the point correspondences are calculated with an appropriate algorithm (e.g. affine-scale-invariant feature transform (ASIFT) (see [MY09])). It is also possible to provide the correspondences manually.

2. Removing outliers of calculated points

After the point correspondences were calculated possible outliers need to be detected and removed afterwards. As an example see figure 4.3 where the red lines are the removed outliers. Outliers may worsen the rectification transformations. The idea behind this is when putting both images next to each other the gradients (m) between the corresponding points should be close together.

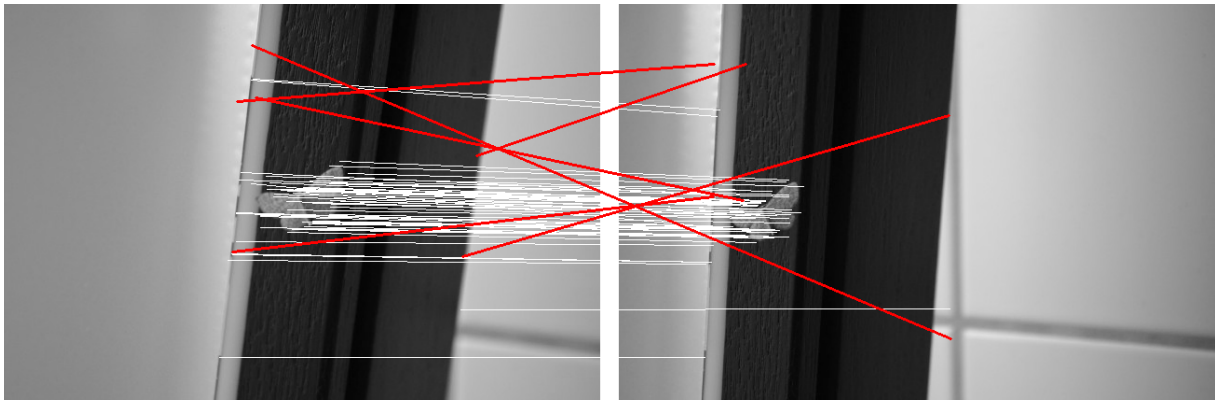


Figure 4.3: Calculated outliers in a stereo image.

Equation (4.12) illustrates the calculation of the gradient between the two points of a point correspondence (with w being the width of one image).

$$m = \frac{q'_y - q_y}{(q'_x + w) - q_x} \tag{4.12}$$

The gradient m is afterwards transformed into the angle of gradient in degree measure (α) with equation (4.13)

$$\alpha = \frac{\arctan(m)}{\Pi \cdot 180^\circ} \tag{4.13}$$

In the following the median angle of gradient ($\tilde{\alpha}$) is taken and all correspondences with an angle of gradient that do fulfill equation (4.14) are considered outliers. The ε is a chosen threshold.

$$|\alpha - \tilde{\alpha}| > \varepsilon \tag{4.14}$$

3. Defining inner and outer points

After the outliers are removed a few point correspondences are defined as *inner points*. These correspondences are used for the rectification. The remaining points are defined as *outer points*. Outer points are used to see if the rectification also rectifies the points, that were not taken into account while calculating the transformations.

4. Calculate the fundamental matrix with the inner points

Hartley's algorithm (section 4.1.2) and Polar rectification (section 4.1.4) need the fundamental matrix as input. Different approaches are used in the experiments which are briefly introduced in the following.

Using 8-point algorithm In order to calculate the fundamental matrix the 8-point algorithm, described in [Lon81], in conjunction with random sample consensus (RANSAC) can be used. "[T]he basic idea of RANSAC is to solve the problem many times using a random subset of the points and then take the particular solution closest to the average or the median solution" [BK08, p. 425]. This is further referred as *8-point-F-matrix*. For more information on RANSAC see [FB81].

Using QER Furthermore QER (section 4.1.3) can be utilized for determining the fundamental matrix. This is further referred to as *QER-F-matrix*.

Using intrinsics Another approach is using a predefined camera intrinsics matrix (K_{pre}) which simulates a calibrated camera. Using this and the point correspondences the essential matrix can be calculated with the 5-point algorithm, described in [Nis04]. Afterwards the fundamental matrix can be calculated with equation (2.8). This is further referred as *calib-F-matrix*.

5. Calculate rectification transformation and rectify the images

After the fundamental matrix is determined the rectifying transformations are calculated and performed. Depending on the algorithm the results are either two lookup maps or two homographies. The lookup maps contain also interpolated values between the rectified points to produce images without gaps [cf. BK08, 436f.].

6. Calculate the metrics

The AELD can be calculated with the fundamental matrix only, while the other two need the rectified points. If the rectification transformations are two homographies than the rectified inner and outer points (q_r) can be calculated with equation (4.15).

$$q_r = H \cdot q \text{ and } q'_r = H' \cdot q' \tag{4.15}$$

If the rectification transformations are two lookup maps the rectified points can not be calculated with equation (4.15) or a similar method, because maps cannot be used on single points but only on whole images. Hence the in figure 4.4 illustrated approach is used to get the rectified correspondences. The illustrated method begins with one black

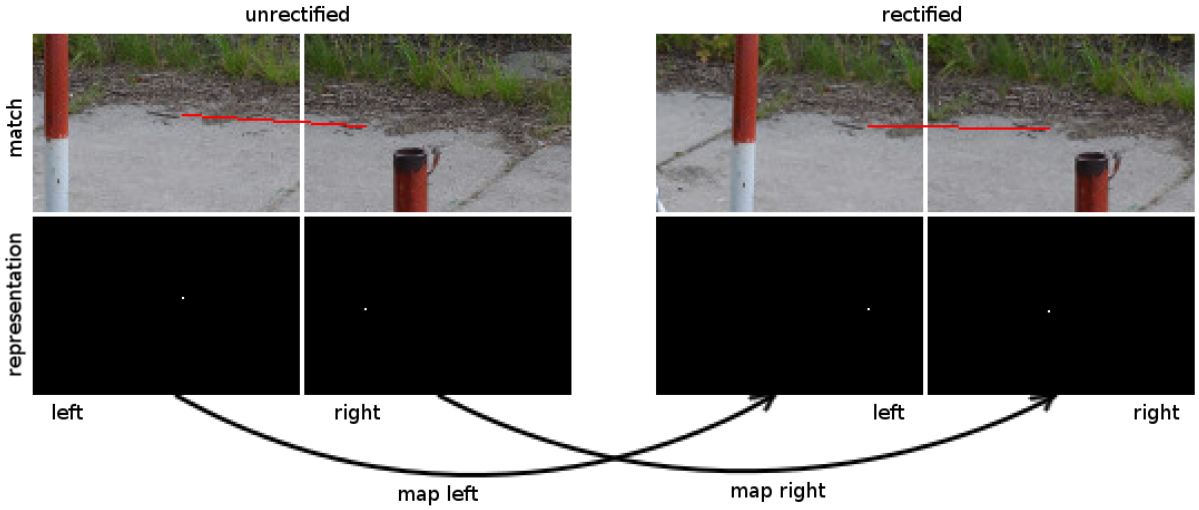


Figure 4.4: Getting rectified correspondences with maps. In the rectified images the white pixels are searched to get the rectified correspondence.

image of the same size as the original image. On this image the left point from a point correspondence is put as a white pixel. Afterwards the black image is rectified with the left rectification map and the single white pixel is searched through the image. This is done again for the right point of the correspondence with the right rectification map.

This procedure is done correspondence-by-correspondence and results in the rectified points. A problem of this approach is that lookup maps sometimes stretch points as illustrated in figure 4.5. In order to gather the approximate correct point all white points in the resulting map are searched and the arithmetic mean is calculated among them (which results in the red point in figure 4.5).

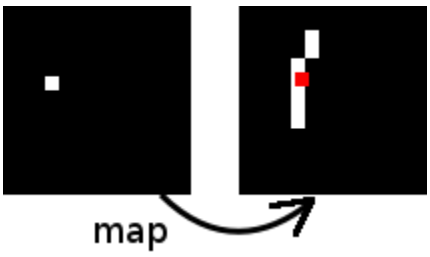


Figure 4.5: Stretching of a point.

Hartley's algorithm and QER return homographies, while Bouget's algorithm and Polar rectification return lookup maps. Either way it can happen that the transformations map points outside a viewable area. With the rectified points the inner and outer ARE and PR can be calculated.

4.2.3 Test scenarios

Different test scenarios and combinations of algorithms are used. The algorithms are as follows:

- Bouget's algorithm (further referred as *Bouget's*) with a predefined camera intrinsics matrix K_{pre}. This allows to calculate the essential matrix with the 5-point algorithm, described in [Nis04] and therefore also the rotation R and translation t.
- QER
- Hartley's algorithm with the 8-point-F-matrix (further referred as *Hartley's 8-point*)
- Hartley's algorithm with the calib-F-matrix (further referred as *Hartley's calib*)
- Hartley's algorithm with the QER-F-matrix (further referred as *Hartley's QER*)
- Polar rectification with the 8-point-F-matrix (further referred as *Polar rect. 8-point*)
- Polar rectification with the calib-F-matrix (further referred as *Polar rect. calib*).
- Polar rectification with the QER-F-matrix (further referred as *Polar rect. QER*)

For the experiments thirty stereo images (test sets) where photographed with a Nikon D5100 with a Nikkor 18-55mm lens. The images have an initial format of 4928×3264 and were resized to 640×424. The test sets can be seen in appendix B. Four test scenarios were defined:

- The first scenario runs over all test sets and uses ASIFT to gather the point correspondences. In this scenario all correspondences are considered inner points. This is further referred as *ASIFTnOut*.
- The second scenario is the same as the first besides that outer points are defined. This is done by defining a rectangle over one of the images and considering all points that lay out of that rectangle as inner points and the remaining as outer points. This scenario is further referred as *ASIFTwOut*.
- The third scenario uses manual matches with all matches as inner points and runs only on a little subset of the whole data. This is further referred as *MANnOut*.

- The last scenario is like the previous one besides that furthermore outer points are manually defined. This is further referred as *MANwOut*.

In each scenario the evaluation criteria are averaged for each algorithm.

4.2.4 Results

For the experiments the threshold ε for removing the outliers is set to 2.5° and the predefined camera intrinsics matrix K_{pre} is chosen as shown in equation (4.16) where w refers to the image width and h to the image height.

$$K_{pre} = \begin{bmatrix} 55 & 0 & \frac{w-1}{2} \\ 0 & 55 & \frac{h-1}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.16}$$

As metric for AELD the euclidean distance is used. In the following tables the algorithms are shown in the rows and the test scenarios are shown in the columns. In the ARE and AELD case the columns are further divided into:

all	which refers to all point correspondences,
inner	which refers to the inner point correspondences and
outer	which refers to the outer point correspondences

AELD The perfect value for the AELD metric is zero. It means that corresponding points lay exactly on the epipolar line. As can be seen in table 4.1 the AELD is best with a fundamental matrix calculated with QER, while it is worst with the calib-F-matrix.

F-mat algorithm	ASIFTnOut	ASIFTwOut	MANnOut	MANwOut
calib-F-matrix	2.72	2.56	3.72	4.53
QER-F-matrix	**1.07**	**1.06**	**0.42**	**0.33**
8-point-F-matrix	1.59	1.40	0.67	0.38

Table 4.1: The average AELD on the test scenarios.

This is expected, because the calib-F-matrix needs a calibrated camera, which is not always present. Even if the camera was calibrated once not all test images were taken with the same parameters. The focal length varied due to a zoom lens and also the rotation and translation between shots is not fixed. Therefore the calib-F-matrix seems unusable for the freehand stereo approach.

ARE The ARE states the distance of the rectified points. The perfect value is zero too which implies that every point correspondence is exactly row aligned. The results of the ARE criterion are shown in table 4.2. The table shows that the inner ARE is usually lower than the outer ARE which is expected, because the outer matches are not considered during the minimization of the rectifying transformation. The table also shows that 8-point-F-matrix worse than QER-F-matrix. Polar rectification has usually higher values than the other algorithms, which is partially due to the stretching of a few points as described in section 4.2.2.

	ASIFTnOut	**ASIFTwOut**		**MANnOut**	**MANwOut**	
Algorithm	all	inner	outer	all	inner	outer
Bouget's	4.54	5.73	5.80	10.37	32.17	41.38
QER	1.23	1.23	1.61	0.44	0.36	0.73
Hartley's 8-point	1.34	2.23	**1.43**	0.58	**0.29**	2.63
Hartley's calib	1.88	1.86	2.73	3.55	2.61	23.11
Hartley's QER	**1.06**	**1.05**	1.45	**0.42**	0.33	**0.69**
Polar rect. 8-point	3.93	3.04	3.15	2.46	1.11	2.85
Polar rect. calib	9.25	12.35	13.48	6.93	9.06	23.08
Polar rect. QER	2.17	2.21	2.75	0.79	0.30	0.76

Table 4.2: The average inner and outer ARE on the test scenarios.

The best algorithms in this criteria are Hartley's QER and QER itself. Table 4.2 also strengthens the assumption that the calib-F-matrix is not usable for this approach as it performs worse in the Hartley's configurations and the Polar rect. ones also Bouget's does not work well, because it also uses calibrated cameras which were only simulated in the experiments.

PR The PR measures the ratio between the point correspondences that are in a viewable area after the rectification and original point correspondences. The best value is one which means every point correspondence is found in the rectified image. The results of the PR criterion are visualized in table 4.3. In this criterion Bouget's, Hartley's calib and Hartley's 8-point perform worse than the other five. It can be seen that Polar rectification works best in most cases and that QER performs worse than Hartley's QER. This further allows the assumption that Polar rectification and Hartley's QER maintain more information in an image than the other algorithms.

These results are also reflected onto the failed rectifications as seen in table 4.4. In this table algorithms that never fail are omitted.

This table shows that Bouget's and Hartley's calib perform worse than the others in this use case. Furthermore QER almost never fails and performs superior to Hartley's 8-point.

Algorithm	ASIFTnOut	ASIFTwOut		MANnOut	MANwOut	
	all	inner	outer	all	inner	outer
Bouget's	0.69	0.64	0.70	0.73	0.41	0.31
QER	0.95	0.95	0.98	**1.00**	0.97	**1.00**
Hartley's 8-point	0.85	0.80	0.83	0.70	0.72	0.69
Hartley's calib	0.73	0.70	0.74	0.95	0.28	0.28
Hartley's QER	0.99	**0.99**	**1.00**	**1.00**	0.97	0.97
Polar rect. 8-point	0.99	**0.99**	0.99	**1.00**	**1.00**	0.97
Polar rect. calib	0.97	0.98	0.99	**1.00**	0.97	0.91
Polar rect. QER	**1.00**	**0.99**	0.99	**1.00**	**1.00**	0.97

Table 4.3: The average inner and outer PR in the test scenarios.

Algorithm	ASIFTnOut	ASIFTwOut	MANnOut	MANwOut
Bouget's	23%	37%	25%	50%
QER	3%	3%	0%	0%
Hartley's 8-point	13%	17%	25%	25%
Hartley's calib	27%	27%	0%	50%

Table 4.4: The percentage of failed rectifications over all tests scenarios.

Runtime The runtime of the algorithms is calculated as the arithmetic mean over all test sets and is divided into two parts. Table 4.5a states the time for the fundamental matrix calculation and table 4.5b for calculating the rectification transformations. Polar rectification and QER perform worse than Bouget's algorithm and Hartley's algorithm which performs significant better than the others as seen in table 4.5b. The calculation of the calib-F-matrix is very slow compared to the others as illustrated in table 4.5a. This is due to the 5-point algorithm which itself is slow.

F-mat algorithm	Average runtime
calib-F-matrix	1754.3ms
QER-F-matrix	26.8ms
8-point-F-matrix	**6.8ms**

(a) Calculating the fundamental matrix.

Algorithm	Average runtime
Bouget's	13.2ms
QER	26.3ms
Hartley's	**0.2ms**
Polar rect.	23.3ms

(b) Calculating rectification transformations.

Table 4.5: Average runtime over all test scenarios and test sets

Summary

The results gathered show that the algorithms that need a calibrated camera are not usable for this freehand stereo approach which was assumed. Furthermore the results suggest that algorithms with the QER-F-matrix work better than the ones using the 8-point-F-matrix. This rules out the algorithms with the 8-point-F-matrix as the rectification algorithm for the preprocessing step of the stereo matching algorithm introduced in chapter 5.

Furthermore it can be seen that polar rectification works best in maintaining information of the images but does worse in rectifying the points than QER and Hartley's QER. Additionally polar rectification often enlarges the images, which enlarges the search space for correspondences and therefore slows down the stereo matching algorithm. Thus polar rectification was ruled out. In the end only QER and Hartley's QER remain. The results show that Hartley's QER outperforms the pure QER rectification which is unexpected and is hence examined further.

Worse ARE values The ARE of QER may be worse because it is possible that Hartley's QER compresses the images which should result in lower ARE. In order to see whether this assumption is true, the following algorithm is performed on each test set:

1. Take three points $(P_1(20, h/2 - 10), P_2(w/2, h/2 - 10), P_3(w - 20, h/2 - 10)$ and three points (P'_i with $i \in \{1, 2, 3\}$) with $P'_{ix} = P_{ix}$ and $P'_{iy} = P_{iy} + 20$.
2. Perform QER rectification and calculate average y-distance (ΔQy) between points P_{irect} and P'_{irect} with $i \in \{1, 2, 3\}$ and P_{irect} as the rectified point of P_i.
3. Do the same with Hartley's QER rectification (ΔHQy)
4. Calculate $\Delta QHQy = \Delta Qy - \Delta HQy$
5. Calculate difference between ARE of QER and ARE of Hartley's QER (ΔARE_{QHQ})

To support the assumption $\Delta QHQy$ and ΔARE_{QHQ} should correlate in a monotonically increasing way which is the case. This is shown in figure 4.6. It can also be seen that the slopes of the correlation lines are $\ll 1$. This strengthens the assumption that QER performs indeed better than Hartley's QER regarding the ARE criteria and is just considered worse because it does not compress images.

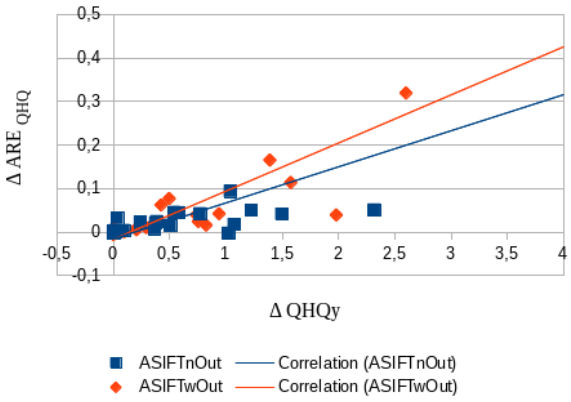


Figure 4.6: Correlation between $\Delta QHQy$ and ΔARE_{QHQ} without failed images.

Worse PR value The PR is worse because some images fail, due to the zoom error, mentioned in section 4.1.5. Figure 4.7 shows a failed QER and its working Hartley's QER rectification. As shown in figure 4.7a the rectification is not entirely wrong but it does omit information on the borders of the image.

(a) QER rectification **(b)** Hartley's QER rectification

Figure 4.7: Rectification of the left image of the bicycles test set.

Conclusion These tests show that the ARE criteria needs to be revised regarding the stretching or compressing error problem or should be replaced by another criteria.

Nevertheless this problem allows the statement that Hartley's QER maybe performs worse in the ARE criteria than QER, but as the PR criteria shows it does not suffer from the zooming problem. Hence Hartley's QER is chosen for this thesis as rectification algorithm.

4.3 Summary

In this chapter algorithms for rectifying an image were discussed. There is one algorithm as representative for algorithms that need calibrated cameras (Bouget's algorithm) and three algorithms for uncalibrated cameras. Furthermore criteria were defined for evaluating the performance of different algorithms.

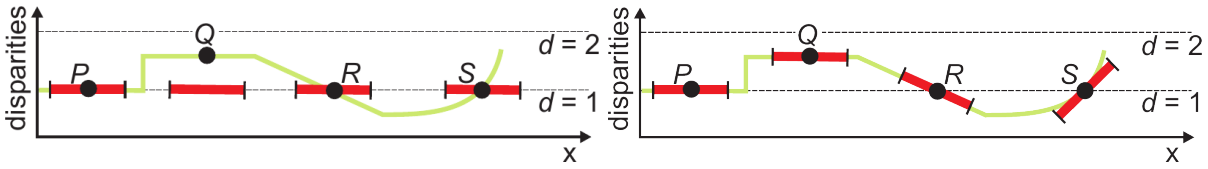
The results gathered suggest that algorithms that need calibrated cameras are not useful in a use case were such can not be assumed. Based on the results Hartley's algorithm in conjunction with the QER fundamental matrix was chosen. It performed best on the chosen metrics. This algorithm is further used as rectification algorithm for PatchMatch Stereo (see chapter 5).

5 Extending PatchMatch Stereo

This chapter aims at the extension of PatchMatch Stereo. First a description of the PatchMatch Stereo algorithm will be given. Afterwards possible improvements will be introduced and evaluated. Finally the knowledge gained from the evaluation will be summarized.

5.1 PatchMatch Stereo

The PatchMatch Stereo algorithm aims at disparity calculation with slanted support windows that consist of 3D planes, that fit the scenery for each point. This way slanted surfaces can be modeled as shown in figure 5.1b which allows even subpixel accuracy. Other algorithms such as [Hos+09],[Ric+10] and [Rhe+11] use fronto-parallel support windows which cannot model the points on slanted surfaces properly as illustrated in figure 5.1a [cf. BRR11].



(a) Fronto-parallel support regions **(b)** Slanted support regionss

Figure 5.1: Support regions in a 1D scenario. The points on the green line are modeled considering the red support regions.

Source: [cf. figure 1 in BRR11]

The algorithm is divided into three components: *1)* a data model with a cost function defined upon it, *2)* an algorithm that minimizes that cost function and *3)* some post processing. The components will be described in the following.

5.1.1 Data model

The data is modeled via planes in the 3D space. A plane f is defined over a point $P = (x_0, y_0, z_0)$ and its normal vector $\vec{n} = (n_x, n_y, n_z)$. It is furthermore parametrized with three parameters a, b, c that can be calculated with equation (5.1) [cf. BRR11].

$$f_a := -\frac{n_x}{n_z},\ f_b := -\frac{n_y}{n_z} \text{ and } f_c := \frac{n_x \cdot x_0 + n_y \cdot y_0 + n_z \cdot z_0}{n_z} \tag{5.1}$$

The disparity d of a point p can then be calculated with equation (5.2) [cf. BRR11].

$$d(p, f) = f_a \cdot p_x + f_b \cdot p_y + f_c \tag{5.2}$$

The goal is to find a plane for each point that minimizes the aggregated cost m of a square window W_p around this point. W_p is called the support window and is centered on p. Its size can be freely defined. The aggregated cost m can be calculated with equation (5.3) [cf. BRR11].

$$m(p, f) = \sum_{q \in W_p} w(p, q) \cdot \rho(q, q - d(q, f)) \tag{5.3}$$

Thereby is w a weight function and ρ computes the point dissimilarity between q and its matching point q'. The support window W_p with the the considered matching points are visualized in figure 5.2.

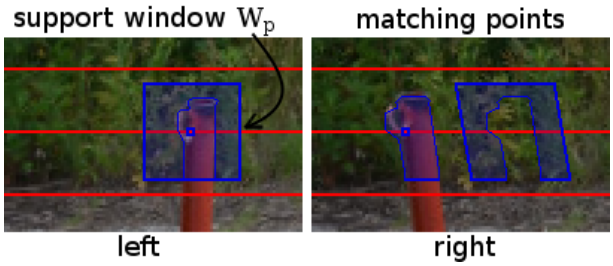


Figure 5.2: Support window with corresponding matchings

As the weight function the likelihood of the L1-distance ($||\cdot||_1$) between the colors of the points in RGB space (I_q) is used. This boils down to equation (5.4) [cf. BRR11] where γ is a user defined parameter.

$$w(p, q) = e^{\frac{-||I_p - I_q||_1}{\gamma}} \tag{5.4}$$

With the user defined parameters α, τ_{col} and τ_{grad} and the gray valued gradient of image I on point q (∇I_q) the point dissimilarity can be calculated, as shown in equation (5.5) [cf. BRR11].

$$\rho(q, q') = (1 - \alpha) \cdot min(||I_q - I'_{q'}||_1, \tau_{col}) + \alpha \cdot min(||\nabla I_q - \nabla I'_{q'}||_1, \tau_{grad}) \tag{5.5}$$

In the next part the procedure for minimizing this cost function is described.

5.1.2 Cost minimization

The problem is that each point has an infinite amount of possible planes which can not be calculated. That is where PatchMatch ([Bar+09]) is used. PatchMatch itself "is an approximate dense nearest neighbor algorithm" [BRR11]. It uses a random initialization and afterwards propagates good guesses. This is also done in PatchMatch Stereo, because

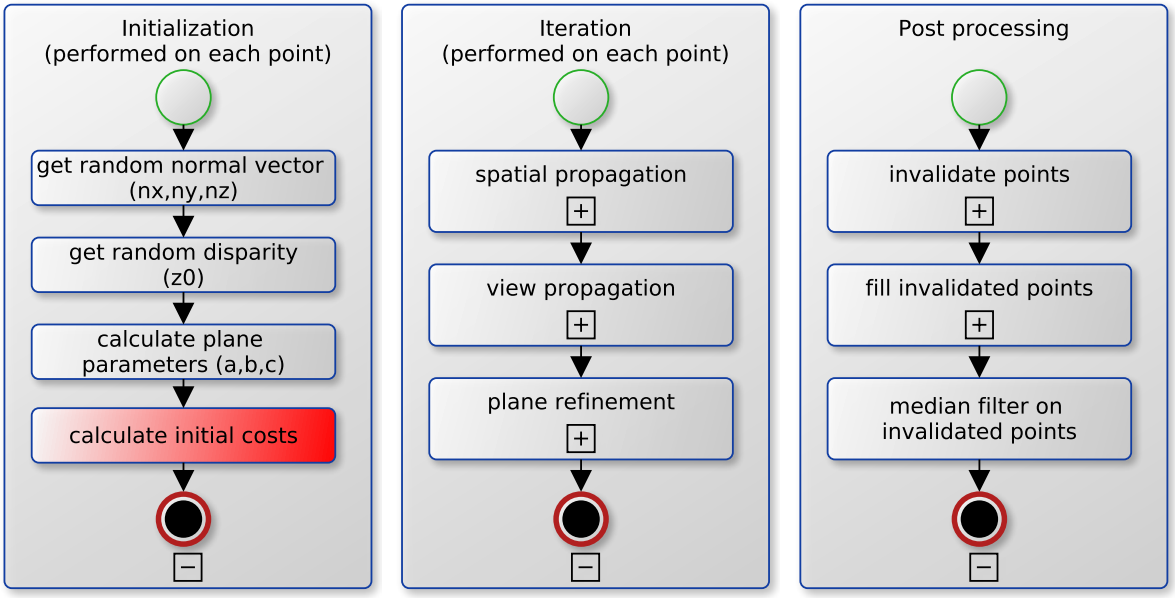
it can be assumed that in a natural stereo image large regions use approximately the same plane and therefore only one good guess in a region is needed, which can be propagated. The algorithm has mainly three steps:

1. initialization,
2. the iterations and
3. post processing.

The elements are described in the following paragraphs.

Initialization

As stated above the initialization of the planes is randomized, in a way that every point has a plane and an initial cost afterwards. Figure 5.3a visualizes the procedure where the red part refers to the cost calculation with equation (5.3). Note that the parameters of a plane are not randomized directly, but instead its normal vector $\vec{n}$ as unit vector and the disparity z_0 of the used point p (resulting in the point $P = (p_x, p_y, z_0)$). The disparity is chosen within the range of allowed continuous disparity values. With these randomized values the plane parameters a, b, c are calculated with equation (5.1). This allows the calculation of the initial costs for p with equation (5.3).



(a) Initialization **(b)** Iterations **(c)** Post processing

Figure 5.3: Activity diagrams for the components of PatchMatch Stereo.

Iterations

In order to minimize these costs the algorithm iterates multiple times over the images where three iterations are usually enough. An iteration works point-by-point. In even iterations the algorithm starts in the upper left corner on the left image, goes row-by-row down to the bottom right corner and starts again at the top left corner of the right image. In odd iteration this is switched in a way that it starts in bottom right corner of the left image and ends in the upper left corner of the right image. In each iteration each point goes through three[4] steps which can be seen in figure 5.3b. In each step it is checked whether the costs of point p can be improved by using another plane. The first two steps (spatial and view propagation) are illustrated by figure 5.4.

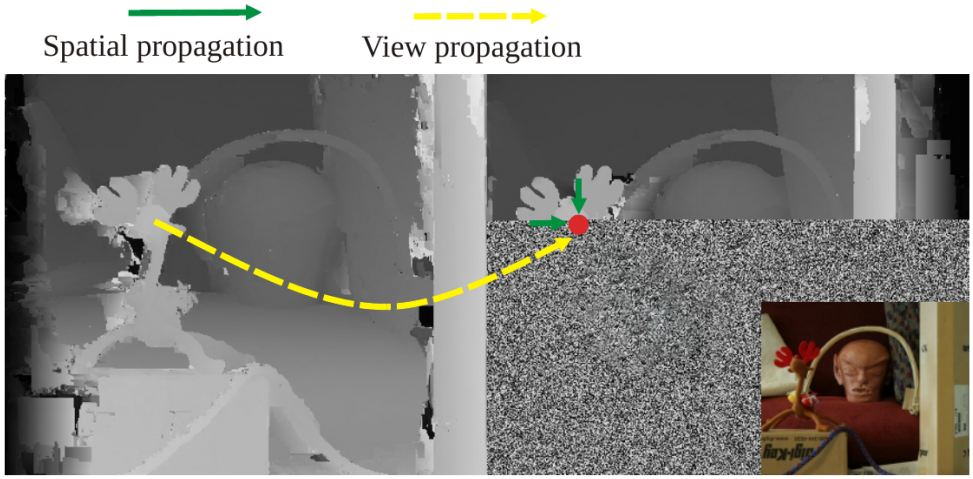


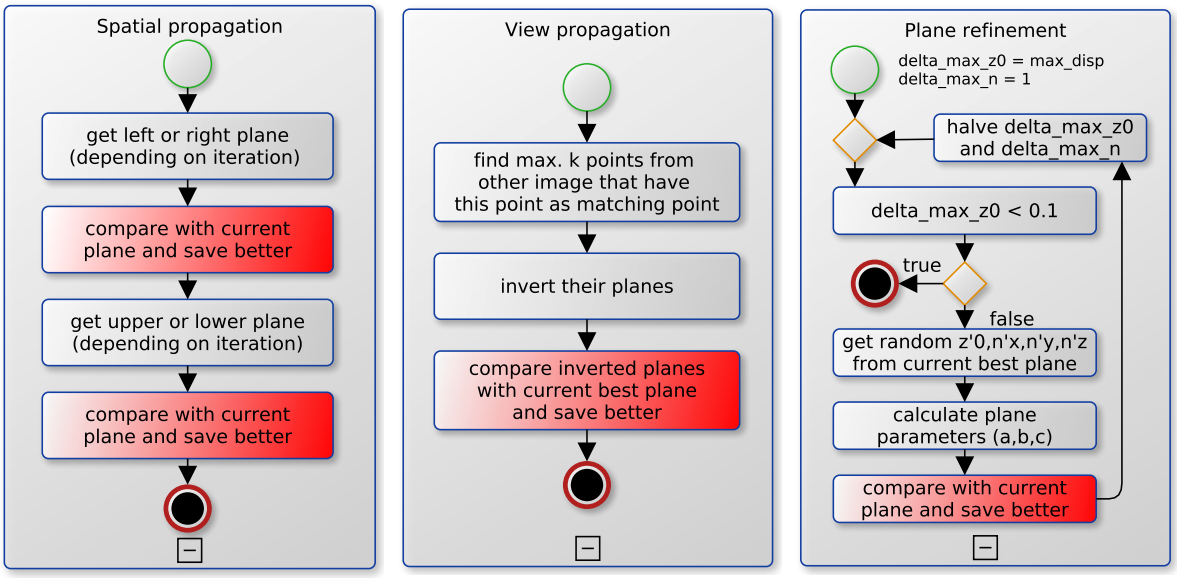
Figure 5.4: Illustration of spatial and view propagation

Source: [cf. figure 3(a) in BRR11]

Spatial propagation The first step is spatial propagation. In this step the planes of the left and upper neighbors (in even iterations) or the right and lower neighbors (in odd iterations) are checked against the current best plane of p. The procedure is illustrated in figure 5.5a.

View propagation Spatial propagation is followed by the view propagation. During this, as visualized in figure 5.5b, at most k planes of points from the other image that have p as matching point are used. These planes are inverted and checked against the current best plane of p. In general k is set to 5.

[4]The paper [BRR11] states four steps. Temporal propagation, which only works on image sequences, was avoided in this thesis.



(a) Spatial propagation **(b)** View propagation **(c)** Plane refinement

Figure 5.5: Activity diagrams for the components of one iteration. In the red elements the costs are calculated with equation (5.3) and are compared.

Plane refinement This step refines the parameters of the plane. It is illustrated in figure 5.5c. First two new parameters are introduced:

- $\Delta_{z_0}^{max}$ which is initially set to maximum disparity and
- Δ_n^{max} which is initially set to one.

These set limits on the maximum allowed changes to the values of the plane's point and normal vector form. The following procedure is repeated until $\Delta_{z_0}^{max} < 0.1$.

From the current best plane a new disparity z_0' is derived as can be seen in equation (5.6) [cf. BRR11] where Δz_0 is randomly picked from the interval $[-\Delta_{z_0}^{max}, \Delta_{z_0}^{max}]$.

$$z_0' := z_0 + \Delta z_0 \tag{5.6}$$

Furthermore a new normal vector $\vec{n'}$ is derived with equation (5.7) [cf. BRR11] where u computes the unit vector and the elements of $\vec{\Delta n}$ are randomly taken from the interval $[-\Delta_n^{max}, \Delta_n^{max}]$.

$$\vec{n'} := u(\vec{n} + \vec{\Delta n}) \tag{5.7}$$

With these values and equation (5.1) the new refined plane can be calculated. This plane is checked against the current best plane of p. After this the values $\Delta_{z_0}^{max}$ and Δ_n^{max} are halved. This allows larger changes in early iterations to compensate completely wrong planes and allow only little changes in higher iterations to capture disparity details.

5.1.3 Post processing

The post processing tackles mismatched and occluded points in the disparity map. Its steps are visualized in figure 5.3c. Points that do not fulfill equation (5.8) [cf. BRR11] are invalidated. In this equation d_p is the disparity of point p and $d_{p'}$ of its matching point.

$$|d_p - d_{p'}| \leq 1 \tag{5.8}$$

In order to replace the disparity of the invalidated points the planes of the closest valid points to the left and to the right are considered. Both disparities are calculated and the lowest one is taken. "Selecting the lower disparity is motivated by the fact that occlusion occurs at the background" [BRR11].

Afterwards a median filter is applied for the replaced disparities with the same weight function as in equation (5.4). This is done to weaken horizontal streaks in the disparity map.

The information for this algorithm are taken from [BRR11].

5.2 Improvements

In order to improve the quality and speed of the PatchMatch Stereo algorithm different attempts were made. These attempts are described in this section.

5.2.1 Extended search space

The improvement is primarily made in the view propagation step. Instead of looking only at a point's matching point a configurable amount of the points above and below that matching point are considered (this user defined parameter will be called *lines*) which can result in a y-disparity. In figure 5.6 the main idea of the improvement is illustrated.

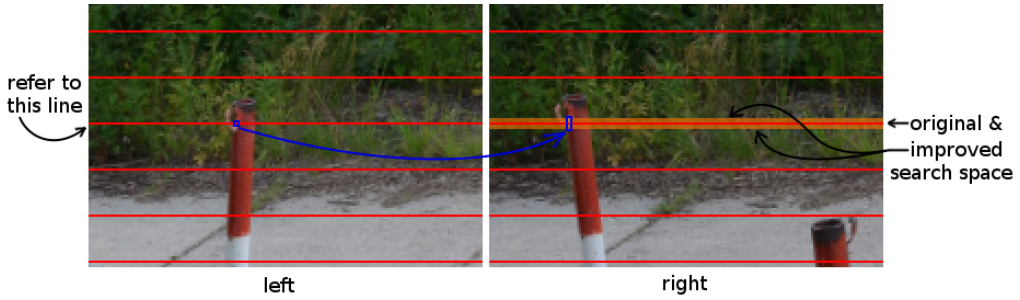


Figure 5.6: Enhanced search space method. The red lines are the rectified epipolar lines.

The blue elements shows the original point in the left image and the considered points in the right image. The orange lines refer to the extended search space for the point on the right image.

This is supposed to improve the results quality-wise especially if the epipolar geometry is not accurate.

Adjustments to the model

To apply these changes to the model a new parameter φ is added to the calculation, which contains the y-disparity for the current point. Hence the cost function, shown in equation (5.3), needs to be adjusted too, because the matches in the cost function should map to the extended search space as seen in figure 5.7. In order to calculate the costs for the green point (that lays in the extended search space) in the right image the green matching points shall be considered. But in equation (5.3) only the blue ones are. In order to use the green matching points equation (5.9) is introduced. Furthermore another cost term (Ψ) is added to this function which adds a penalty for higher y-disparity.

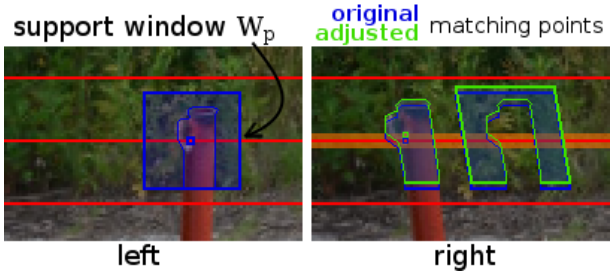


Figure 5.7: Adjustments on the cost function.

$$m(p, f, \varphi) = \sum_{q \in W_p} w(p, q) \cdot (\rho(q, q^{\varphi}) + \Psi(\rho(q, q^{\varphi}), \varphi)) \quad \text{with } q_x^{\varphi} = q_x - d(q, f),\ q_y^{\varphi} = q_y + \varphi \tag{5.9}$$

The used term Ψ is a linear function shown in equation (5.10) that adds a configurable part of ρ to the cost depending on the current φ. In this equation ψ is a user defined parameter

$$\Psi(\rho, \varphi) = \psi \cdot \rho \cdot |\varphi| \tag{5.10}$$

Note that the parameter φ is only altered for a point during the view propagation step. Initially it is set to zero. In the refinement step the value of φ stays the same.

Adjustment of the post processing

The post processing step invalidates points. In order to make this process aware of φ during invalidation the adjustments of the model have to be propagated to the post processing. This is done by equation (5.11).

$$|d_p - d_{p'^{\varphi}}| \leq 1, \text{ with } p_x'^{\varphi} = p_x' \text{ and } p_y'^{\varphi} = p_y + \varphi \tag{5.11}$$

5.2.2 Open Computing Language implementation

In order to improve the speed of the algorithm an Open Computing Language (OpenCL) implementation is considered. This would provide the possibility to utilize the present graphics processing unit (GPU).

OpenCL

"Open Computing Language (OpenCL) is an open [...] standard for general purpose parallel programming across CPUs, GPUs and other processors, giving software developers portable and efficient access to the power of these heterogeneous processing platforms" [Khr11, p. 12]. Therefore a program is divided into kernels which run on an OpenCL device and a host program which runs on the host and manages kernel execution [cf. Khr11, p. 23]. Figure 5.8 shows a simplification of the OpenCL execution model. The figure illustrates that the host program executes a kernel on a device. Each kernel is processed by multiple processing units in parallel and afterwards the next kernel is executed on the device.

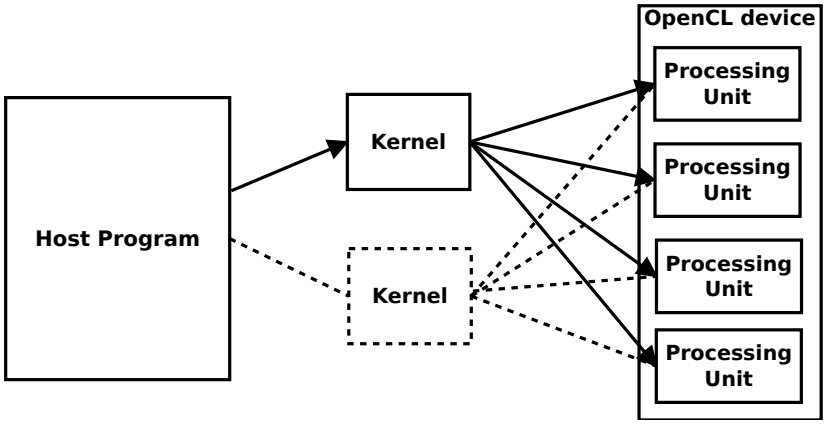


Figure 5.8: The simplified OpenCL execution model.

Note that this explanation only includes the basics and hence ignores certain details of the more complex execution model.

Porting PatchMatch Stereo to OpenCL

In order to achieve this some thoughts about the structure of the program became necessary. The trivial implementation with the complete iteration in one kernel is not sufficient because the used Nvidia GeForce GT550M GPU does not support this many operations

in one kernel. In order to overcome this problem the program is split into more kernels, which is illustrated in figure 5.9.

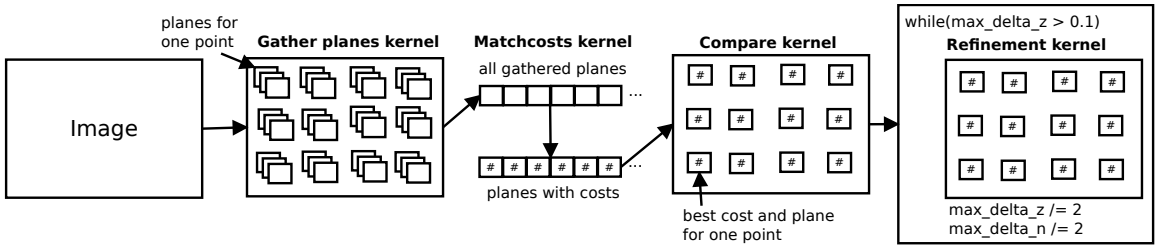


Figure 5.9: The basic execution sequence of the OpenCL kernels. Each block is executed parallel.

Kernel description The first kernel gathers all planes for each point, which should be two for spatial propagation and $k \cdot (2 \cdot lines + 1)$ for view propagation. The next kernel computes the costs with equation (5.9) for each plane. Afterwards the costs will be compared and the best plane is stored for its point. This can be done, because the cost function is not dependent on the current best plane nor the current best cost. The plane refinement instead should be done in another way because it depends on the current best plane for calculating a new one. Therefore the host code contains the iteration for the refinement, while the kernel only contains one calculation of a cost and its comparison with the current cost.

This could speed up the algorithm depending on the used hardware. However this approach contains another problem, due to the fact that spatial propagation relies on planes that were calculated before. Unfortunately this can not be assumed because each point is calculated independently and completely parallel. That means that the spatial propagation does not get the correct plane of the point before. In order to reduce this problem two possible solutions were considered.

Consider more points The first idea is to use $m > 1$ points, instead of taking only one point before. With this the spatial propagation gets the points as if m iterations were done. Therefore the correct plane is closer. In order to get the correct plane after the second iteration m should cover all points before but this has the downside of introducing to many more cost calculations. Figure 5.10 illustrates

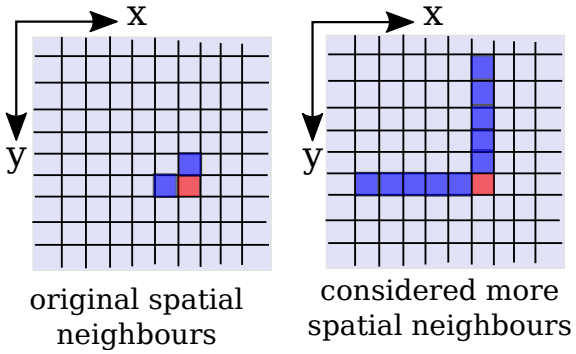


Figure 5.10: The original spatial propagation (left) and the one with more points (right).

this approach where the spatial propagation is performed on the red point. The blue points are the considered planes for spatial propagation.

Slicing the image The second idea is to consider the problem in each iteration row-by-row or column-by-column, instead of running the kernel on the whole image. That means that each point in a row or column is calculated parallel, but the rows or columns itself are computed in the right order. With this the spatial propagation should get the right plane at least in one direction in each iteration. Figure 5.11 shows the row-by-row execution sequence, the column-by-column sequence works in a similar fashion. The figure states that at the beginning the first row of the image goes through all steps and after its completion the second row is calculated and so on.

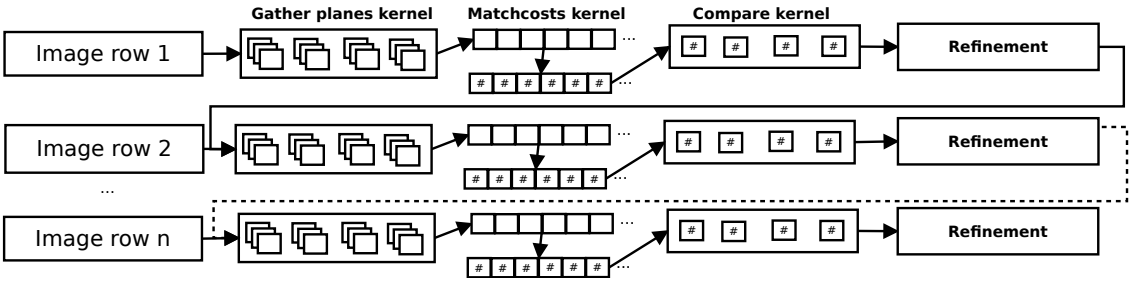


Figure 5.11: The row-by-row execution sequence of the OpenCL kernels

5.3 Implementation

This section will give an overview of the reimplementation as well details on special parts of the algorithm. Furthermore some other useful features of the program are described.

5.3.1 Reimplementation

A basic C++ implementation of PatchMatch Stereo was already present, but in order to introduce the adjustments a complete reimplementation became necessary, due to problems with the used data structures.

The main problem was the use of arrays that were allocated with *malloc* instead of the use of vectors. This may lead to memory leaks. Furthermore occasional segmentation faults occurred after the adjustments were implemented. In order to not spend too much time on debugging a reimplementation was the faster case. This way the data structure could be adapted to the needs of the algorithm and the algorithm itself could be further understood. Additionally the reimplementation uses the newer Open Source Computer Vision Library (OpenCV) application programming interface (API) which is easier to work

with than the old one that was used before. OpenCV "is an open source [...] computer vision library [...] written in C and C++" [BK08, p. 1]. It is a cross-platform library and is designed to be easy-to-use and computational efficient [cf. BK08, p. 1].

5.3.2 The planes

The planes are calculated to fit the scene and used for the disparity calculation. A plane is stored for each point in the image. Such a plane is a data structure, which consists of eight float values. The three plane parameters a, b, c, the three normal vector parameters n_x, n_y, n_z, the disparity z_0 and the parameter φ for the y-disparity.

It is possible to mark planes as dummy planes (which is needed by the view propagation step) by adding a value $c \in \mathbb{R}$ with $0 < c < 1$ to φ. This can be done because only the integer parts of the stored φ are used. These structure has the advantage that the OpenCL implementation can read and write a plane as a single float vector without considering other data types.

5.3.3 View propagation

In order to get the planes for the view propagation step, the matching planes are preprocessed before the current image is processed. Therefore the algorithm, which is shown in listing 5.1, iterates over the other image's planes and calculates the matching point for each plane.

```
for each point fp in image I'
  f = plane from image I' at fp;
  p = calculated matching point from f on image I;
  if(p out of border or
       p contains more neighbors than k)
    continue;
  else
    raise neigbor count of p by 1;

  for(φ = -lines; φ <= lines; ++φ)
    np = new Point(p_x, p_y + φ);
    if(np is out of border)
      mark as dummy plane;
      store in viewneighbors list;
    else
      nf = plane from other image at np;
```

```
invert nf;
save φ in nf;
store nf in viewneighbors list;
```

Listing 5.1: Pseudocode of preparing view propagation for image I.

If the matching point is in the image and does not already contain too much neighbors a plane for each φ in the interval $[-lines, lines]$ is stored.

In the iteration itself, during the view propagation step, all available neighbors are checked as shown in listing 5.2.

```
count = get the neighbor count of current point p;
for(i = 0; i < count; ++i)
  for(φ = -lines; φ <= lines, ++φ)
    f = get plane i from viewneighbors list at p with φ;
    if(f.is_dummy_plane())
      continue;
    else
      check whether f is better than current plane;
```

Listing 5.2: Pseudocode of view propagation step in the iteration

5.3.4 Validation features

In order to allow easier evaluation of the algorithm other features were implemented into the program. These are explained in the following.

Neighbor image

It is possible that the algorithm prefers a plane with $|\varphi| > 0$. In order to visualize those y-disparity an image is constructed. Such an image can be seen in figure 5.12.

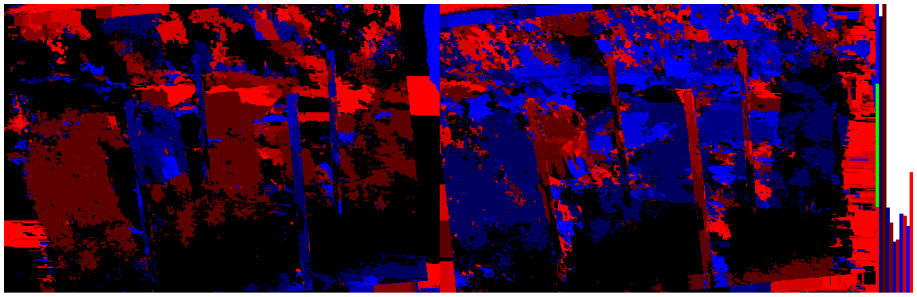
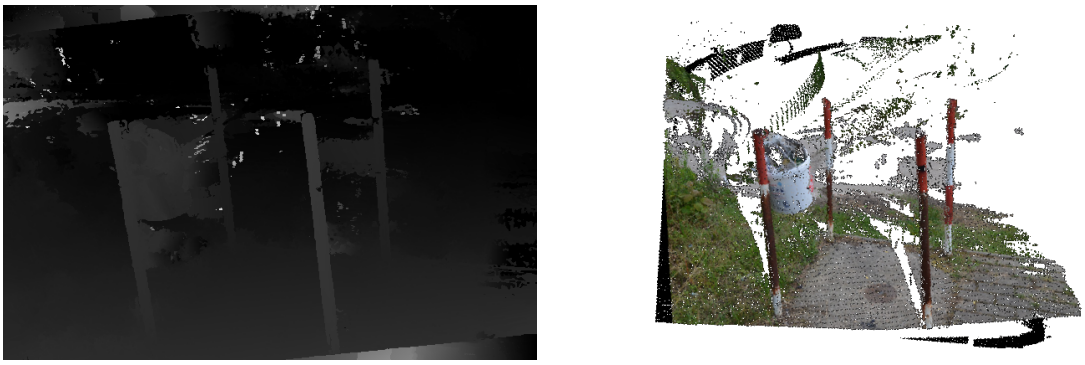


Figure 5.12: Neighbor image of the rectified poles test set with $lines = 5$ and $\psi = 0$.

The red parts refer to $\varphi < 0$ and the blue ones to $\varphi > 0$. The intensity of the color refers to the deviation from 0. The right part shows a histogram of the distribution of φ. As a side effect this image may allow a rudimentary segmentation of the objects in the image which may be useful for further improvements of the algorithm.

3D visualization

A method based upon equation (2.10) is implemented to visualize the results in 3D space. Therefore the missing parameters focal length f is set to 100 and the translation t is set to the disparity range. This is motivated by the fact that the maximum disparity is related to the translation of the closest object in the image. Together with the disparity map the depths of the points are calculated and can be visualized as a point cloud via the Point Cloud Library (PCL). In figure 5.13 a disparity map with the corresponding 3D visualization is shown.



(a) right disparity map **(b)** point cloud

Figure 5.13: Disparity map and its point cloud of the poles test set with $lines = 5$ and $\psi = 0$.

5.4 Evaluation

In order to see how the implemented improvements perform against vanilla PatchMatch Stereo an evaluation on defined scenarios has been done. First the used scenarios are described. Second the used evaluation criteria are introduced and last but not least the results of the experiments are presented and analyzed.

5.4.1 Test scenarios

Different test scenarios were used in order to determine in which circumstances the improvements are useful. These scenarios are PatchMatch Stereo with unrectified images (*unrect*) and with rectified images (*rightRect*).

Each test scenario itself is divided into different parameter configurations (parConfs). The considered parameters were as follows:

lines refers to the maximum y-disparity that is considered during view propagation step (above and below), while

ψ weights the y-disparity in a plane and adds costs for it (as in equation (5.10)).

For reference values the original PatchMatch Stereo (further referred as *origPMS*) is used. This can be achieved by setting $lines = \psi = 0$. Based on different constellations of *lines* and ψ the combinations shown in table 5.1 were chosen.

lines	ψ	referred as
5	0	*l5p0*
10	0	*l10p0*
5	0.01	*l5p0.01*
10	0.01	*l10p0.01*

Table 5.1: The different parameter configurations.

5.4.2 Evaluation criteria

It is hard to evaluate the adjustments of the algorithm properly, because freehand stereo images are used instead of images with ground truth data. Ground truth data is the data that was measured by humans point by point and is considered correct. This approach is useful because images with ground truth data are usually correctly rectified where the adjustments are not expected to achieve further improvements. Furthermore this allows to check how well PatchMatch Stereo performs on realistic data sets. In order to hold the evaluation short and concise the following evaluation criteria were defined.

Average costs The first criteria used are the average costs. The average was restricted to the median (instead of the arithmetic mean), because the arithmetic mean is not robust against outliers. Such outliers often occur at the border of an image. These values are unfortunately incomparable, because if the parameter $\psi \neq 0$ the cost functions are different. In order to partially overcome this issue equation (5.9) is used with $\psi = 0$ for calculating the median. Furthermore the coefficient of variation (CoV) over all median values is calculated which can be done with equation (5.12).

$$CoV = \frac{standard\ deviation}{arithmetic\ mean} \tag{5.12}$$

Distribution of costs A second criteria used was the distribution of costs over the image. This is visualized with a histogram as shown in figure 5.14, were the x-axis are classes of costs and the y-axis is the relative amount of costs in this class.

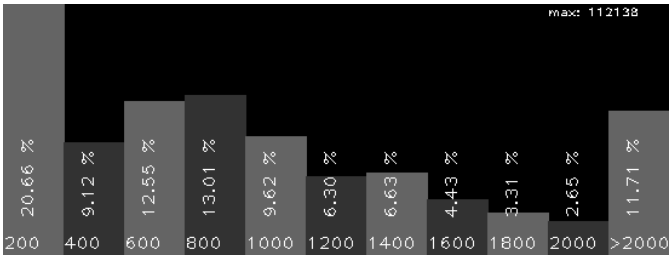


Figure 5.14: Histogram of a test set (butterfly l5p0)

Invalidated points Another important criterion is the amount of invalidated points (invPts) during the post processing step. If lesser points are invalidated the disparities are more likely to be correct. This criterion is independent of the used cost function.

Runtime The forth criterion used was the runtime for the algorithm needed to terminate. This allows to set the improvements in relation with the changed time consumption.

Rank In order to measure the quality of the results a subjective ranking is performed on them. Therefore the disparity maps and their 3D visualization, described in section 5.3.4, are ranked in the following ranks:

good If the shape of the objects can be seen clearly and only a few points are distorted.

ok If the objects itself have an appropriate shape but many other points are distorted.

bad If the objects can be determined but are in a bad shape.

fail If the result is highly distorted or the objects can not be determined or if huge parts from the background are put into the foreground or visa versa.

Furthermore the results are visually compared to each other and one is picked as *best* for each test set. This is not done if all parConfs of one test set are ranked *fail*.

5.4.3 Results

The evaluation was done on the explained test scenarios with respect to the criteria chosen. This led to the results which are described in this section. This section will first explain the test environment followed by the results of the extended search space. Afterwards the results of the OpenCL implementation are described.

Test environment

The experiment was performed on an ASUS X4GS Laptop with a Linux operating system and 4 GB random access memory (RAM). It utilizes an Intel Core I5 2410M central processing unit (CPU) with 2.3 Ghz with 2 cores and 2 threads each. The GPU used in the experiment was an Nvidia GeForce GT 550M with 1 GB video RAM.

Further a window of size 31×31 pixels was applied for the experiment and the other parameters were set to $\{\gamma, \alpha, \tau_{col}, \tau_{grad}\} := \{10, 0.9, 30, 4\}$.

Extended search space

This section describes the results of PatchMatch stereo with extended search space. It is described criterion by criterion.

Average costs Running the algorithms with the parConfs led to results for the average costs as visualized in table 5.2. The average costs (which is the arithmetic mean over the median costs from all test sets) are getting lower with a higher *lines* value. In the *unrect* scenario this has even more impact. The CoV shows that the deviation of the costs over all images is approximately the same. That means that the distribution function does not change in shape only in height and thus the adjustments have a similar impact on each test set.

parConf	median	change*	CoV
origPMS	332.74	-	47.79%
l5p0	320.40	-3.71%	47.95%
l10p0	312.03	-6.22%	48.33%
l5p0.01	322.25	-3.15%	47.95%
l10p0.01	316.19	-4.97%	48.25%

(a) *unrect* scenario

parConf	median	change*	CoV
origPMS	322.56	-	58.07%
l5p0	315.88	-2.07%	58.02%
l10p0	312.32	-3.17%	57.60%
l5p0.01	316.91	-1.75%	58.37%
l10p0.01	314.32	-2.56%	57.95%

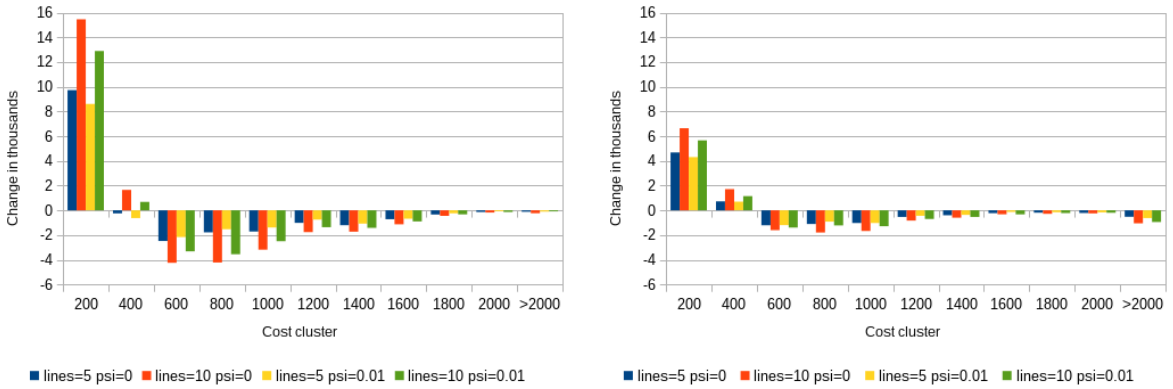
(b) *rightRect* scenario

*percental change to origPMS

Table 5.2: The results of the median criteria.

Furthermore it can be seen, that the average costs with $\psi = 0.01$ are worse than their counterparts with $\psi = 0$. This is expected due to different cost functions for calculating the planes, which was described in section 5.4.2. This makes the average costs not suitable for a decision whether the term Ψ is useful or not. The results allow the assumption that the extended search space works on both scenarios but works better in the *unrect* one.

Distribution of costs The distribution of costs strengthens the assumptions of the average costs. As can be seen in figure 5.15 the lower cost cluster are increasing and the higher cost clusters are decreasing in either case. This shows that the algorithm indeed lowers the costs and that the higher *lines* is the lower the costs are, even with $\psi = 0.01$. As mentioned above the adjustments work even better in the *unrect* scenario which is expected, because in the rectified case the images are already row aligned where the adjustment should not have too much impact on the result.



(a) *unrect* scenario **(b)** *rightRect* scenario

Figure 5.15: Average change over all test sets in each cost cluster referring to *origPMS*

Invalidated points Table 5.3 visualizes the invPts depending on *lines*. They are lower, if *lines* is higher. The "more inv." value states the percentage of tests sets per parConf that have more invPts than the *origPMS* parConf. In table 5.3a the "more inv." column is omitted because only one test set in *l10p0.01* had more invPts than *origPMS*.

parConf	invPts	change*
origPMS	456438	-
l5p0	391525	-14.22%
l10p0	357395	-21.70%
l5p0.01	403100	-11.69%
l10p0.01	384718	-15.71%

(a) *unrect* scenario

parConf	invPts	change*	more inv.**
origPMS	334318	-	-
l5p0	324141	-3.04%	26.67%
l10p0	327802	-1.95%	46.67%
l5p0.01	322093	-3.66%	6.67%
l10p0.01	322456	-3.55%	6.67%

(b) *rightRect* scenario

*percental change to origPMS **percentage of test sets with more invalids than origPMS

Table 5.3: The results of the invalidated points criteria.

Interestingly in the *rightRect* scenario the parConfs with $lines = 10$ perform worse than the parConfs with $lines = 5$ because a higher search space, in an image where the rows are already aligned, allows more incorrect planes to be taken. Additionally the table shows that the term Ψ does perform better and reduces the "more inv." value in the *rightRect* scenario because the algorithm is restrained to not chose planes with too high y-disparity and thus the planes are more correct.

Another discovery is that the adjustments work significant better in the *unrect* scenario in this criteria too. Furthermore the Ψ term worsens the results in this scenario, because in an unrectified image the correct plane is more likely to lie on a different scan line and restraining those gives worse results. This allows the assumption that additional cost term Ψ is useful with rectified images, while it does not perform so well with unrectified ones.

Runtime Table 5.4 illustrates runtimes of the various scenarios. Each line shows the normalized runtime in relation to *origPMS*. It can be seen that runtime depends heavily on *lines*. If the parameter is doubled the change compared to *origPMS* is also approximately doubled which is due to the doubled cost calculations in the view propagation step. Furthermore table 5.4 shows that the algorithm terminates faster on unrectified images and that the term Ψ has also a slight impact on them.

parConf	runtime	normalized*
origPMS	358s	1.00
l5p0	504s	1.41
l10p0	652s	1.82
l5p0.01	484s	1.35
l10p0.01	615s	1.72

(a) *unrect* scenario

parConf	runtime	normalized*
origPMS	392s	1.00
l5p0	551s	1.41
l10p0	727s	1.85
l5p0.01	601s	1.53
l10p0.01	739s	1.88

(b) *rightRect* scenario

*runtime normalized to *origPMS*

Table 5.4: The results of the runtime criteria.

Rank In this highly subjective criterion the adjustments have an huge impact on the results of the *unrect* scenario as can be seen in figure 5.16a. The figure visualizes for each scenario the amount of test sets in each rank. It can be seen that at least approximately one third of the test sets fail for each parConf. In the *origPMS* parConf the algorithm produces even more often bad results. Much better results are achieved with the *l10p0* parConf. Given that the *best* parConf is in most cases *l10p0* as seen in table 5.5a. In this case the term Ψ does not work well and sometimes even worsens the result.

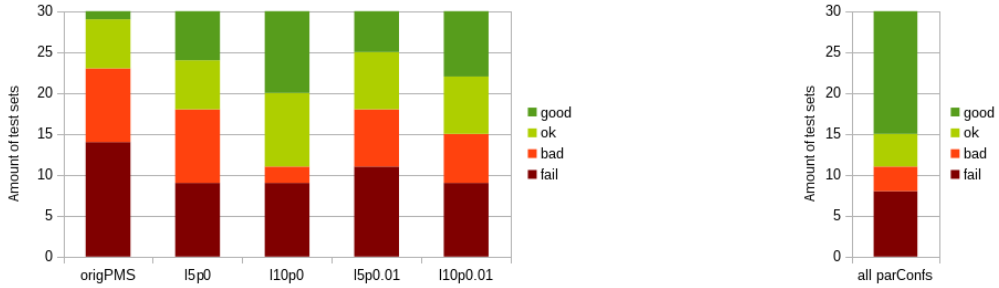
parConf	best
origPMS	0
l5p0	1
l10p0	**16**
l5p0.01	1
l10p0.01	3

(a) *unrect* scenario

parConf	best
origPMS	2
l5p0	3
l10p0	6
l5p0.01	**8**
l10p0.01	3

(b) *rightRect* scenario

Table 5.5: The best parConfs.



(a) *unrect* scenario **(b)** *rightRect* scenario

Figure 5.16: The amount of test sets in the different ranks.

In the *rightRect* scenario most of the results are ranked at least *ok*. In figure 5.16b the ranks over all test sets for this scenario are shown. The figure states only "all parConfs" because most parConfs among one test set are ranked the same. The reason is that the results look very similar and only slight differences occur. Hence it is hard to decide which parConf performs *best*. The figure also shows that a few test scenarios could not be used with PatchMatch Stereo and completely *fail*.

Table 5.5b states that the *l5p0.01* parConf works best but as can be seen the *l10p0* parConf is nearby and may even be better due to the subjective method. This states that the adjustments indeed work but the term Ψ need to be revised or removed.

The results of this criterion strengthens the previous expectations that PatchMatch Stereo with extended search space performs better on unrectified images and has little impact on rectified ones. Furthermore the additional cost term Ψ does not work well in the *unrect* scenario, while it may work in the *rightRect* one. Additionally this proves that PatchMatch Stereo works with freehand stereo images but not accurate for all images.

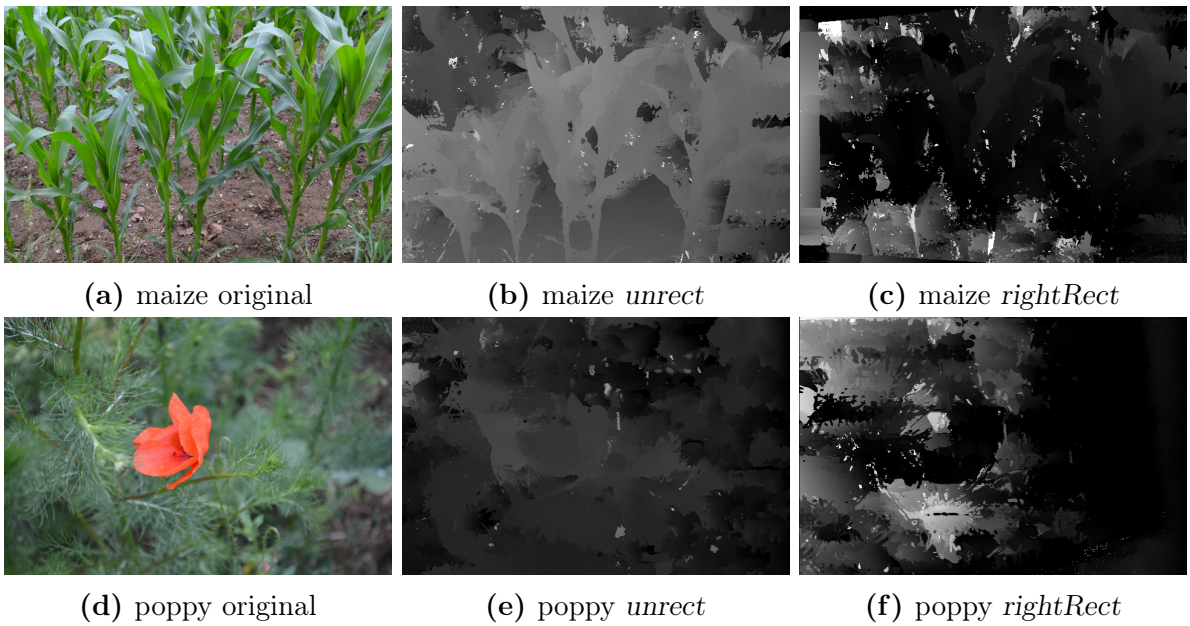
The detailed results for each test set for this criterion can be found in appendix B.

Special test sets Some test sets show special behavior which are described in the following. In test sets such as *dresden* and *farBuilding* the relevant information are really far away and the foreground changes much. Hence the disparity maps looks distorted as seen in figure 5.17.

(a) left image **(b)** right image **(c)** right image (*unrect l5p0*)

Figure 5.17: The images of the dresden test set and a disparity map

The test sets *maize* and *poppy* work much better in the *unrect* scenario than in the *rightRect* one. Those are visualized in figure 5.18. That means that in some cases the rectification worsens the disparity calculation.

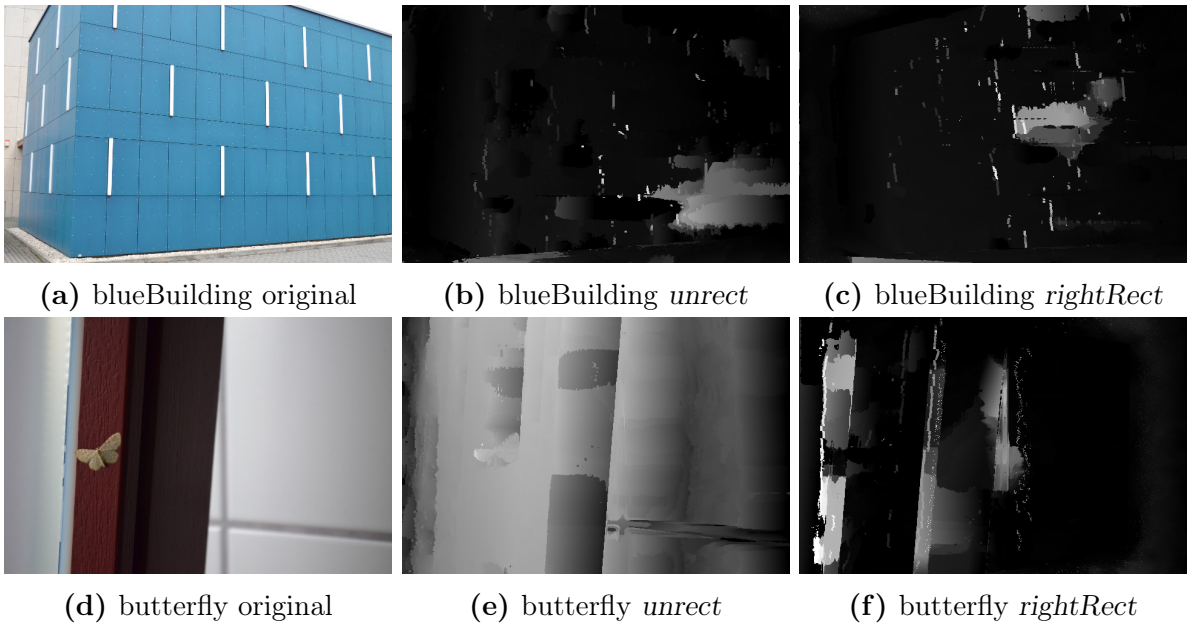


(a) maize original **(b)** maize *unrect* **(c)** maize *rightRect*

(d) poppy original **(e)** poppy *unrect* **(f)** poppy *rightRect*

Figure 5.18: The right image of the maize (a,b,c) and the poppy (d,e,f) test set (*l5p0.01*)

Other test sets such as *blueBuilding* and *butterfly* suffer form huge areas with similar texture where PatchMatch Stereo does not find correct matchings (see figure 5.19).

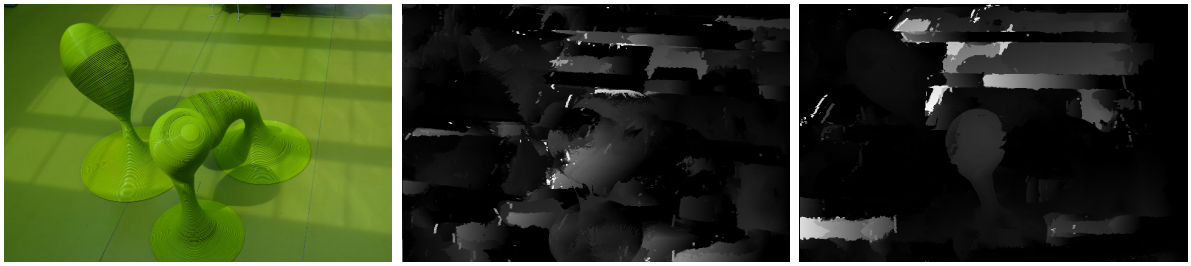


(a) blueBuilding original **(b)** blueBuilding *unrect* **(c)** blueBuilding *rightRect*

(d) butterfly original **(e)** butterfly *unrect* **(f)** butterfly *rightRect*

Figure 5.19: The right image of the blueBuilding (a,b,c) and butterfly (d,e,f) test set (*l5p0.01*)

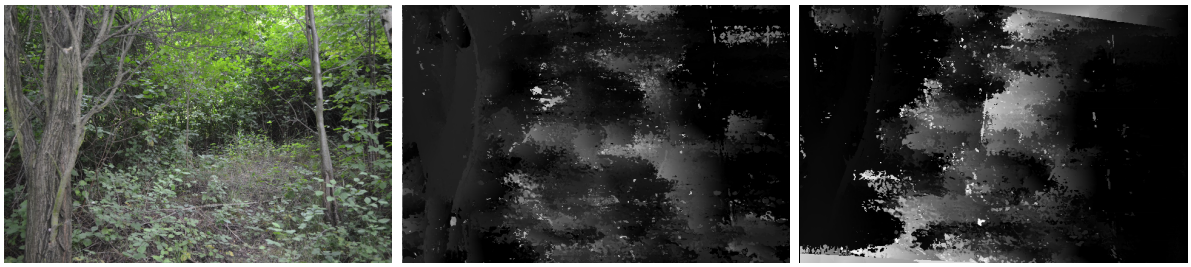
The *faculty* test set was furthermore taken from above and has difficult lightning conditions which does not work well as seen in figure 5.20. In this case background is pushed forward due to the shadows.



(a) original **(b)** *unrect* **(c)** *rightRect*

Figure 5.20: The right image of the faculty test set (*l5p0.01*)

In the *forest* test set as seen in figure 5.21 the tree on the left is recognized well but it is put into the background while the other parts are pushed into the foreground. This should be the other way round. This could mean that the algorithm has problems with high depth changes on a small area.



(a) original **(b)** *unrect* **(c)** *rightRect*

Figure 5.21: The right image of the forest test set (*l5p0.01*)

Summary The evaluation showed that PatchMatch Stereo works with most images. The algorithm can be improved by extending the search space. This is especially in the *unrect* scenario, but slows down the algorithm by a considerable amount. The term Ψ improves results of *rightRect* scenario slightly, while it worsens the results in the *unrect* scenario. Hence Ψ should be revised. It was furthermore showed that in some cases the rectification procedure reduces the quality of the results. Additionally PatchMatch Stereo has problems with difficult lightning conditions, too much depth changes in a small area and huge areas with homogeneous texture.

OpenCL results

The OpenCL implementation has a memory allocation problem if the whole image is processed because to much data needs to be allocated beforehand which does not fit into the RAM. In order to overcome this problem the algorithm is only used with the row-by-row approach which reduces the memory allocation to only one row instead of the whole image.

This approach runs without any further adjustments up to two times faster than the CPU implementation as seen in table 5.6. These results also show that the runtime does benefit more in the *rightRect* scenario. Interestingly the higher the *lines* parameter is the faster is the OpenCL implementation compared to the CPU one except for the *origPMS* parConf.

parConf	CPU	OpenCL	norm.*
origPMS	358s	167s	0.47
l5p0	504s	406s	0.81
l10p0	652s	488s	0.75
l5p0.01	484s	452s	0.93
l10p0.01	615s	543s	0.88

(a) *unrect* scenario

parConf	CPU	OpenCL	norm.*
origPMS	392s	190s	0.48
l5p0	551s	421s	0.76
l10p0	727s	469s	0.64
l5p0.01	601s	391s	0.65
l10p0.01	739s	436s	0.59

(b) *rightRect* scenario

*OpenCL runtime normalized to CPU one

Table 5.6: Changes in the runtime criteria.

In table 5.7 the percentage of change referring to the CPU implementation for the median and the invalidated points criteria is shown. This illustrates that the results of the proposed OpenCL implementation are worse than that of the CPU one.

parConf	median	invPts
origPMS	-0.76%	46.54%
l5p0	1.31%	80.55%
l10p0	2.06%	74.91%
l5p0.01	1.01%	93.42%
l10p0.01	1.51%	88.30%

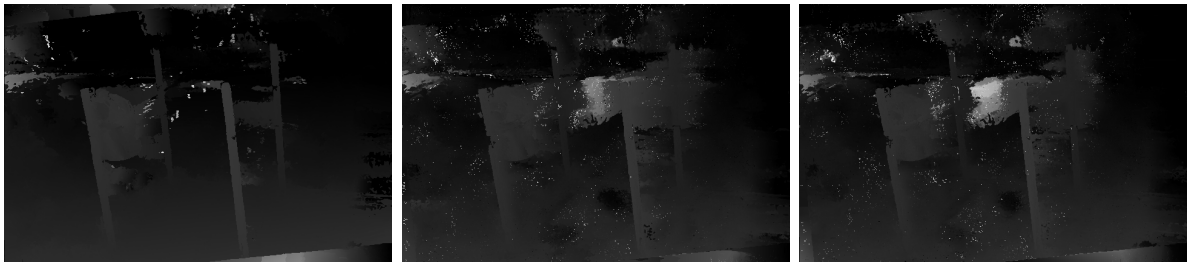
(a) *unrect* scenario

parConf	median	invPts
origPMS	7.08%	48.47%
l5p0	7.33%	76.38%
l10p0	6.95%	64.42%
l5p0.01	7.44%	65.14%
l10p0.01	7.51%	58.91%

(b) *rightRect* scenario

Table 5.7: Changes in the median and invalidated points criteria.

While in the *unrect* scenario the median does not change much it does in the *rightRect* scenario. The values of invalidated points are even worse. The disparity maps look more distorted than that of the CPU implementation in every parConf as seen in figure 5.22.



(a) CPU (*l5p0*). **(b)** OpenCL (*l5p0*). **(c)** OpenCL (*l10p0.01*).

Figure 5.22: The right disparity map of the poles test set with CPU implementation and OpenCL one.

In order to evaluate whether the problem can be reduced considering more points during spatial propagation (spatial offset), described in section 5.2.2, an experiment was performed. Therefore five test sets (*bridge,koenigstein,pole,poles,sewerCover*) were chosen and calculated with considering five more spatial points in the x-direction. These test sets were run with the parConfs: *origPMS*, *l5p0* and *l5p0.01*. To calculate the results illustrated in table 5.8 the arithmetic mean over the given five test sets was used. In table 5.8a the change compared to the OpenCL approach is illustrated, while in table 5.8b the changed compared to the CPU implementation is used.

parConf	**median**	**invPts**	**runtime**	**norm.***
origPMS	-4.21%	-3.60%	245s	1.22
l5p0	-2.91%	-5.92%	640s	1.49
l5p0.01	-3.67%	-3.54%	445s	1.11

(a) Change compared to initial OpenCL

parConf	**median**	**invPts**	**norm.****
origPMS	19.24%	148.11%	0.60
l5p0	12.48%	119.33%	1.08
l5p0.01	13.02%	115.75%	0.70

(b) Change compared to CPU

*runtime normalized to initial OpenCL approach **runtime normalized to CPU implementation

Table 5.8: The change compared to the initial OpenCL approach and to the CPU implementation.

Table 5.8a states that the use of a spatial offset does improve the quality of the OpenCL implementation but also makes it slower. Further table 5.8b shows that it still performs worse than the CPU implementation. This is illustrated in figure 5.23.

(a) CPU implementation **(b)** OpenCL no spatial offset **(c)** OpenCL with spatial offset

Figure 5.23: The right image of the pole test set (*l5p0*) calculated with different methods.

These experiments show that with the proposed methods the issues of the OpenCL implementation could not be removed. The OpenCL implementation is not entirely worse because disparity maps look similar and test sets that failed on the CPU implementation get better results with the OpenCL one but still fail.

Conclusion

The experiment showed that PatchMatch Stereo works good with rectified freehand stereo images and that the proposed adjustments do give slight improvements on the results. In unrectified images the adjustments have an huge impact and perform well which allows the assumption that these will also contribute to images where the rectification is inaccurate. For the given unrectified stereo images a good parConf was found which is $lines = 10$ and $\psi = 0$. The experiment further showed that for rectified images $lines = 5$ and $\psi = 0.01$ seems a good parConf while the additional cost term Ψ needs to be revised. The adjustments slow down the algorithm in a considerable amount and even the proposed OpenCL implementation could not contribute to the runtime due to its worse quality.

5.5 Summary

In this section the PatchMatch Stereo algorithm was described and some performance improvements were presented. Afterwards an evaluation approach for these improvements was envisioned. The evaluation was performed to see whether the improvements are useful or not. In the end it was found out that the quality improvements work and the presented speed improvements did not led to the expected results.

6 Conclusion

This chapter summarizes the knowledge gained from the previous chapters and sections. Further it will sketch ideas for further improvements.

6.1 Conclusion

In this thesis algorithms for rectifying images were described and compared. Therefore an evaluation approach was defined and used. An algorithm for the rectification procedure was chosen which is Hartley's algorithm in conjunction with the QER fundamental Matrix.

Performance improvements for the PatchMatch Stereo algorithm were introduced. One is the extended search space and the other is an OpenCL impelementation.

Extended search space The extended search space allows the algorithm to work on images where the epipolar geometry is not correct. Furthermore the cost function for PatchMatch Stereo was adjusted with an additional term Ψ to compensate the introduced y-disparity.

Some criteria could be defined to evaluate those improvements for stereo images with no ground truth available. These criteria are the average costs, the cost distribution, the invalidated points, the runtime and the ranking of the results.

Experiments show that the extended search space improves the results especially for unrectified images. They also show that PatchMatch Stereo does not work optimally on all images and has problems with difficult lightning conditions and too homogeneous surfaces. Furthermore the algorithm is slowed down by the extension. The term Ψ does not work well and in some cases even reduces the quality of the results.

OpenCL implementation Furthermore an OpenCL implementation was created which speeds up the algorithm but suffers from worse results because of the problem that the spatial propagation is sequential and is thus not suited for an parallelization approach.

Overall the depth calculation is enhanced in the "freehand stereo" case by using PatchMatch Stereo with the extended search space.

6.2 Future work

For the evaluation of rectification algorithms the ARE criteria should be revised that it is independent of the stretching or compressing of an image. Additionally a more robust method should be considered. Maybe by transforming the images into another representation and do some calculations there.

In the PatchMatch Stereo algorithm with extended search space the additional cost term Ψ should be revised in a way that it does not worsen results. Maybe the cost function itself should be replaced by another one to better fit within the extended search space.

The algorithm itself may work better with an adaptive change of the *lines* parameter. This could be done by determining the average over all φ in an image or in the support window. Another idea is to allow the φ parameter to only change in the direction it was in the iteration before. This would shrink the search space and therefore speed up the algorithm.

Appendix

A Contents of the DVD

The thesis was made on a Linux operating system. The DVD contains:

- the references (*References/*)
- images that were self created (*Images/*)
- the program (*Program/PatchMatchStereo/*)
- a text file that describes the contents more complete and the usage of the program (*readme.txt*)
- the input test data (*Input/*)
- the statistics for the data (*Statistics/*). The statistics contain:
  - *rectification_stats.ods* contain the statistics for the rectification
  - *PMS_rect.ods* contain the statistics for the rightRect scenario
  - *PMS_unrect.ods* contains the statistics for the unrect scenario
- the results (*Data/*). The Data contains:
  - *data.7z* which contains the calculated disparity maps and neighbor images. These are labeled with respect to their used parameters:
    - *_unrect* means that its an unrectified image (rectified if not present)
    - *_l#* means the *lines* parameter (zero if not present)
    - *_c#* means the ψ value (zero if not present)
    - *_ocl* means that it was calculated using the OpenCL approach (CPU implementation if not defined)
    - *disp* refers to a disparity map
    - *neighbor* refers to the neighbor image

  - *planes* refers to the planes that were used for calculating the disparity map
  - *costHist* refers to the cost distribution histogram
- *rectify/* contains the rectification information:
  - *ASIFTnOut.7z, ASIFTwOut.7z, MANnOut.7z, MANwOut.7z* contain the results of the recitification
    - *bouget* refers to Bouget's algorithm
    - *hartleys* refers to Hartleys algorithm
    - *qer* refers to Quasi-Euclidean epipolar rectification (QER) (either rectification or fundamental matrix)
    - *pollefey* refers to Polar rectification
    - *_8point* refers to the 8-point-F-matrix
    - *_calib_5point* refers to the calib-F-matrix
    - *_rect* the rectified image
    - *_unrect* the unrectified image with epipolar lines.
    - *matches_inner* contain the inner (and all) matches
    - *matches_outer* shows the outer matches
    - *choosenRect* shows the chosen rectangle for outer matches
  - *asiftKeys and asiftMatchings* are the used matchings for asift

B Test stereo images

In the following the used thirty stereo test sets as well as the resulting images are presented. A page is structured as seen in figure B.1.

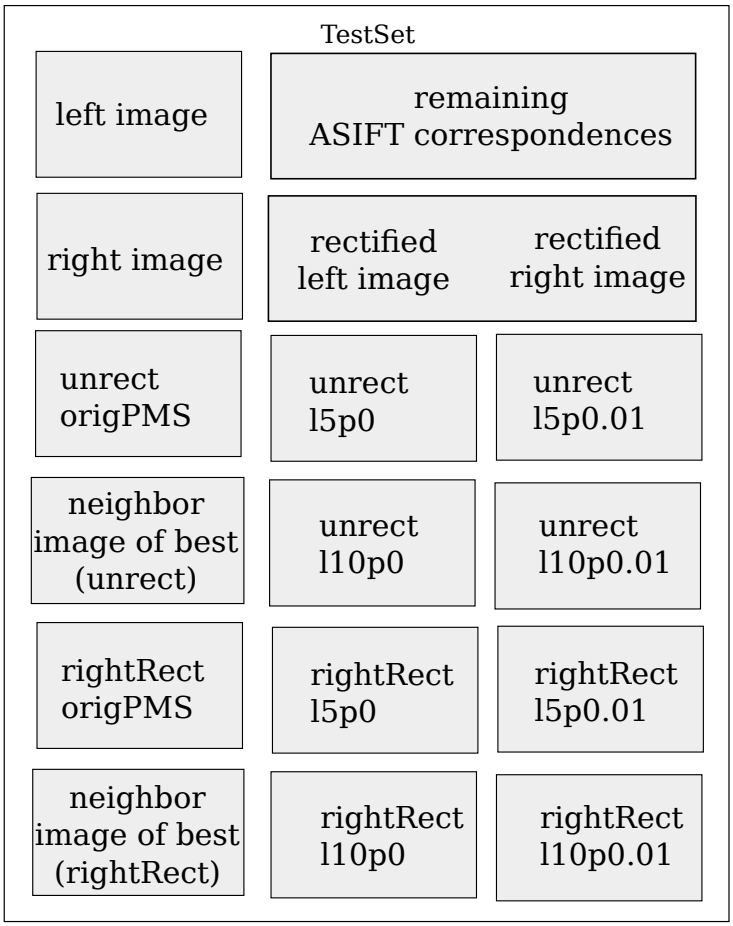


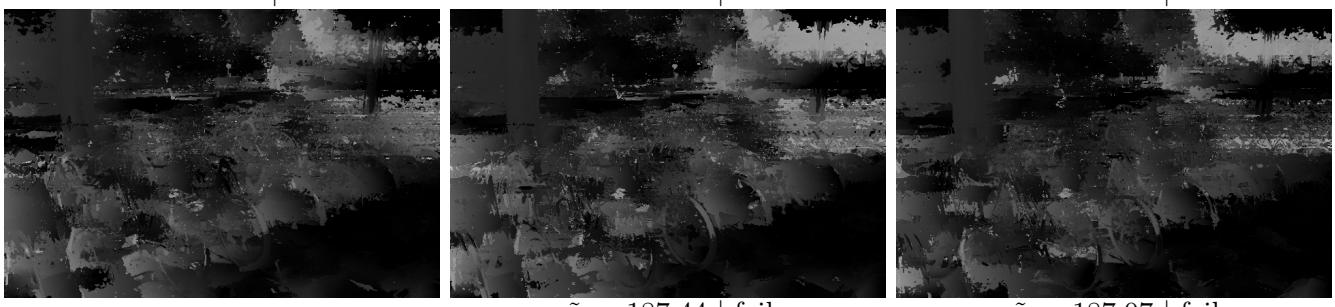
Figure B.1: The description of a page consisting of the tests sets.

The first two rows refer to the original image, the found asift matchings and the rectified images using Hartley's QER. The third and the fourth row represent the disparity maps of the *unrect* test scenario whereas the fifth and sixth row depict the disparity maps from the *rightRect* scenario. Note that always the right map is shown. The disparity maps are always in the following order: *1) origPMS*, *2) l5p0*, *3) l5p0.01*, *4)* the neighbor image of the *best* map (if present), *5) l10p0* and *6) l10p0.01*

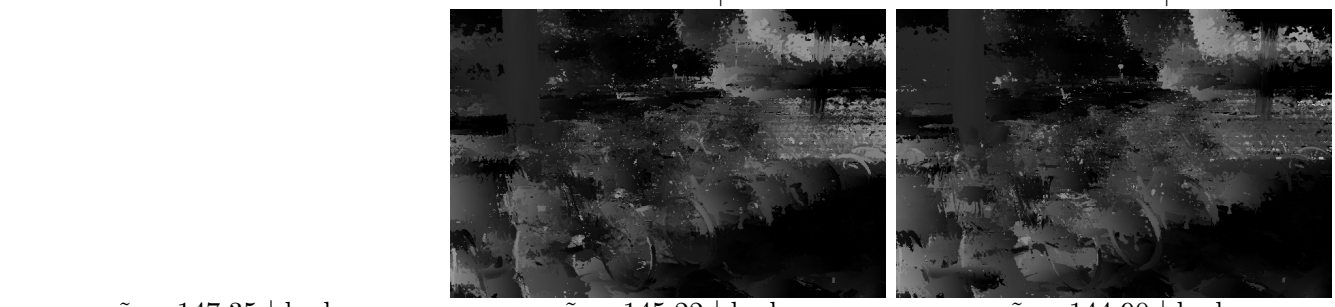
Additional information (e.g. the rank and the median $\tilde{m}$) is given above the images.

bicycles

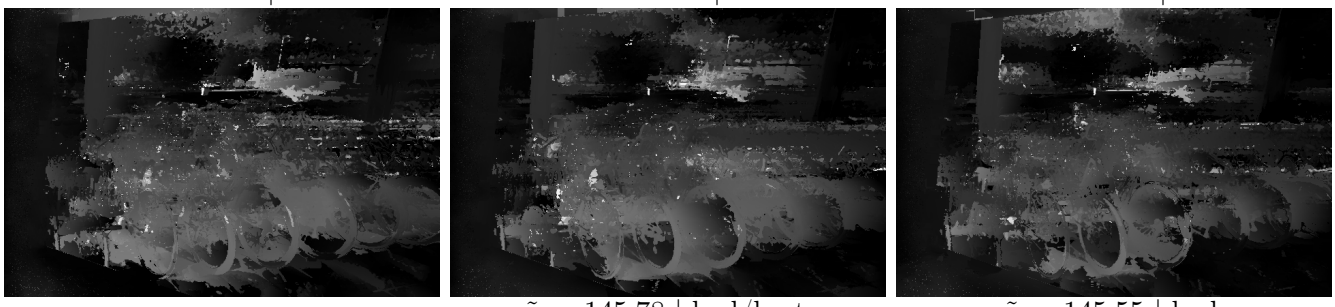
$\tilde{m} = 188.69$ | fail $\tilde{m} = 187.46$ | fail $\tilde{m} = 185.19$ | fail



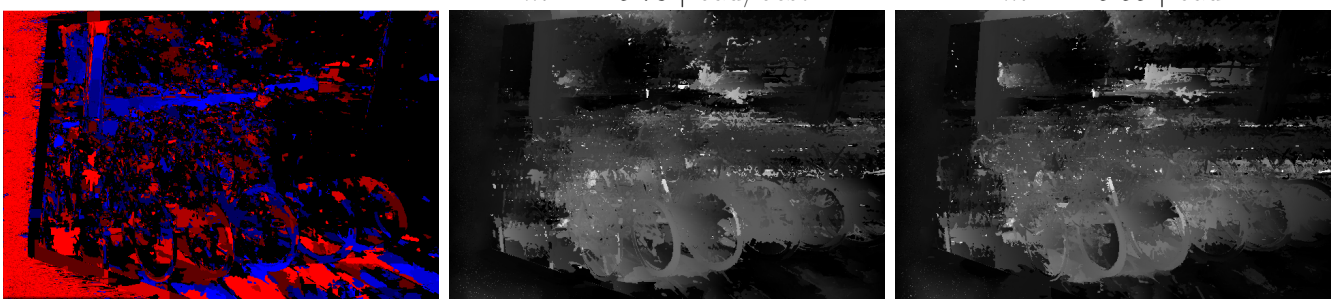
$\tilde{m} = 187.44$ | fail $\tilde{m} = 187.07$ | fail



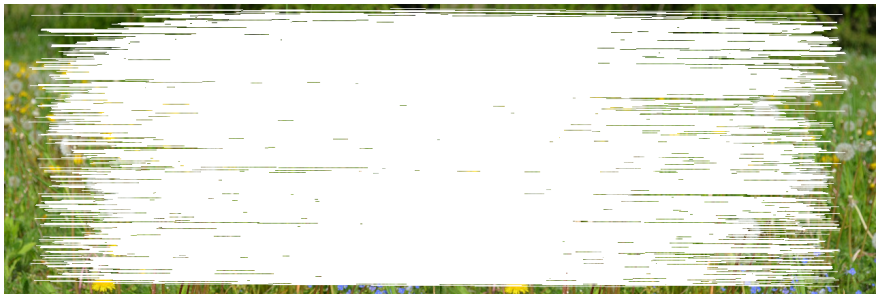
$\tilde{m} = 147.35$ | bad $\tilde{m} = 145.22$ | bad $\tilde{m} = 144.90$ | bad

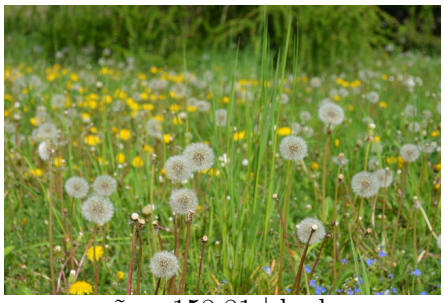


$\tilde{m} = 145.78$ | bad/best $\tilde{m} = 145.55$ | bad

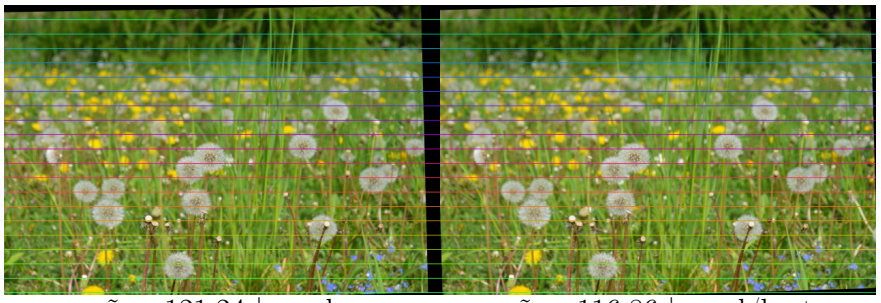


blowball



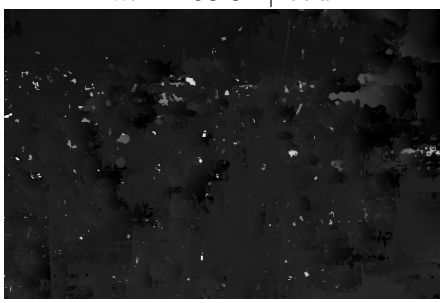


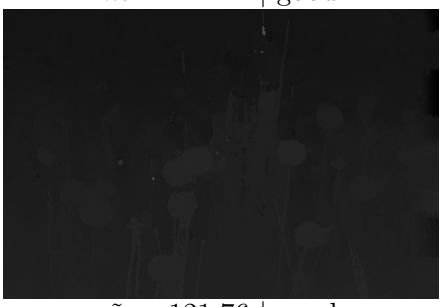
$\tilde{m} = 158.81$ | bad

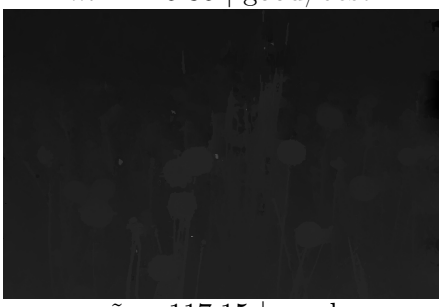


$\tilde{m} = 121.24$ | good

$\tilde{m} = 116.86$ | good/best

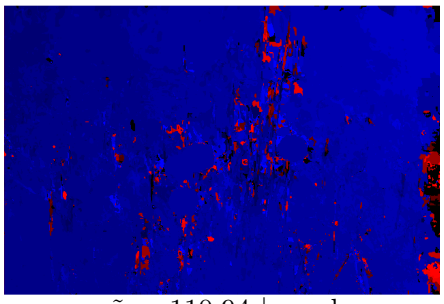


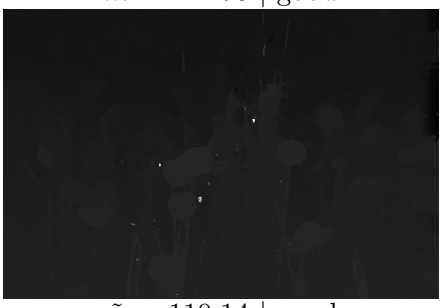


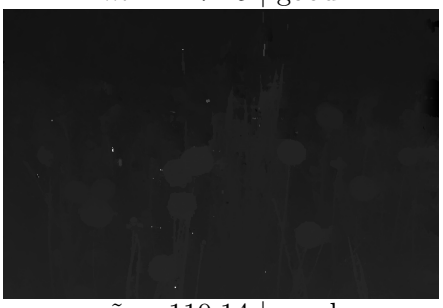


$\tilde{m} = 121.76$ | good

$\tilde{m} = 117.15$ | good



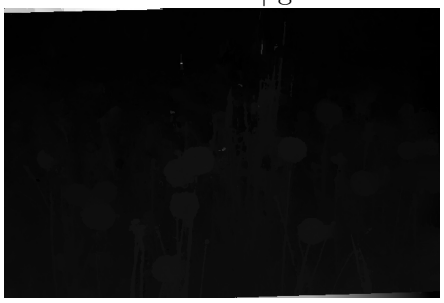


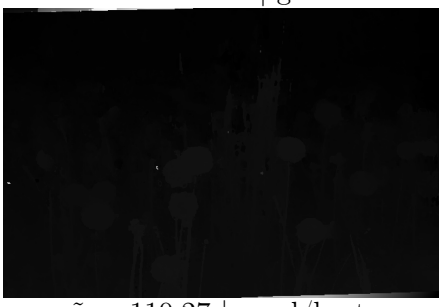


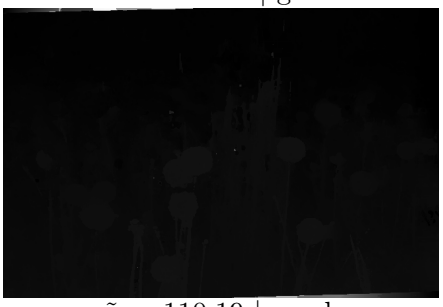
$\tilde{m} = 110.94$ | good

$\tilde{m} = 110.14$ | good

$\tilde{m} = 110.14$ | good

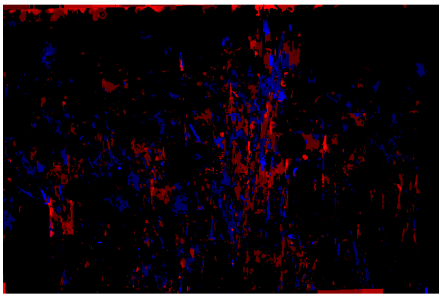


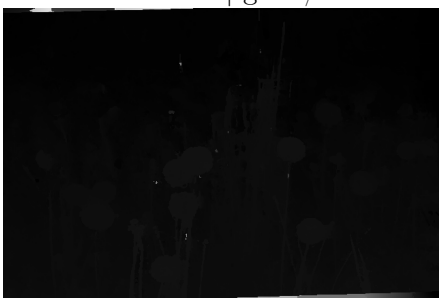




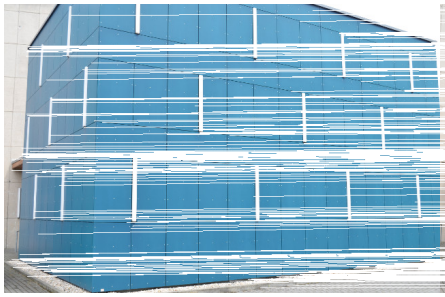
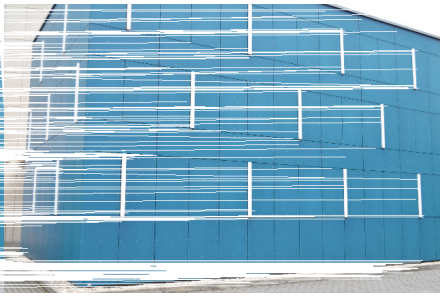
$\tilde{m} = 110.27$ | good/best

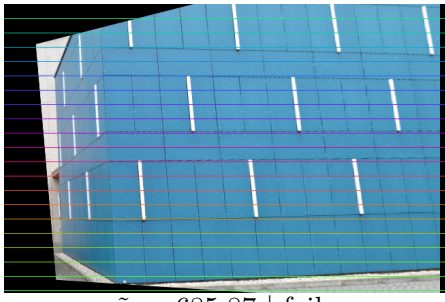
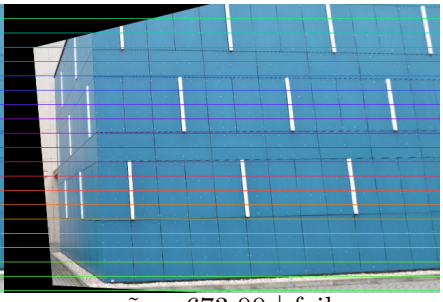
$\tilde{m} = 110.19$ | good



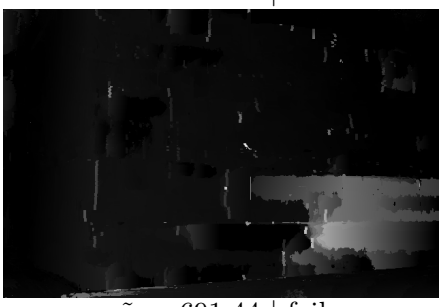
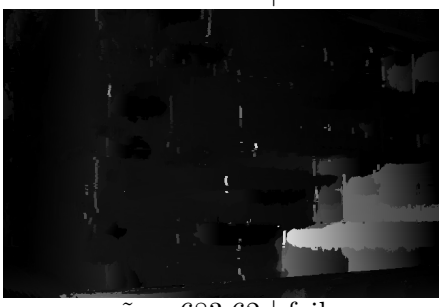


blueBuilding

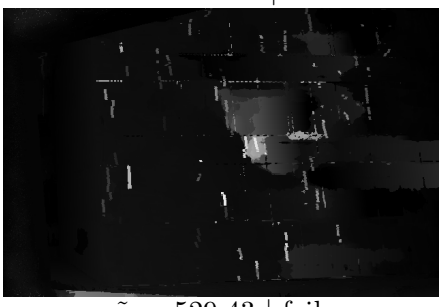
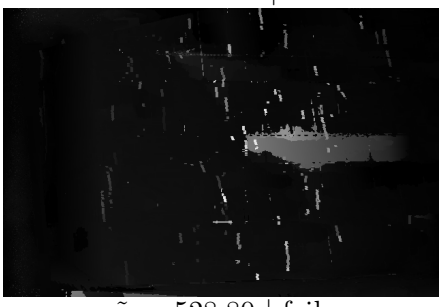




$\tilde{m} = 711.27$ | fail $\tilde{m} = 685.87$ | fail $\tilde{m} = 673.00$ | fail

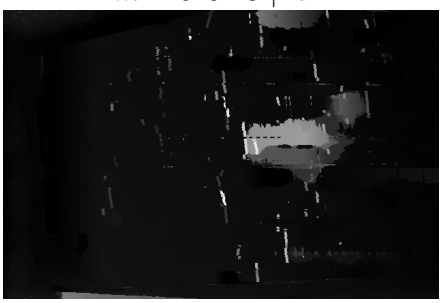
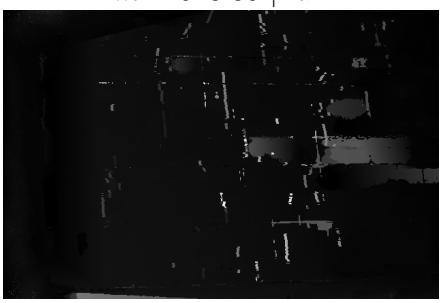




$\tilde{m} = 691.44$ | fail $\tilde{m} = 683.62$ | fail


$\tilde{m} = 537.60$ | fail $\tilde{m} = 527.04$ | fail $\tilde{m} = 523.57$ | fail

$\tilde{m} = 529.43$ | fail $\tilde{m} = 528.80$ | fail

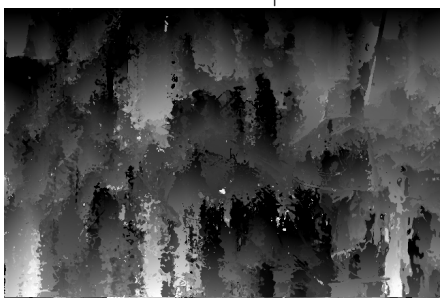
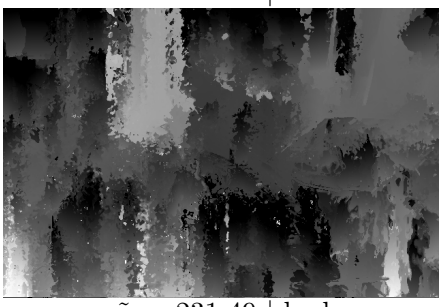
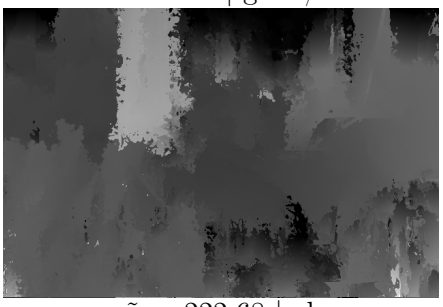



bridge (rotated)

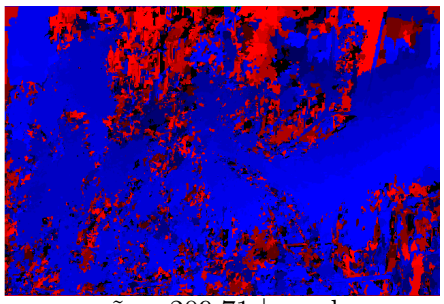
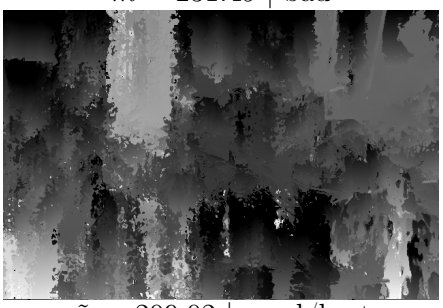
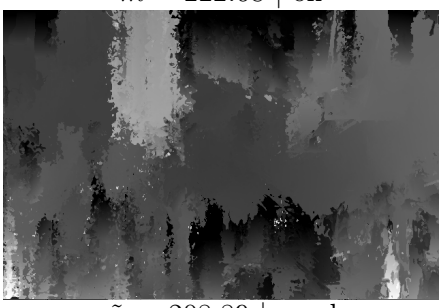




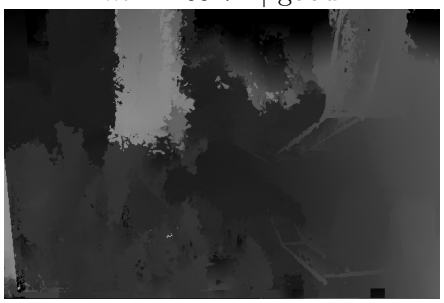
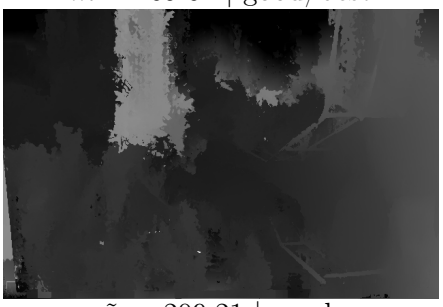
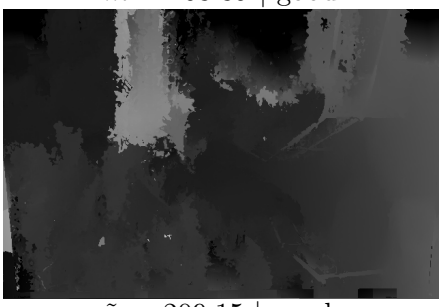
$\tilde{m} = 233.32$ | fail $\tilde{m} = 230.08$ | bad $\tilde{m} = 218.84$ | good/best

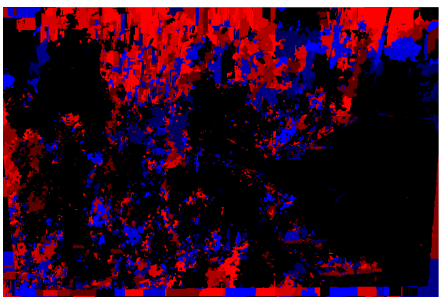
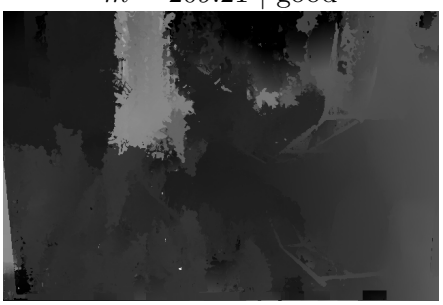
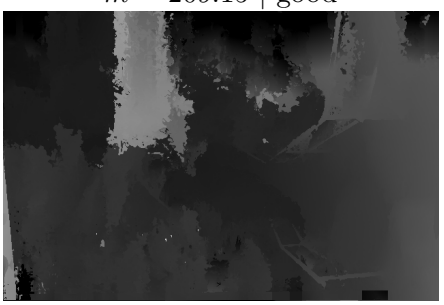
$\tilde{m} = 231.49$ | bad $\tilde{m} = 222.68$ | ok

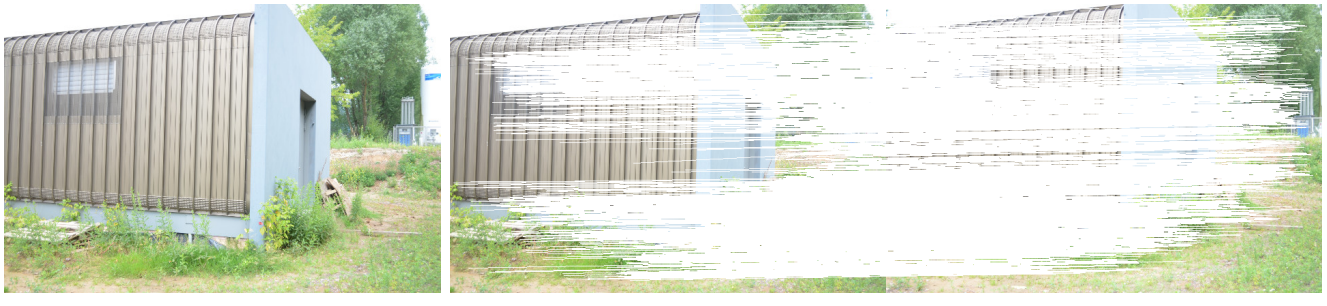
$\tilde{m} = 209.71$ | good $\tilde{m} = 209.02$ | good/best $\tilde{m} = 208.89$ | good

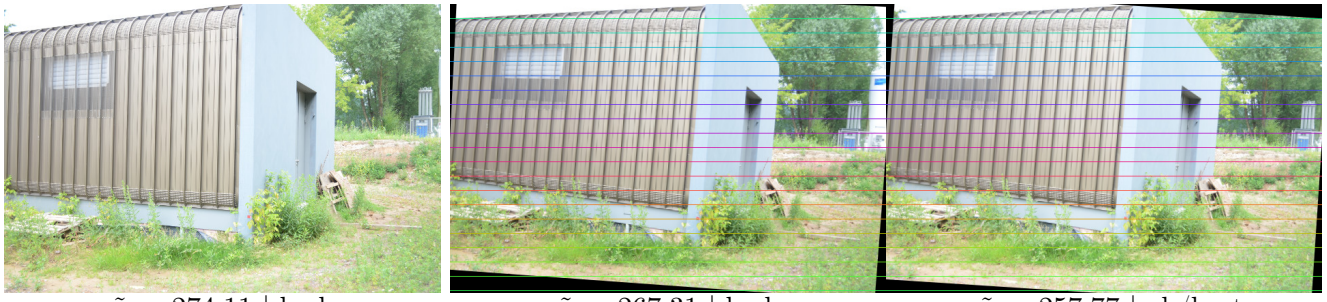




$\tilde{m} = 209.21$ | good $\tilde{m} = 209.15$ | good

brownBuilding





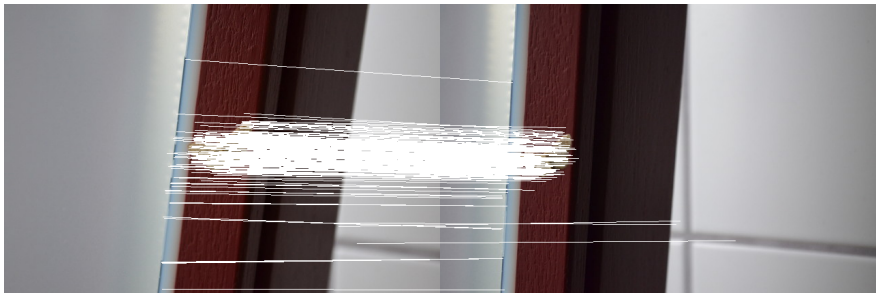
$\tilde{m} = 274.11$ | bad $\tilde{m} = 267.31$ | bad $\tilde{m} = 257.77$ | ok/best

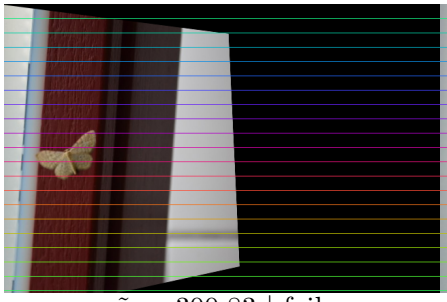
$\tilde{m} = 269.43$ | bad $\tilde{m} = 263.53$ | bad

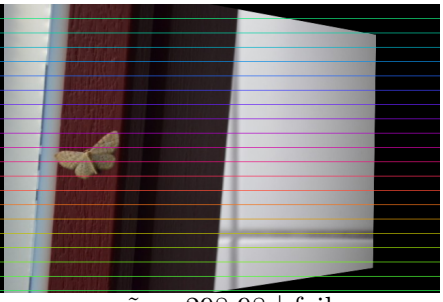
$\tilde{m} = 262.24$ | ok $\tilde{m} = 255.14$ | ok $\tilde{m} = 252.03$ | ok

$\tilde{m} = 256.12$ | ok $\tilde{m} = 254.13$ | ok/best

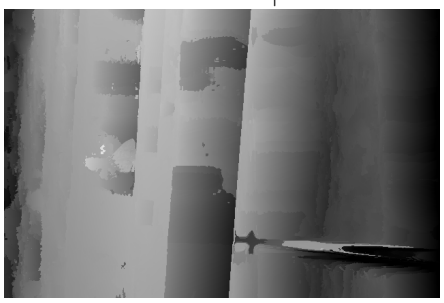
butterfly

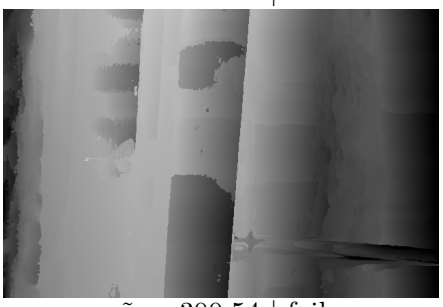


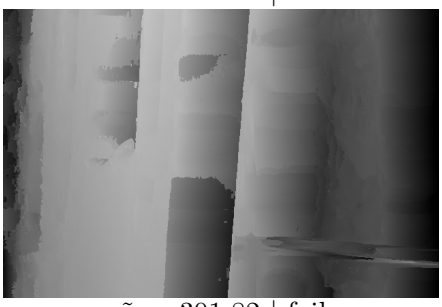




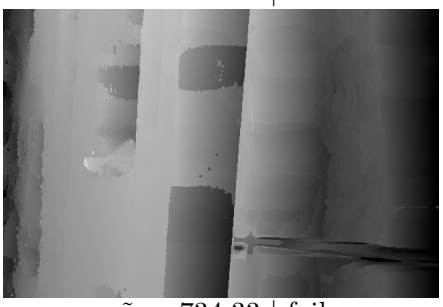
$\tilde{m} = 303.81$ | fail $\tilde{m} = 300.83$ | fail $\tilde{m} = 298.98$ | fail

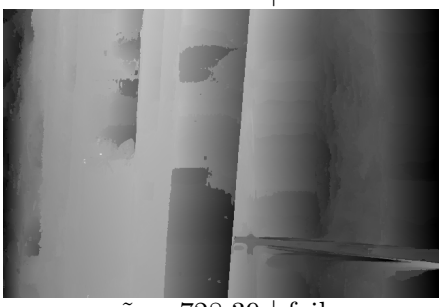




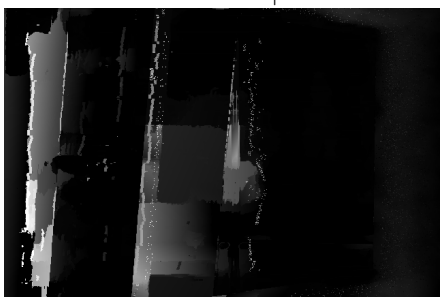


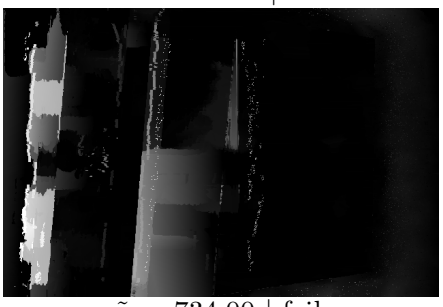
$\tilde{m} = 300.54$ | fail $\tilde{m} = 301.82$ | fail

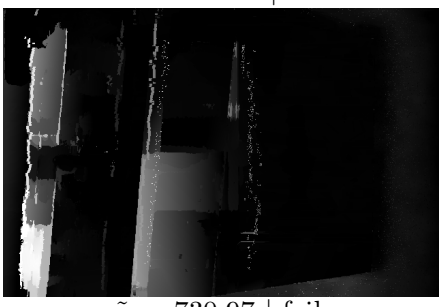




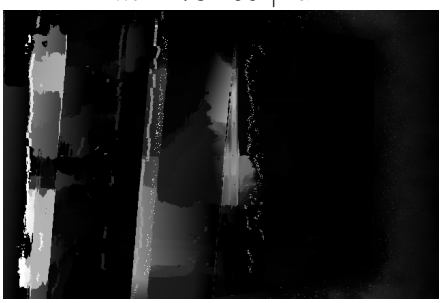
$\tilde{m} = 734.71$ | fail $\tilde{m} = 734.33$ | fail $\tilde{m} = 728.39$ | fail

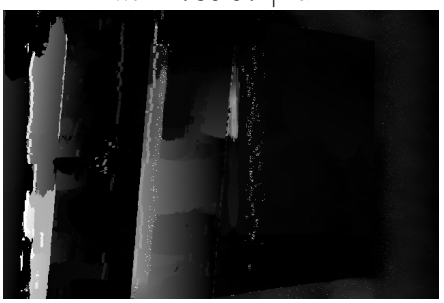






$\tilde{m} = 734.00$ | fail $\tilde{m} = 730.97$ | fail

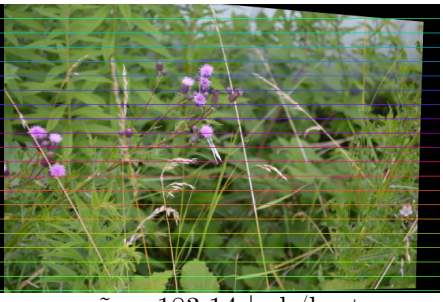




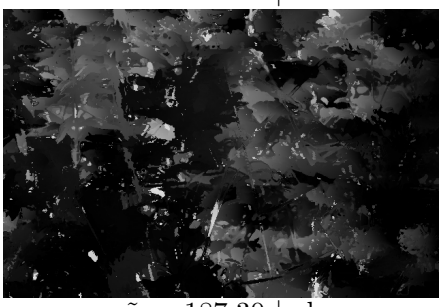
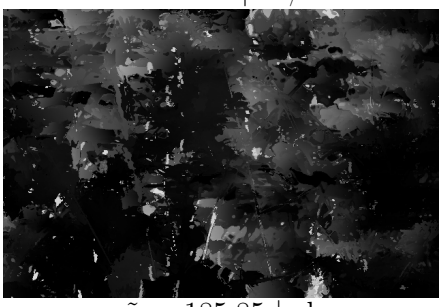
butterfly1



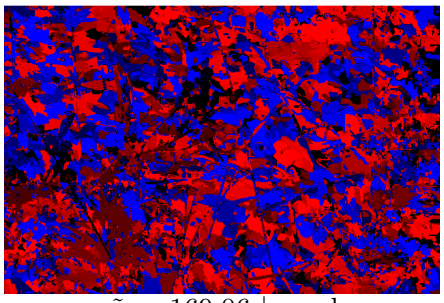
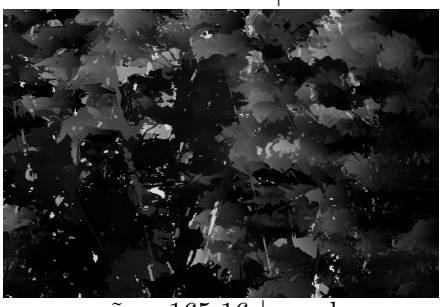
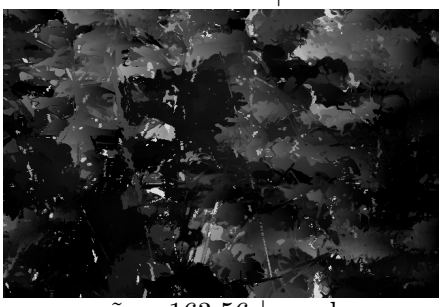




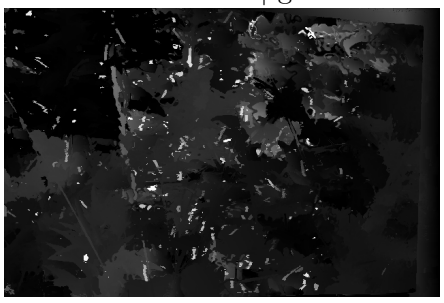
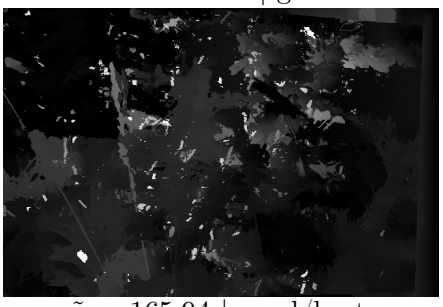
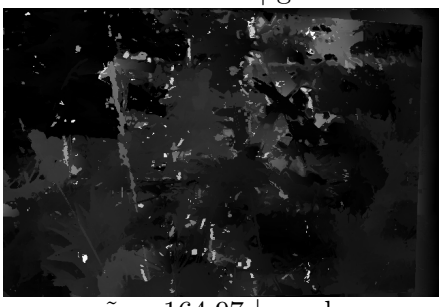
$\tilde{m} = 189.94$ | ok $\tilde{m} = 185.82$ | ok $\tilde{m} = 183.14$ | ok/best

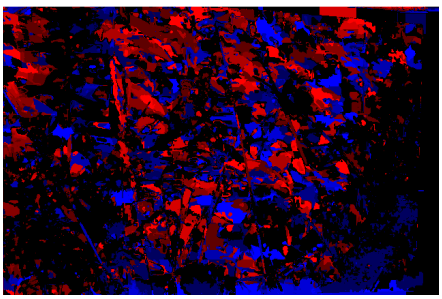
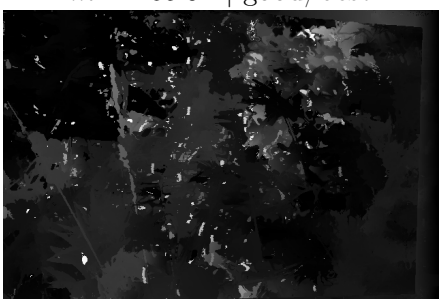
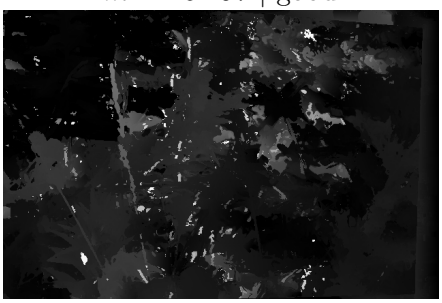
$\tilde{m} = 187.30$ | ok $\tilde{m} = 185.85$ | ok

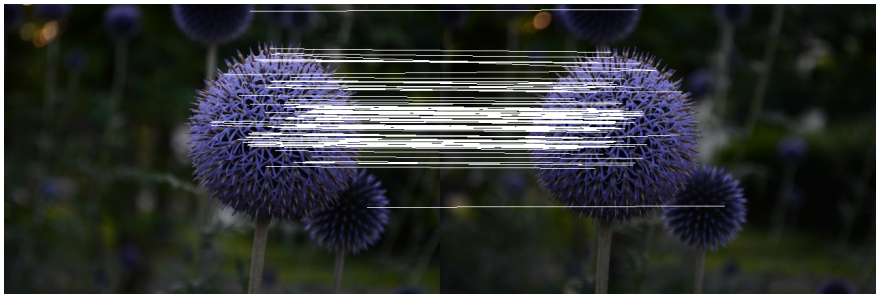
$\tilde{m} = 169.06$ | good $\tilde{m} = 165.16$ | good $\tilde{m} = 163.56$ | good

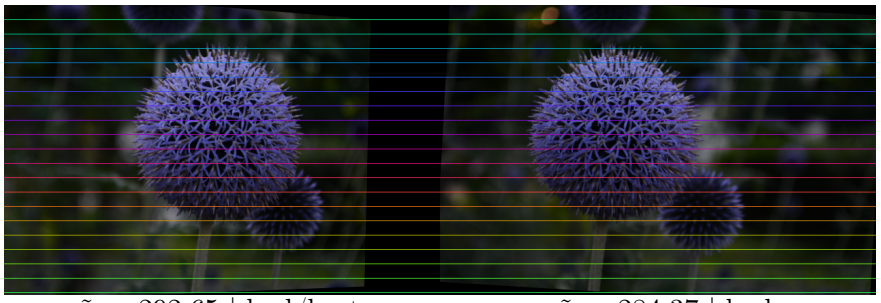




$\tilde{m} = 165.94$ | good/best $\tilde{m} = 164.97$ | good

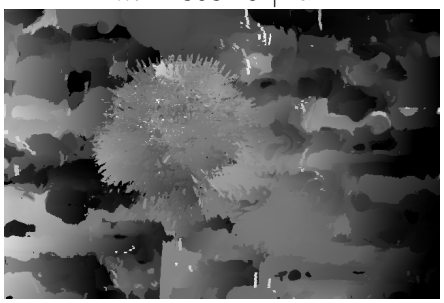
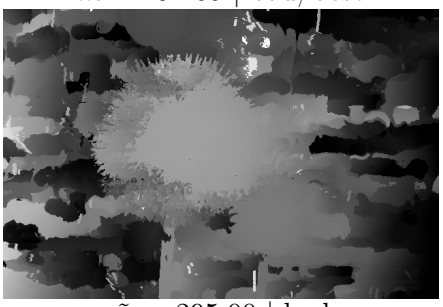
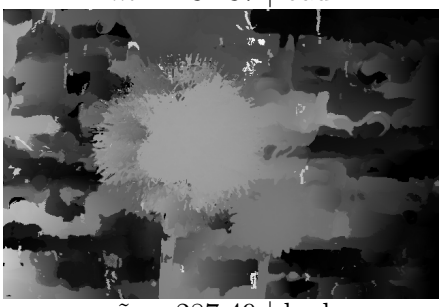




distel

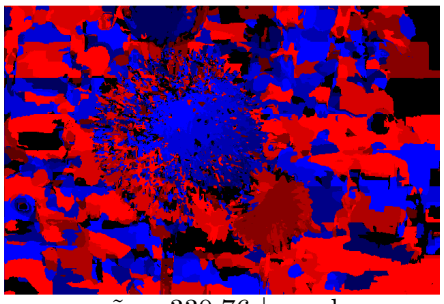
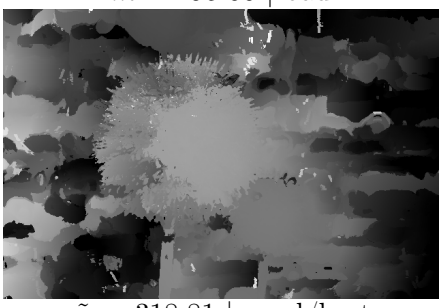
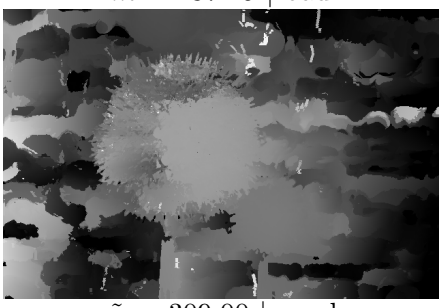





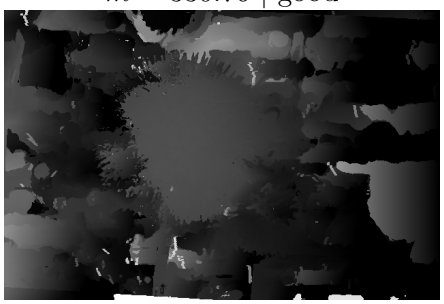
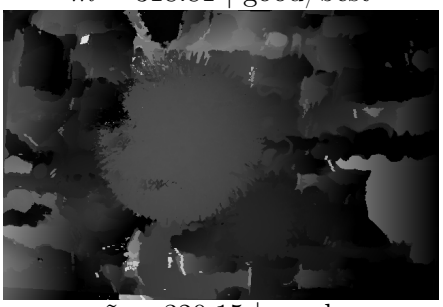
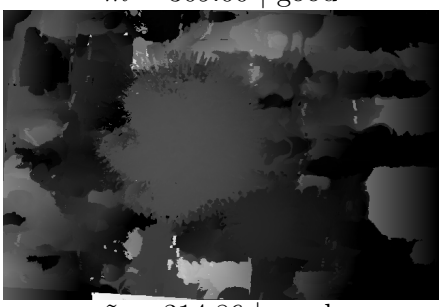
$\tilde{m} = 303.49$ | fail $\tilde{m} = 292.65$ | bad/best $\tilde{m} = 284.37$ | bad

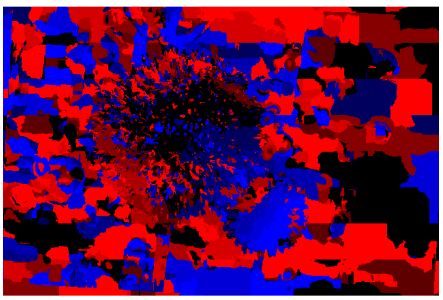
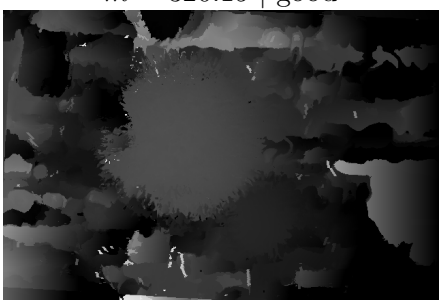
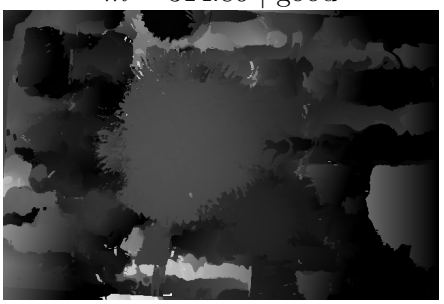
$\tilde{m} = 295.00$ | bad $\tilde{m} = 287.49$ | bad

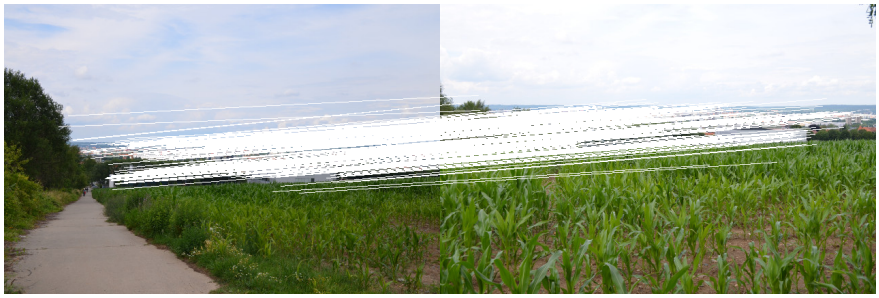
$\tilde{m} = 330.76$ | good $\tilde{m} = 318.81$ | good/best $\tilde{m} = 309.00$ | good

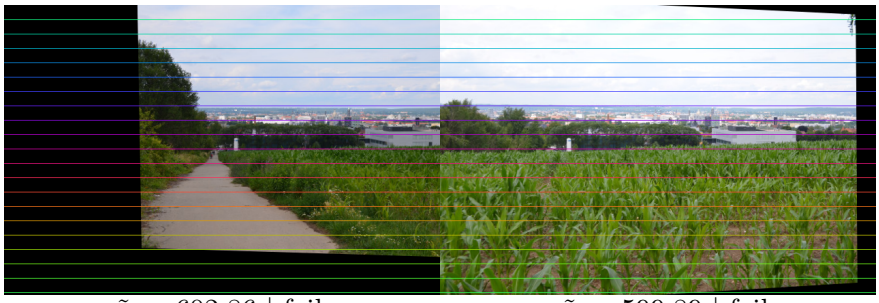




$\tilde{m} = 320.15$ | good $\tilde{m} = 314.86$ | good

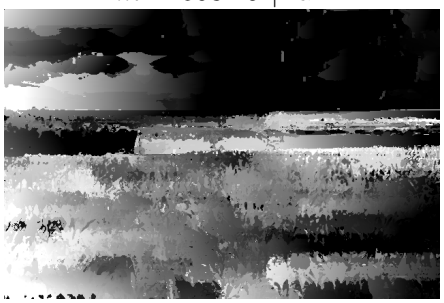
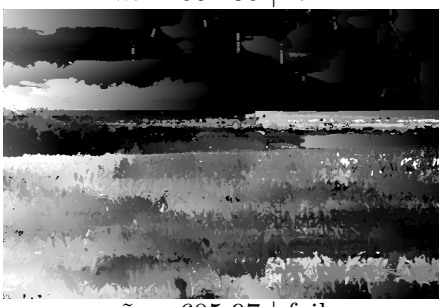
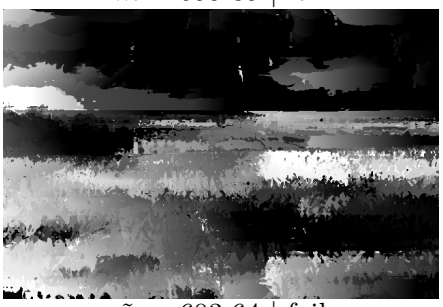




dresden

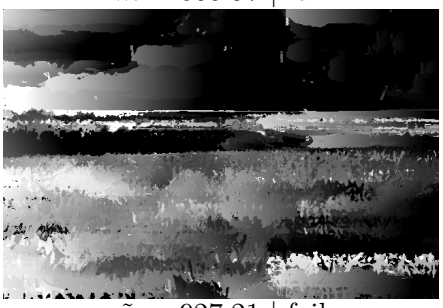



$\tilde{m} = 608.19$ | fail $\tilde{m} = 602.86$ | fail $\tilde{m} = 599.89$ | fail

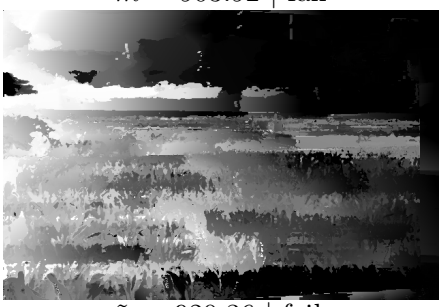




$\tilde{m} = 605.97$ | fail $\tilde{m} = 603.64$ | fail



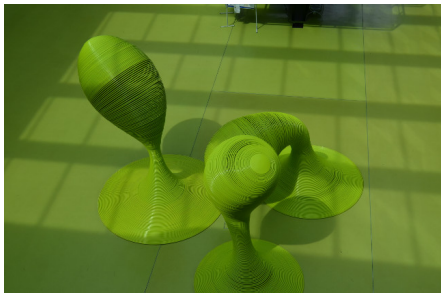

$\tilde{m} = 960.98$ | fail $\tilde{m} = 927.21$ | fail $\tilde{m} = 903.92$ | fail

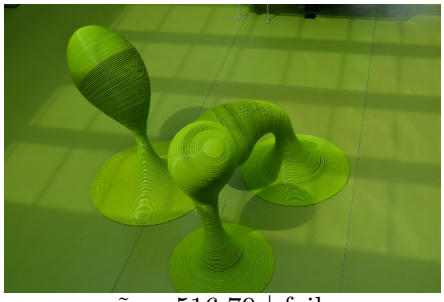


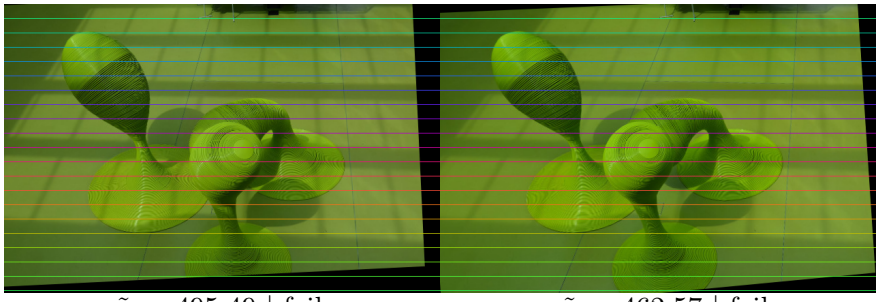


$\tilde{m} = 940.93$ | fail $\tilde{m} = 920.36$ | fail


faculty



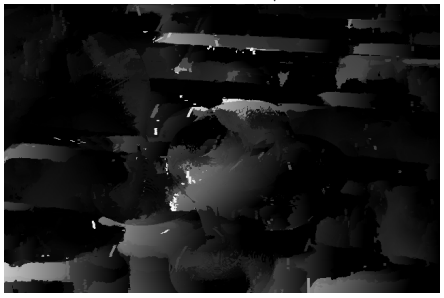


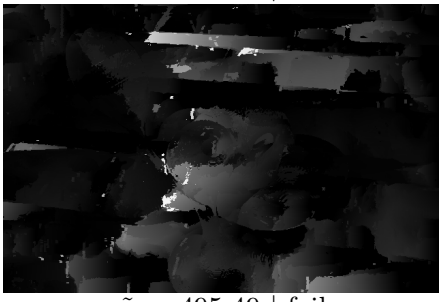


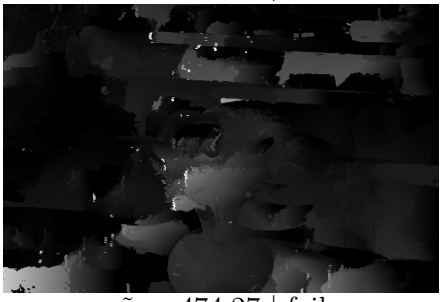
$\tilde{m} = 516.79$ | fail

$\tilde{m} = 495.49$ | fail

$\tilde{m} = 462.57$ | fail

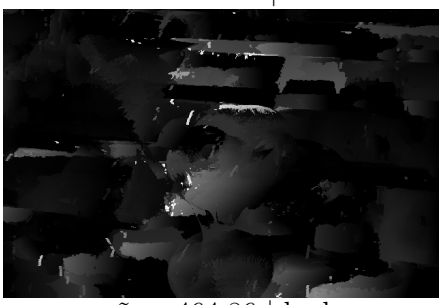


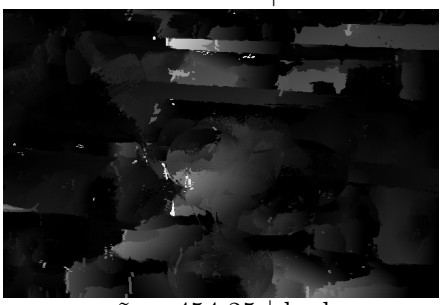




$\tilde{m} = 495.49$ | fail

$\tilde{m} = 474.27$ | fail

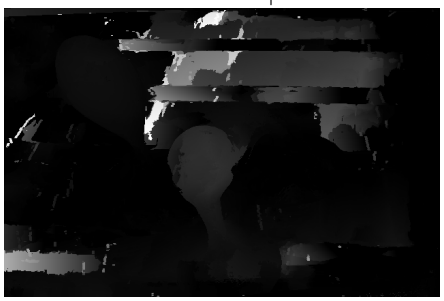




$\tilde{m} = 476.43$ | bad

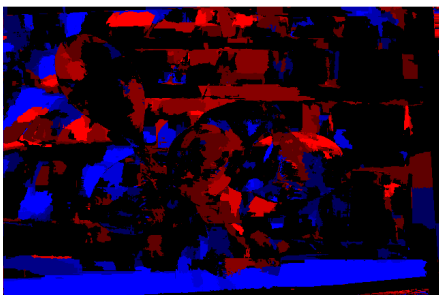
$\tilde{m} = 464.86$ | bad

$\tilde{m} = 454.25$ | bad

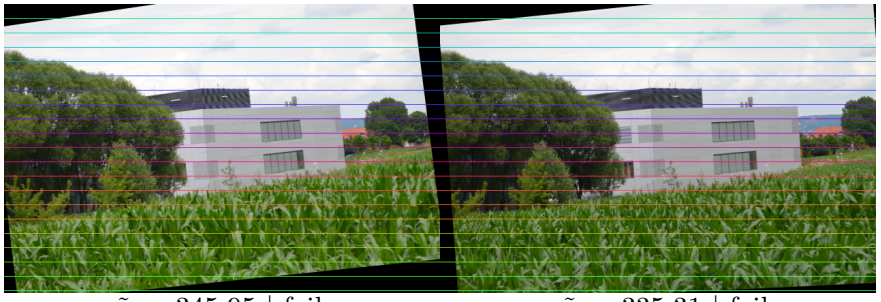


$\tilde{m} = 466.11$ | bad/best

$\tilde{m} = 457.82$ | bad



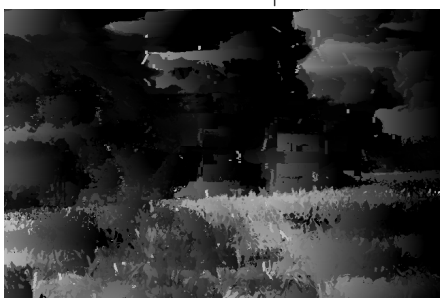
farBuilding

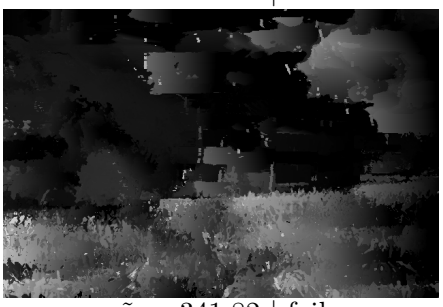


$\tilde{m} = 354.60$ | fail

$\tilde{m} = 345.95$ | fail

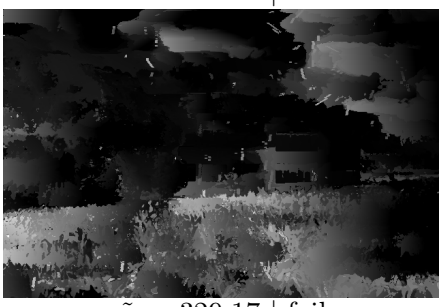
$\tilde{m} = 335.31$ | fail





$\tilde{m} = 349.64$ | fail

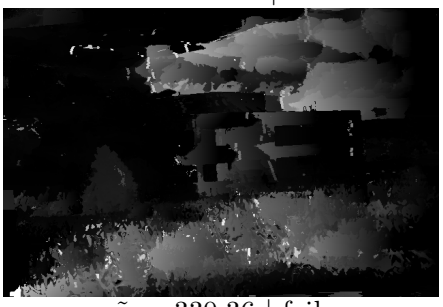
$\tilde{m} = 341.82$ | fail



$\tilde{m} = 338.53$ | fail

$\tilde{m} = 329.17$ | fail

$\tilde{m} = 320.37$ | fail

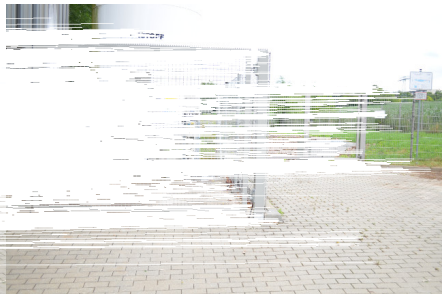


$\tilde{m} = 330.36$ | fail

$\tilde{m} = 325.17$ | fail

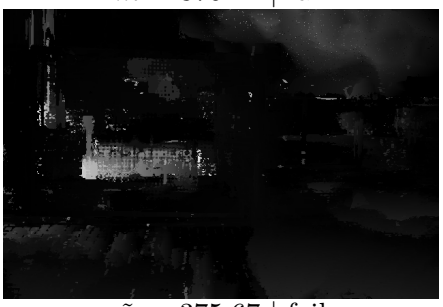
fence





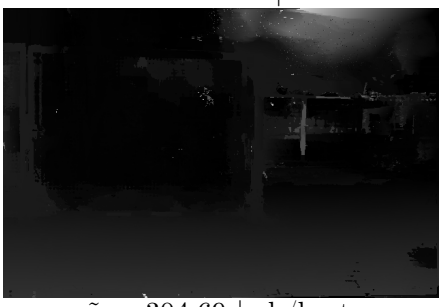
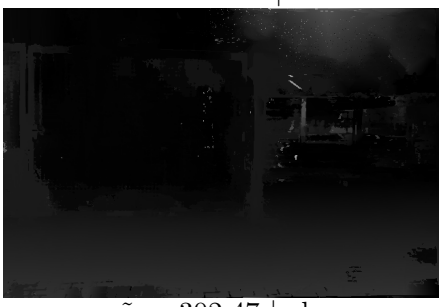

$\tilde{m} = 397.20$ | fail $\tilde{m} = 382.90$ | fail $\tilde{m} = 370.24$ | fail





$\tilde{m} = 387.34$ | fail $\tilde{m} = 375.67$ | fail

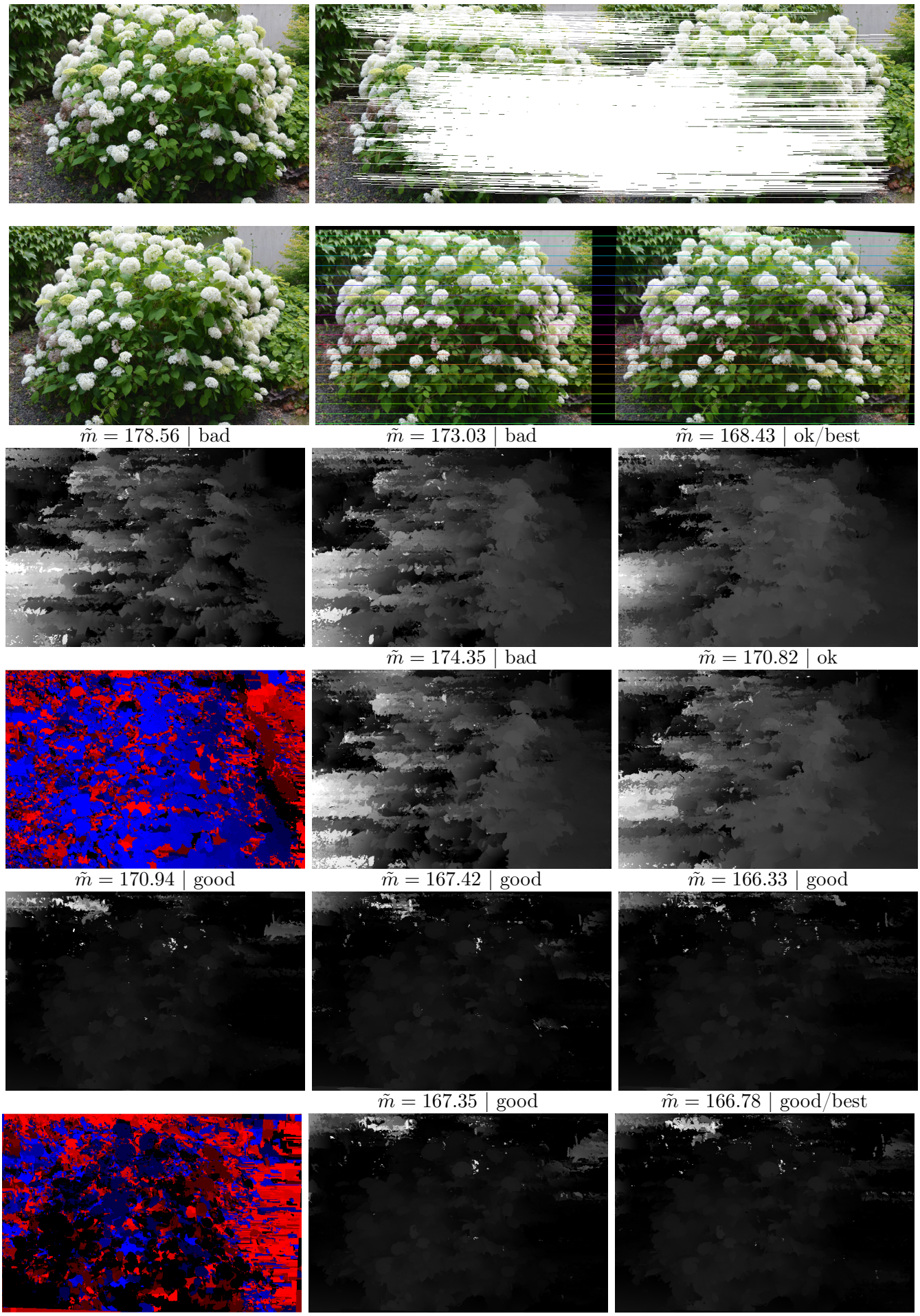

$\tilde{m} = 309.38$ | ok/better door $\tilde{m} = 303.91$ | ok $\tilde{m} = 302.73$ | ok

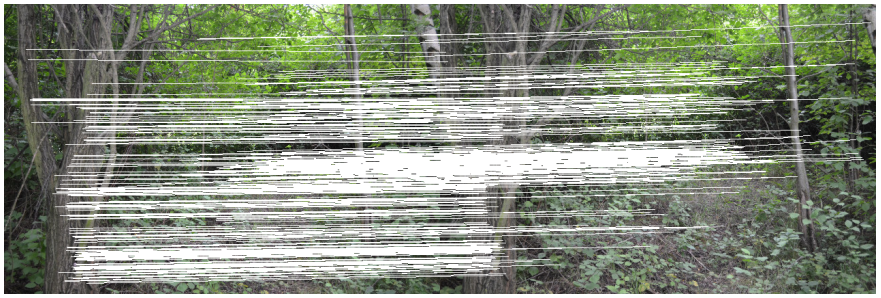
$\tilde{m} = 304.69$ | ok/best $\tilde{m} = 302.47$ | ok



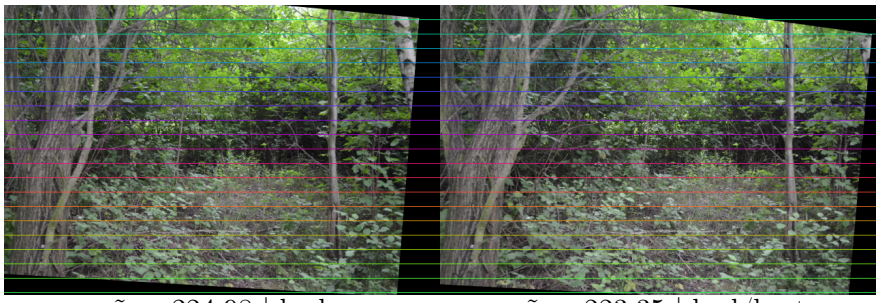
flowers


$\tilde{m} = 178.56$ | bad
$\tilde{m} = 173.03$ | bad
$\tilde{m} = 168.43$ | ok/best
$\tilde{m} = 174.35$ | bad
$\tilde{m} = 170.82$ | ok
$\tilde{m} = 170.94$ | good
$\tilde{m} = 167.42$ | good
$\tilde{m} = 166.33$ | good
$\tilde{m} = 167.35$ | good
$\tilde{m} = 166.78$ | good/best

forest

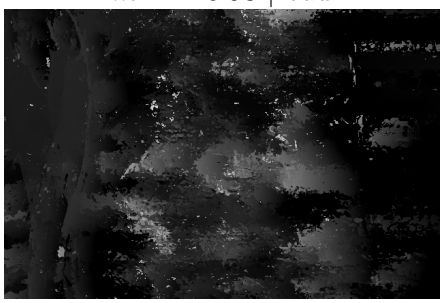


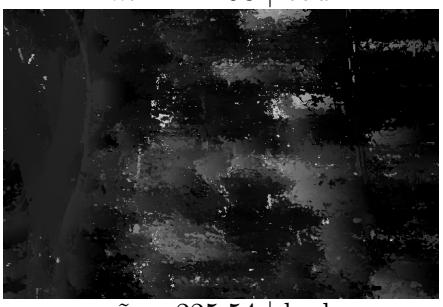
$\tilde{m} = 226.93$ | bad



$\tilde{m} = 224.98$ | bad

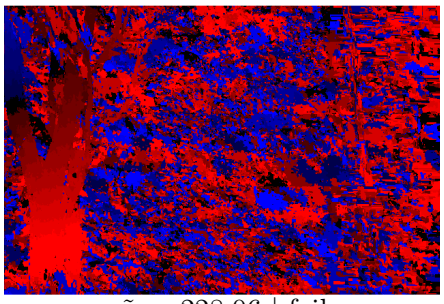
$\tilde{m} = 223.35$ | bad/best

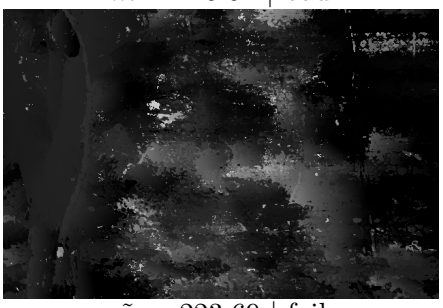


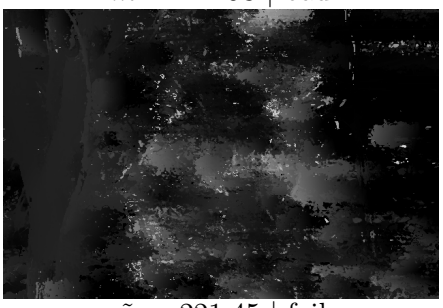


$\tilde{m} = 225.54$ | bad

$\tilde{m} = 224.93$ | bad







$\tilde{m} = 228.06$ | fail

$\tilde{m} = 223.69$ | fail

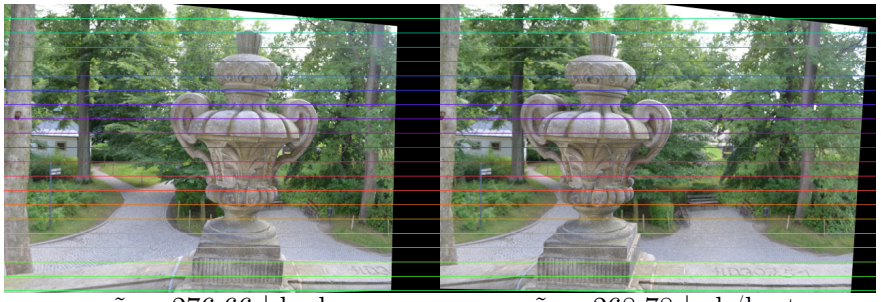
$\tilde{m} = 221.45$ | fail

$\tilde{m} = 224.25$ | fail

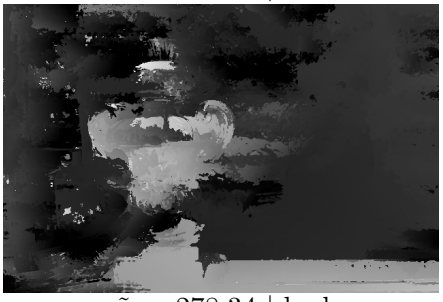
$\tilde{m} = 223.29$ | fail

koenigstein

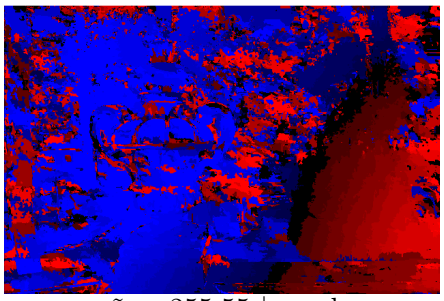
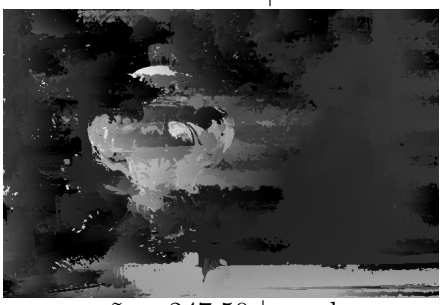
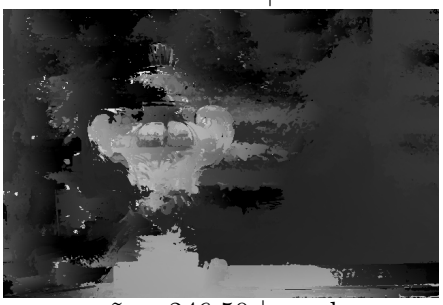


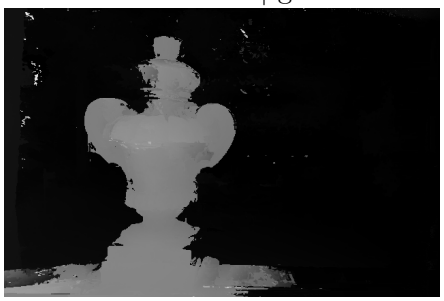
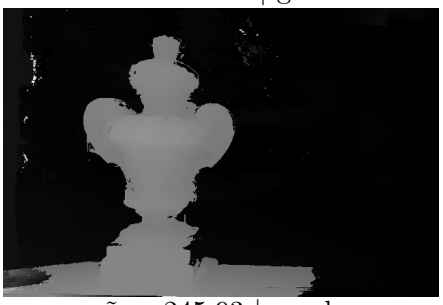
$\tilde{m} = 285.78$ | bad $\tilde{m} = 276.66$ | bad $\tilde{m} = 268.78$ | ok/best

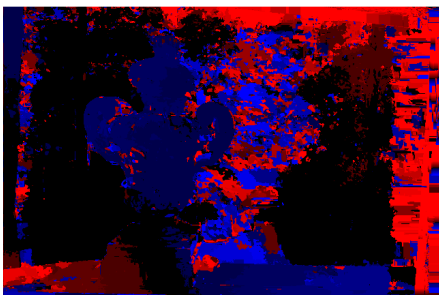

$\tilde{m} = 278.34$ | bad $\tilde{m} = 274.43$ | bad

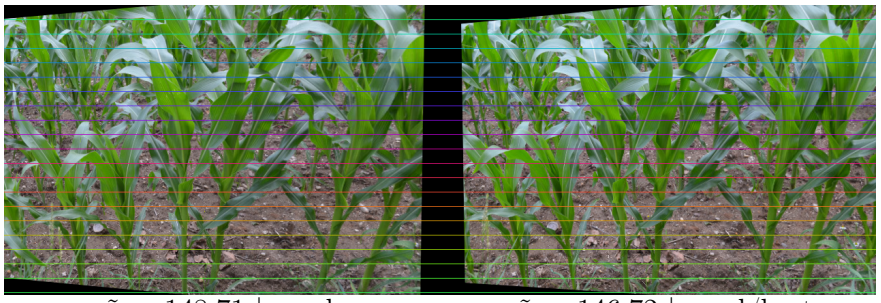
$\tilde{m} = 255.55$ | good $\tilde{m} = 247.50$ | good $\tilde{m} = 246.50$ | good


$\tilde{m} = 245.93$ | good $\tilde{m} = 244.55$ | good/best



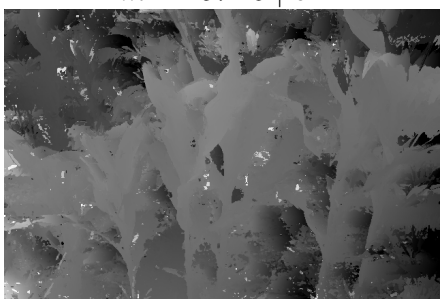
maize

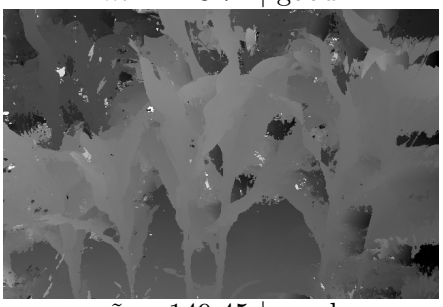


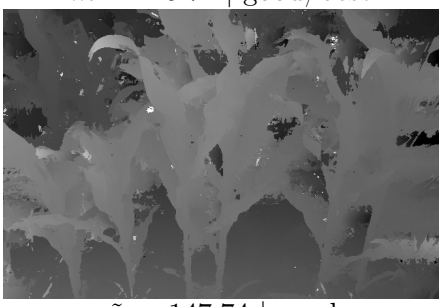
$\tilde{m} = 157.46$ | ok

$\tilde{m} = 148.71$ | good

$\tilde{m} = 146.72$ | good/best

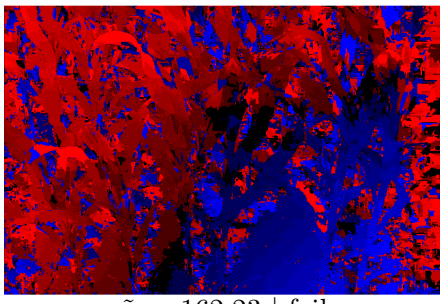


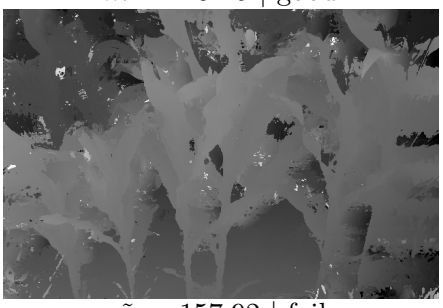


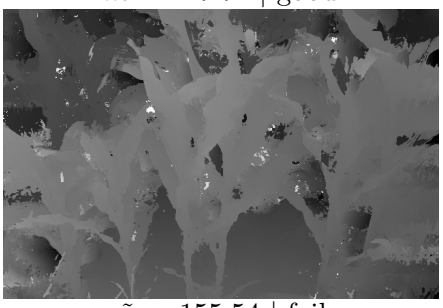


$\tilde{m} = 149.45$ | good

$\tilde{m} = 147.74$ | good



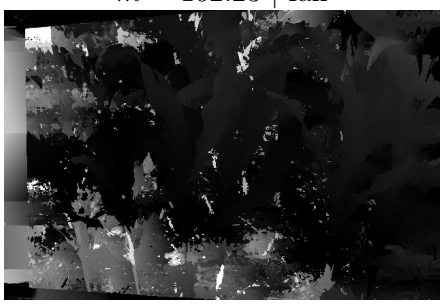




$\tilde{m} = 162.23$ | fail

$\tilde{m} = 157.92$ | fail

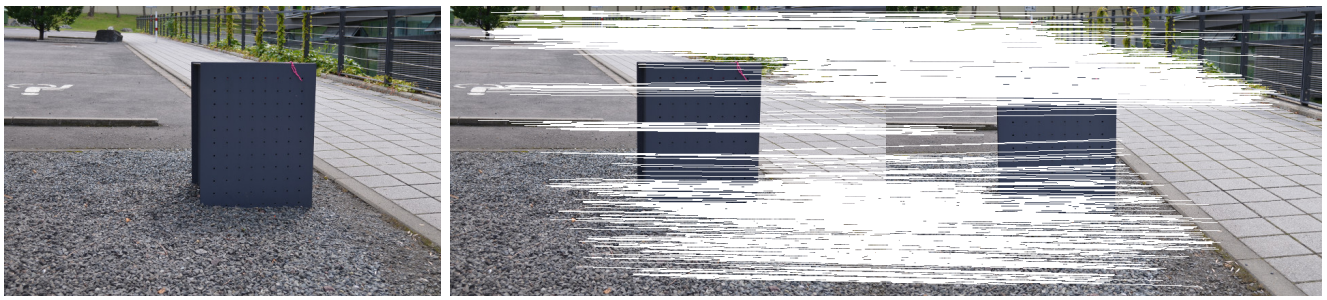
$\tilde{m} = 155.54$ | fail

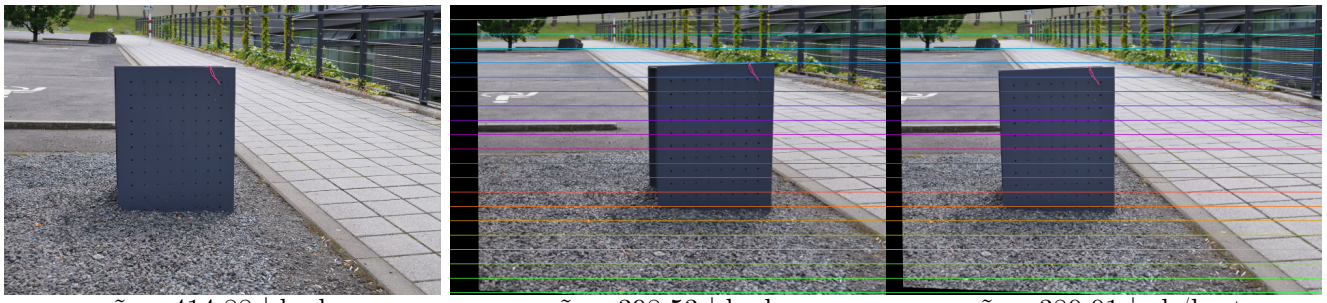


$\tilde{m} = 158.30$ | fail

$\tilde{m} = 156.60$ | fail

metal

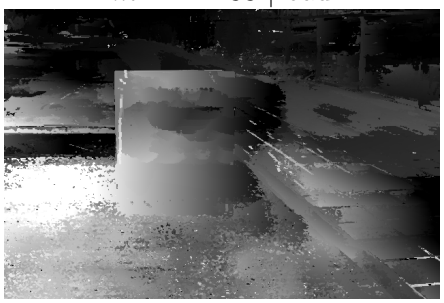


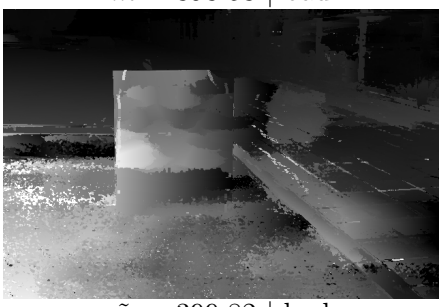


$\tilde{m} = 414.88$ | bad

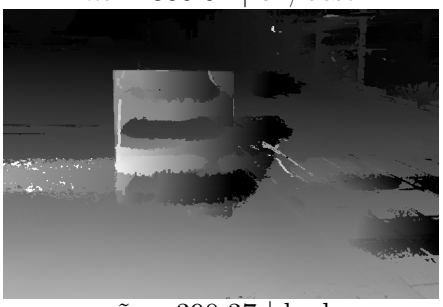
$\tilde{m} = 398.53$ | bad

$\tilde{m} = 380.01$ | ok/best

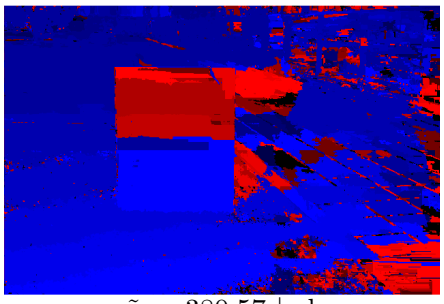




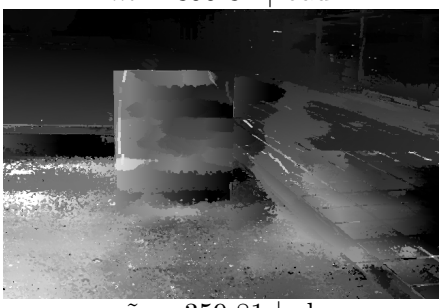
$\tilde{m} = 399.82$ | bad



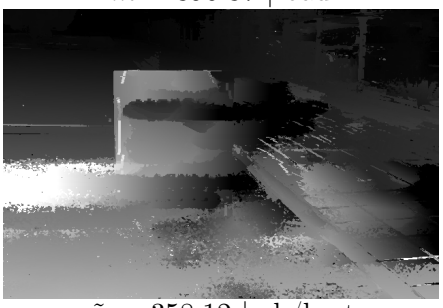
$\tilde{m} = 390.37$ | bad



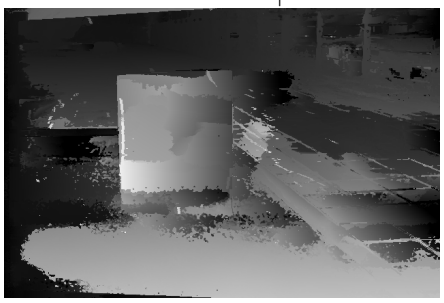
$\tilde{m} = 380.57$ | ok

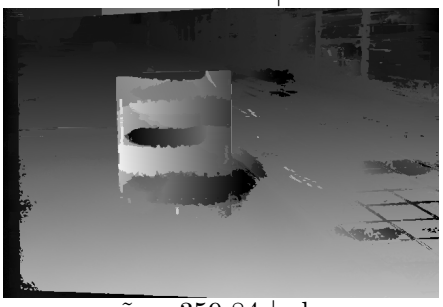


$\tilde{m} = 359.81$ | ok

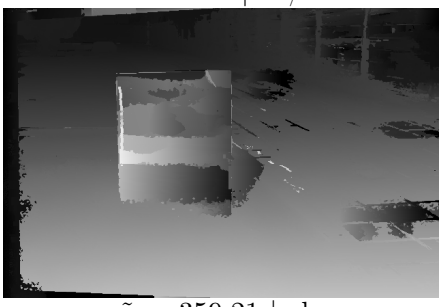


$\tilde{m} = 358.12$ | ok/best

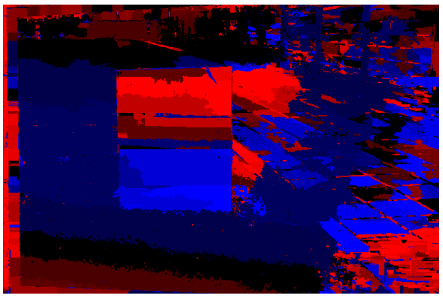


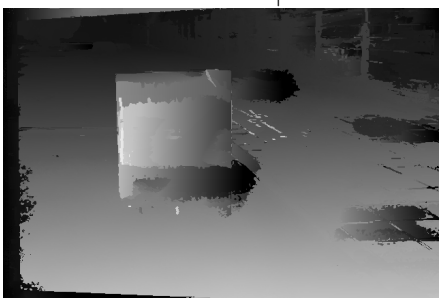


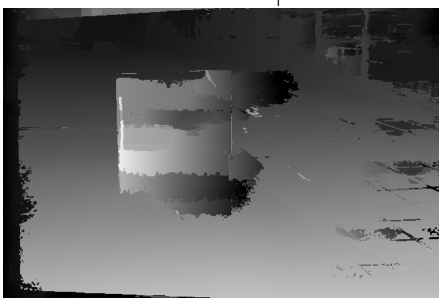
$\tilde{m} = 359.84$ | ok



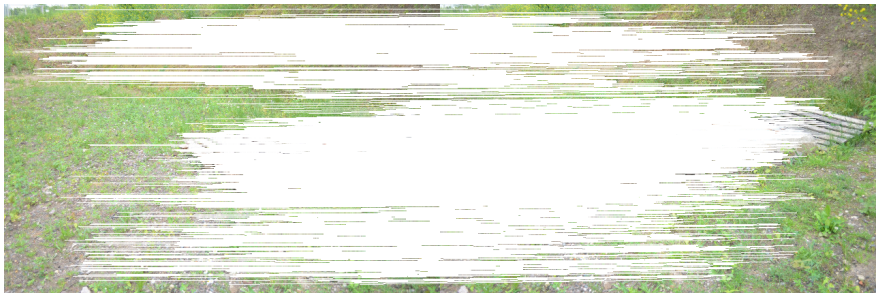
$\tilde{m} = 359.21$ | ok

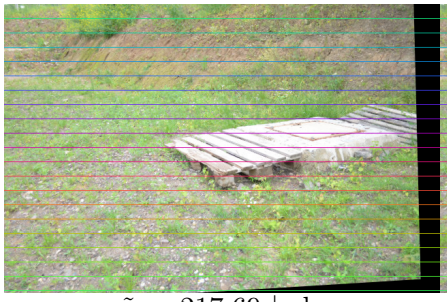


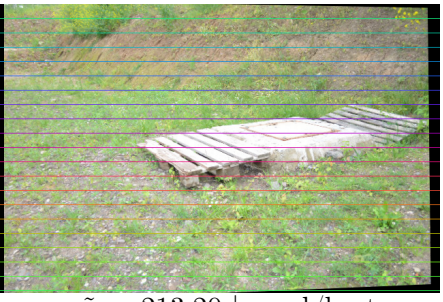




palett



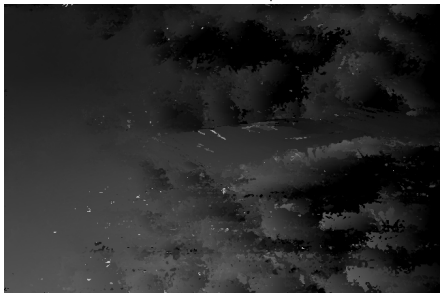


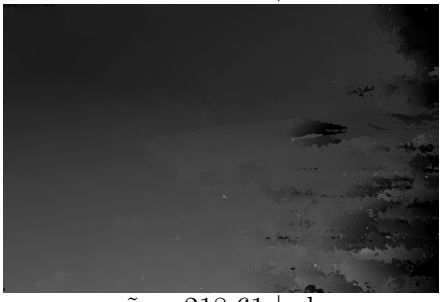


$\tilde{m} = 229.94$ | bad

$\tilde{m} = 217.69$ | ok

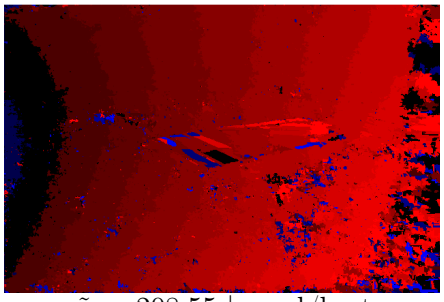
$\tilde{m} = 213.20$ | good/best

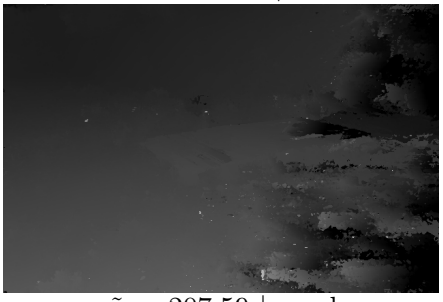


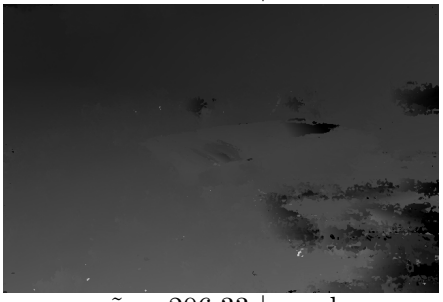


$\tilde{m} = 218.61$ | ok

$\tilde{m} = 215.20$ | good



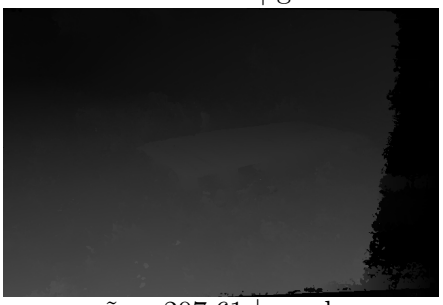




$\tilde{m} = 208.55$ | good/best

$\tilde{m} = 207.59$ | good

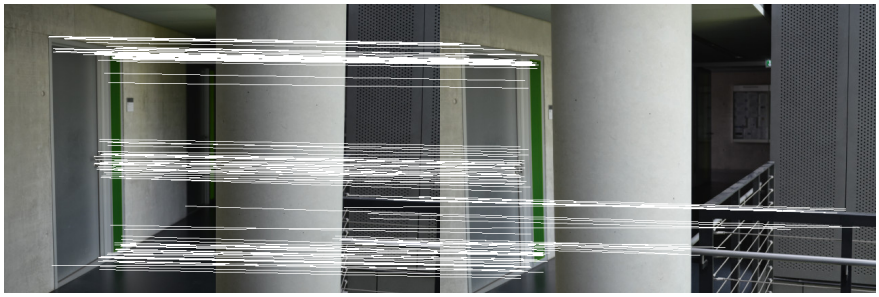
$\tilde{m} = 206.33$ | good

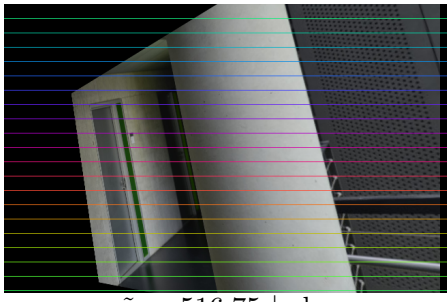


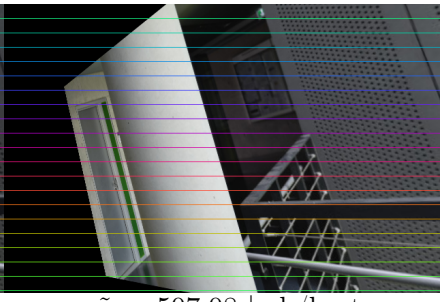
$\tilde{m} = 207.61$ | good

$\tilde{m} = 206.63$ | good

pillar



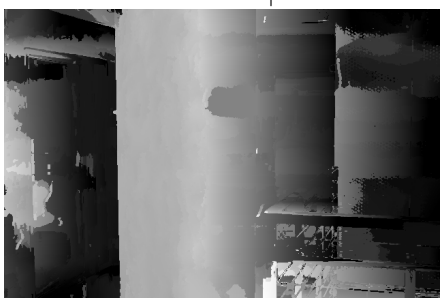


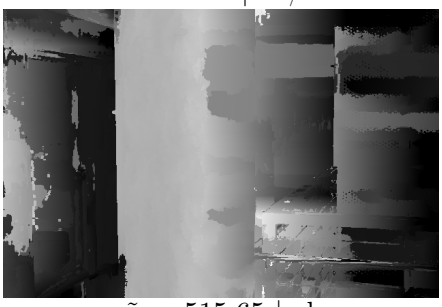


$\tilde{m} = 531.36$ | bad

$\tilde{m} = 516.75$ | ok

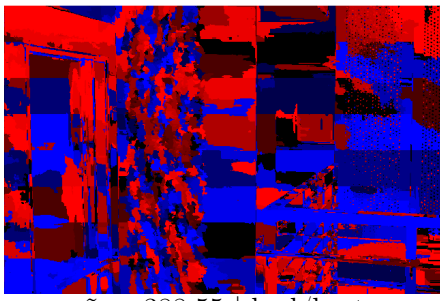
$\tilde{m} = 507.08$ | ok/best

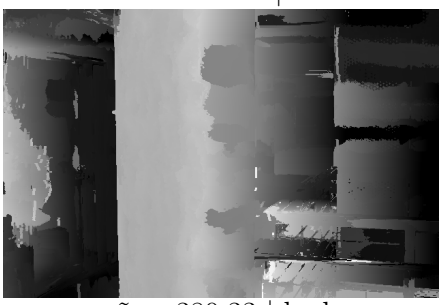




$\tilde{m} = 524.53$ | ok

$\tilde{m} = 515.65$ | ok

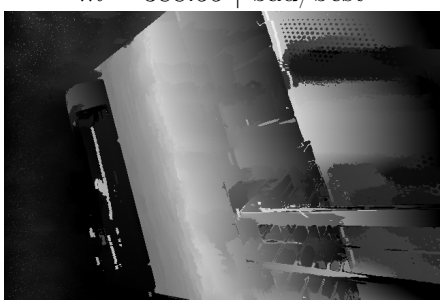


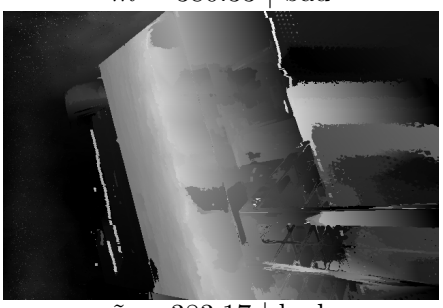


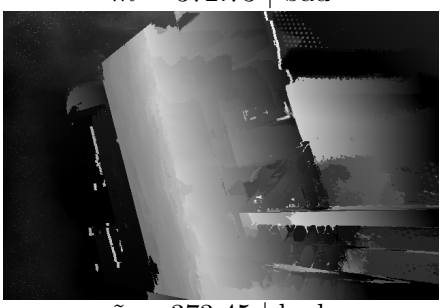
$\tilde{m} = 388.55$ | bad/best

$\tilde{m} = 380.33$ | bad

$\tilde{m} = 372.73$ | bad

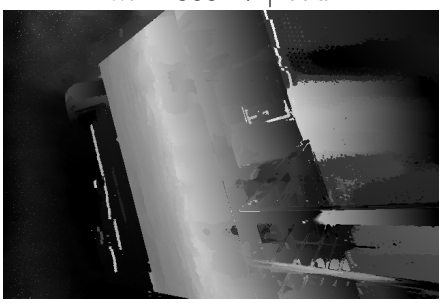


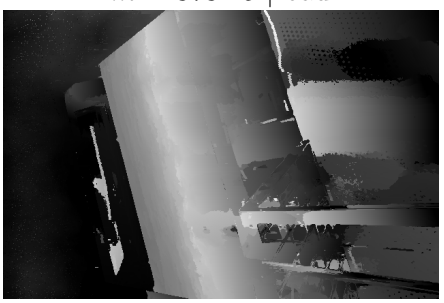




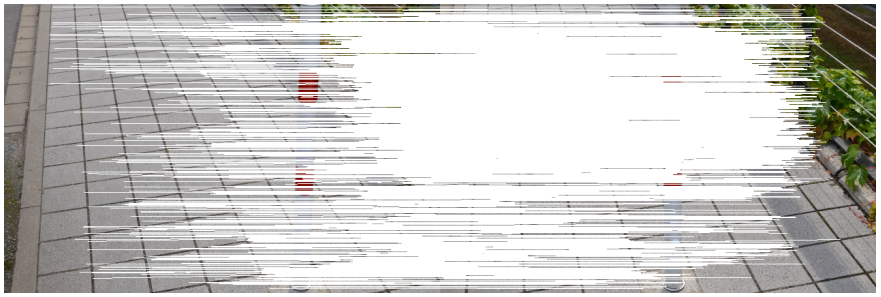
$\tilde{m} = 383.17$ | bad

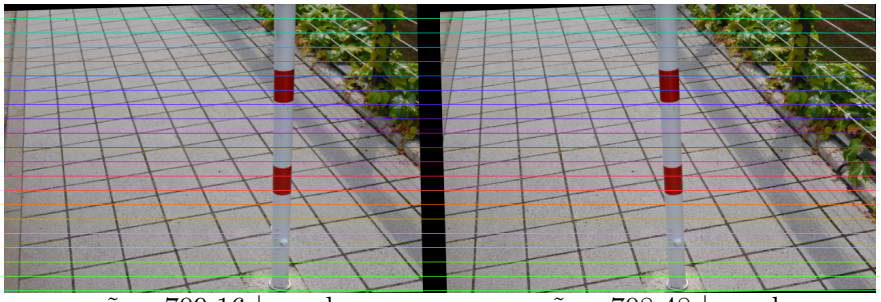
$\tilde{m} = 373.45$ | bad





pole

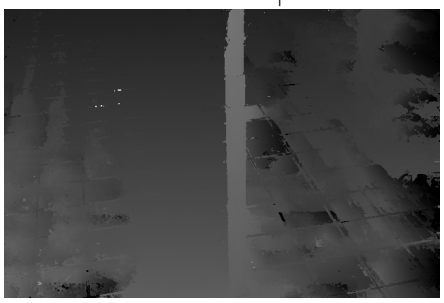


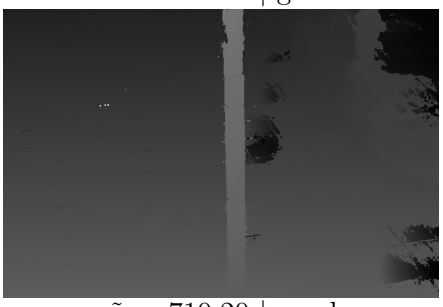


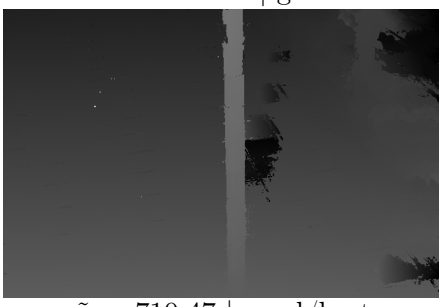
$\tilde{m} = 752.32$ | ok

$\tilde{m} = 709.16$ | good

$\tilde{m} = 708.48$ | good

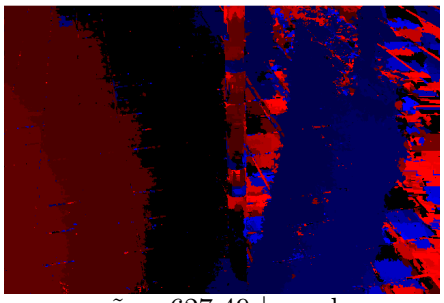


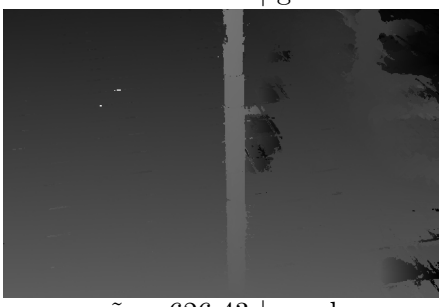


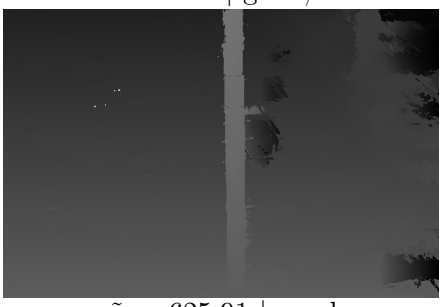


$\tilde{m} = 710.20$ | good

$\tilde{m} = 710.47$ | good/best



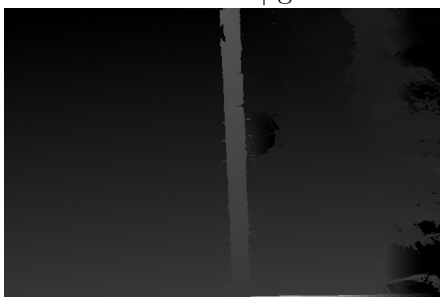


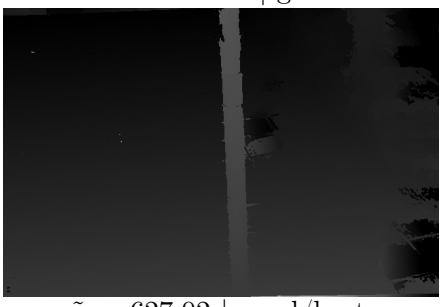


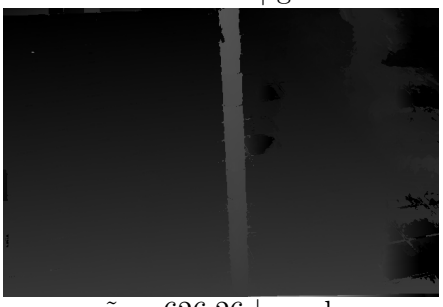
$\tilde{m} = 627.49$ | good

$\tilde{m} = 626.43$ | good

$\tilde{m} = 625.91$ | good

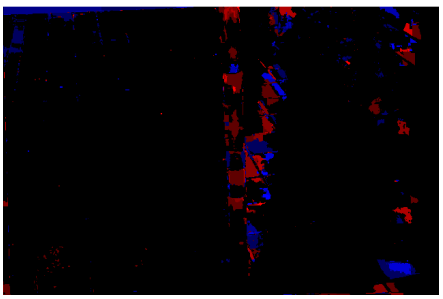


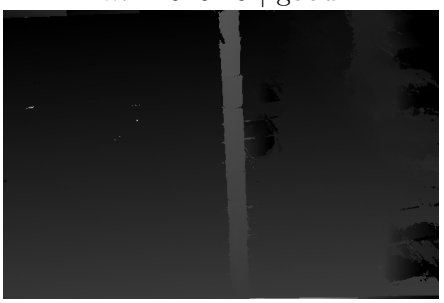




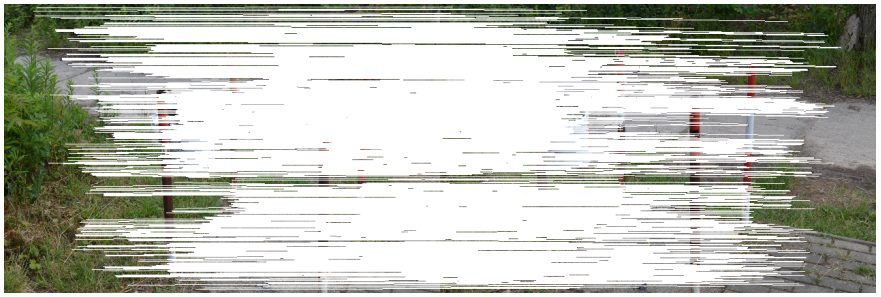
$\tilde{m} = 627.02$ | good/best

$\tilde{m} = 626.26$ | good





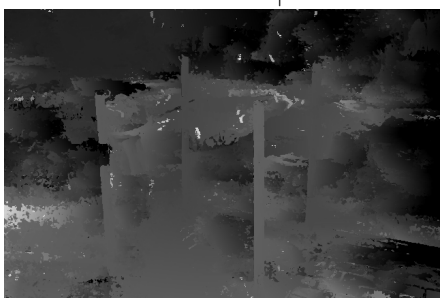
poles

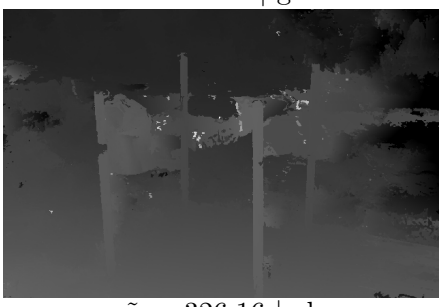


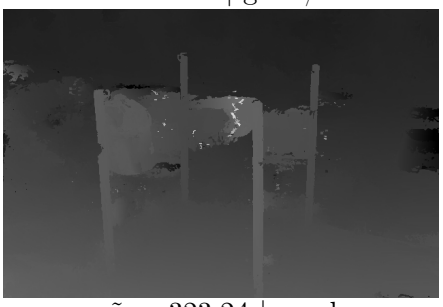
$\tilde{m} = 335.64$ | ok

$\tilde{m} = 323.86$ | good

$\tilde{m} = 318.83$ | good/best

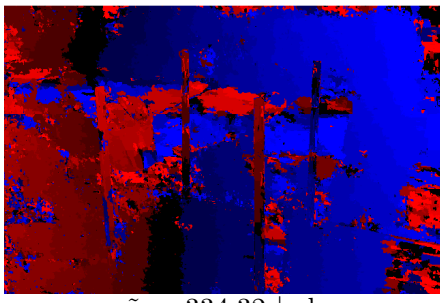


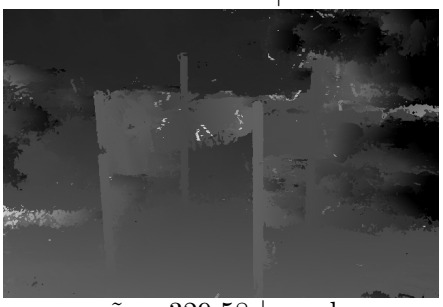


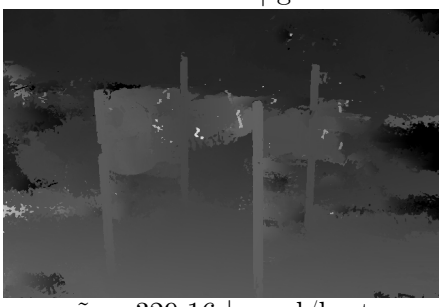


$\tilde{m} = 326.16$ | ok

$\tilde{m} = 323.24$ | good



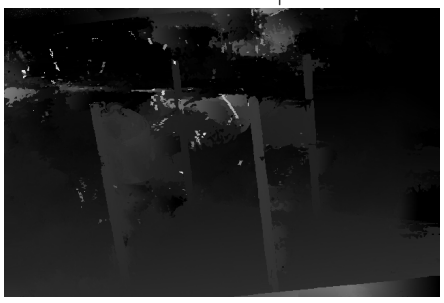


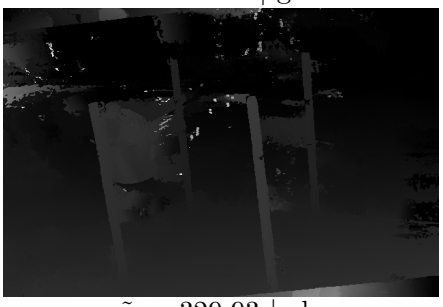


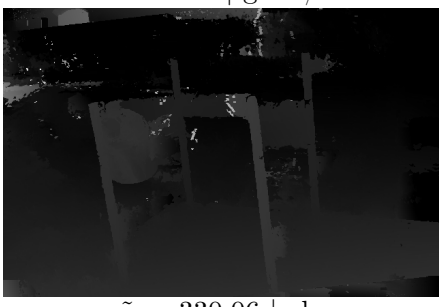
$\tilde{m} = 334.32$ | ok

$\tilde{m} = 329.58$ | good

$\tilde{m} = 329.16$ | good/best

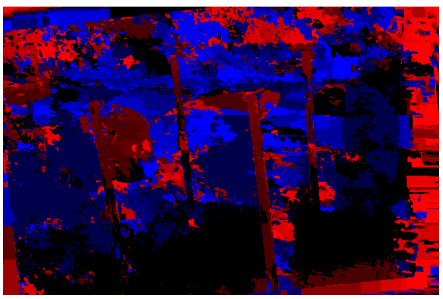


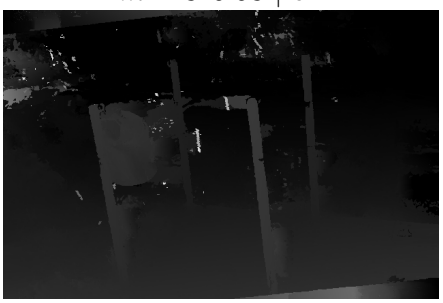


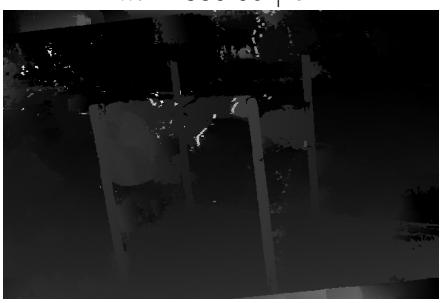


$\tilde{m} = 329.93$ | ok

$\tilde{m} = 330.06$ | ok





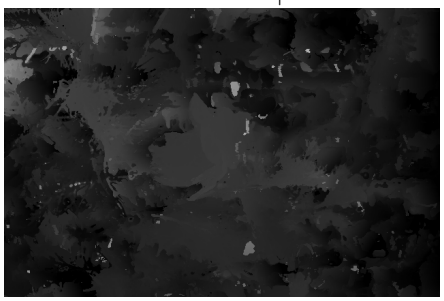
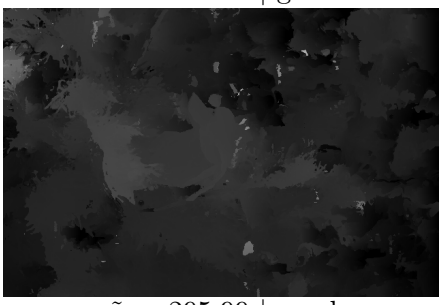
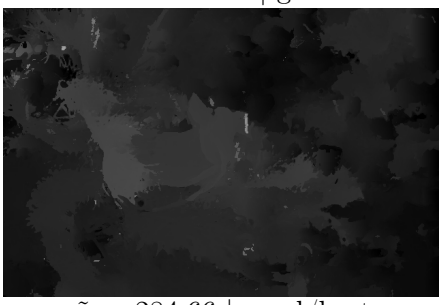


poppy

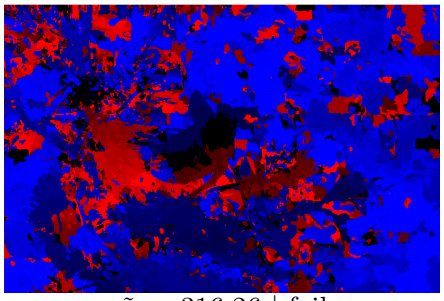
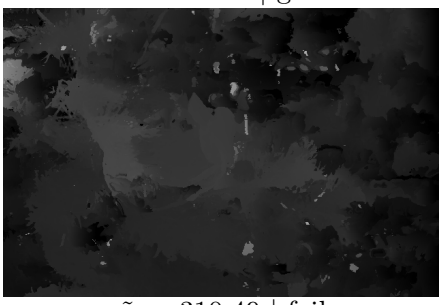
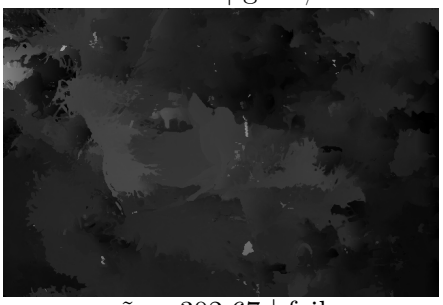




$\tilde{m} = 307.76$ | ok $\tilde{m} = 294.04$ | good $\tilde{m} = 281.50$ | good

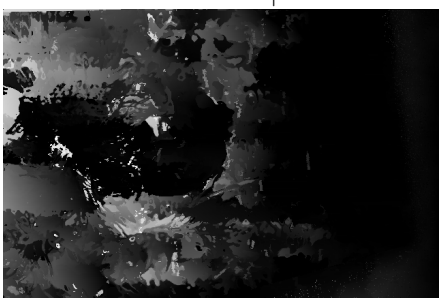
$\tilde{m} = 295.99$ | good $\tilde{m} = 284.66$ | good/best

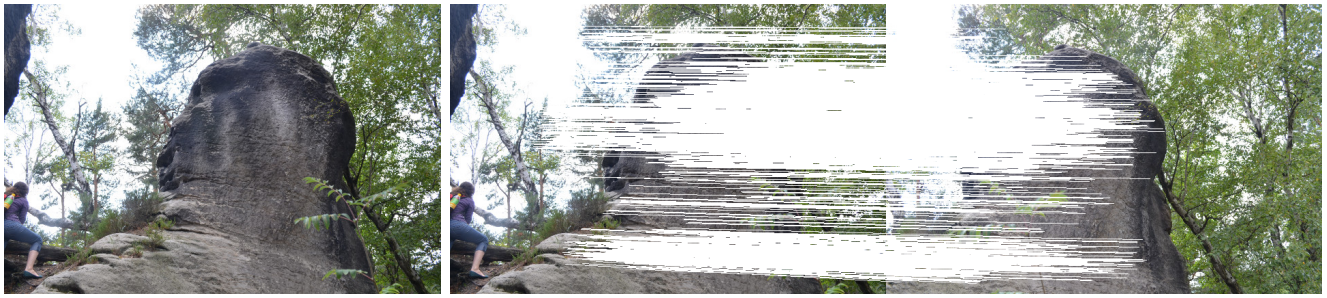
$\tilde{m} = 316.26$ | fail $\tilde{m} = 310.49$ | fail $\tilde{m} = 302.67$ | fail



$\tilde{m} = 311.75$ | fail $\tilde{m} = 306.72$ | fail

rock1



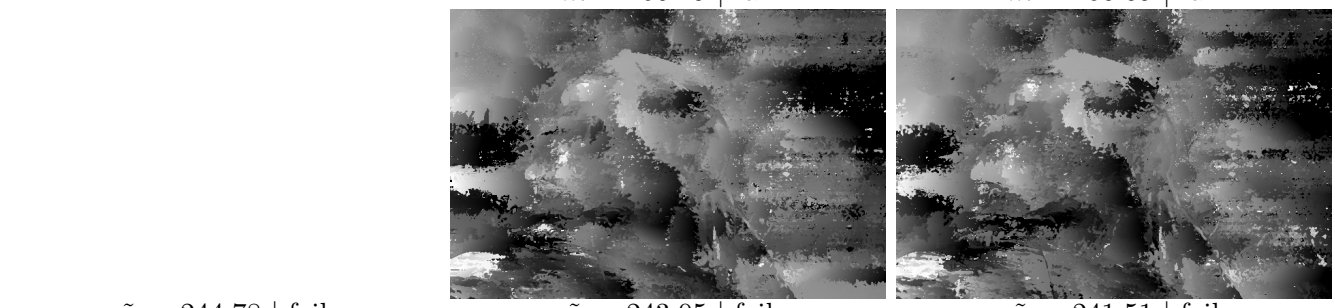
$\tilde{m} = 261.09$ | fail

$\tilde{m} = 259.48$ | fail

$\tilde{m} = 257.74$ | fail

$\tilde{m} = 260.49$ | fail

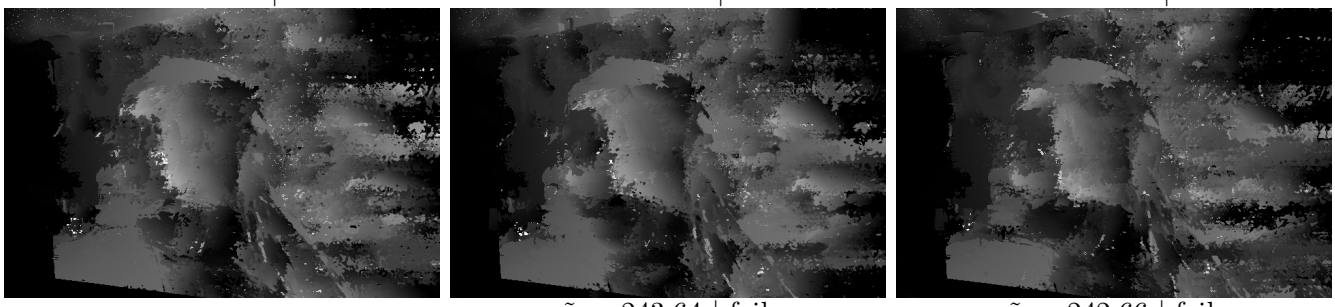
$\tilde{m} = 260.09$ | fail



$\tilde{m} = 244.78$ | fail

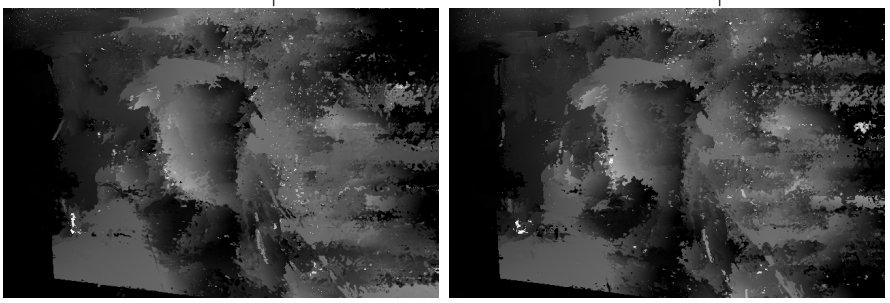
$\tilde{m} = 243.05$ | fail

$\tilde{m} = 241.51$ | fail



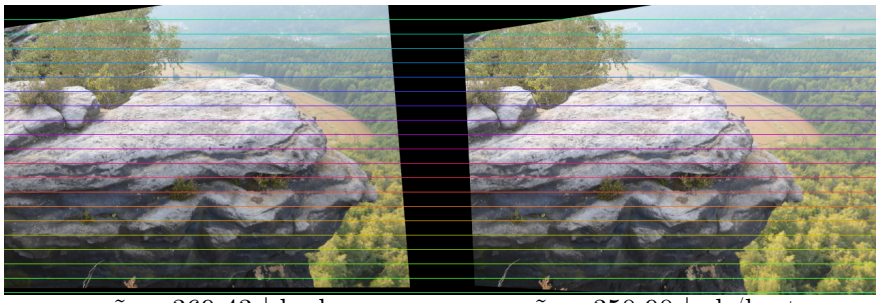
$\tilde{m} = 243.64$ | fail

$\tilde{m} = 242.66$ | fail

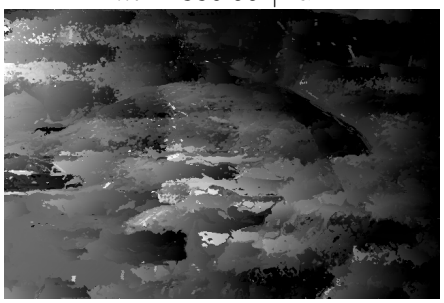
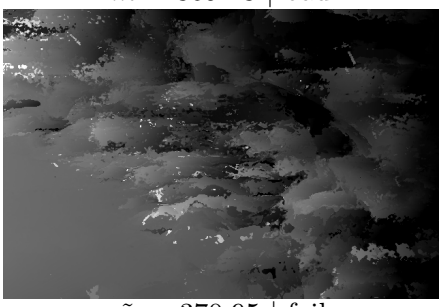


rock2

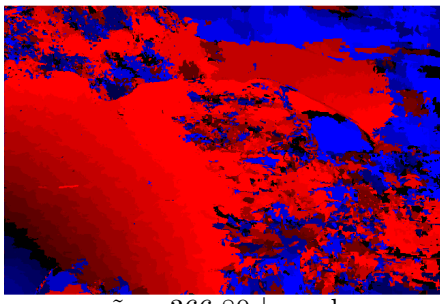
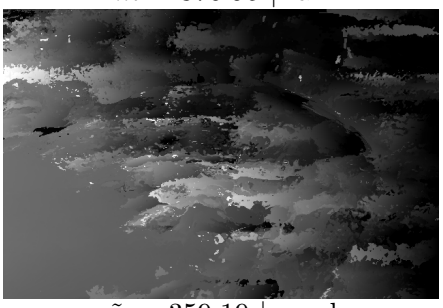
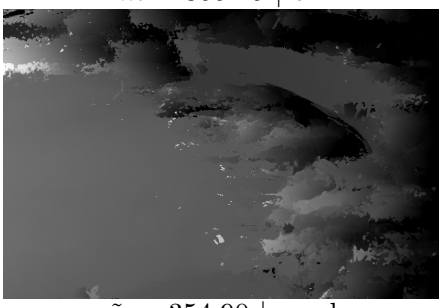


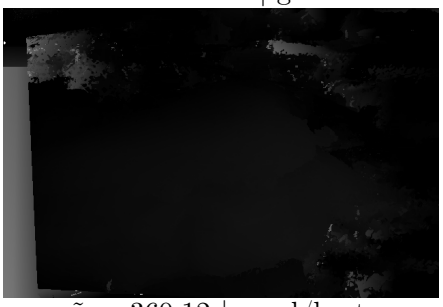
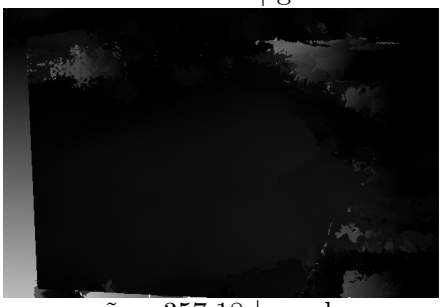
$\tilde{m} = 380.99$ | fail $\tilde{m} = 369.43$ | bad $\tilde{m} = 350.90$ | ok/best

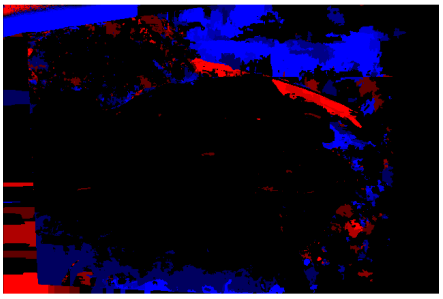
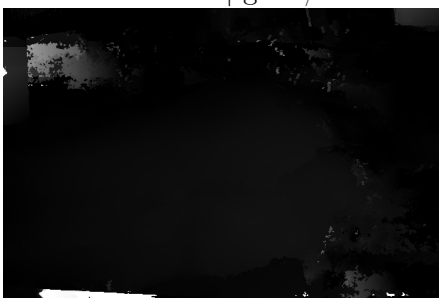

$\tilde{m} = 370.95$ | fail $\tilde{m} = 355.15$ | ok

$\tilde{m} = 366.80$ | good $\tilde{m} = 359.19$ | good $\tilde{m} = 354.90$ | good

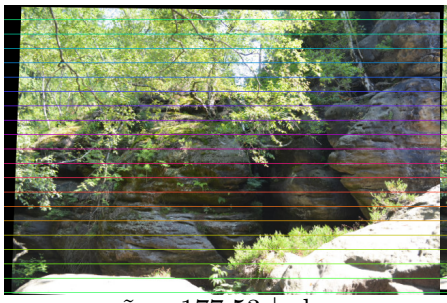




$\tilde{m} = 360.12$ | good/best $\tilde{m} = 357.18$ | good

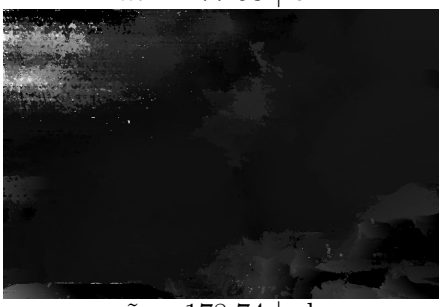
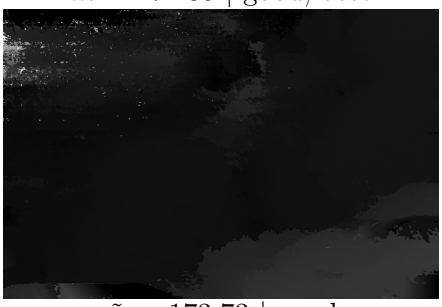




rock3

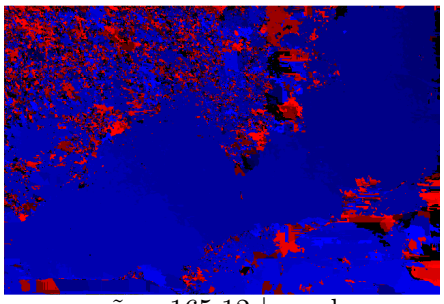


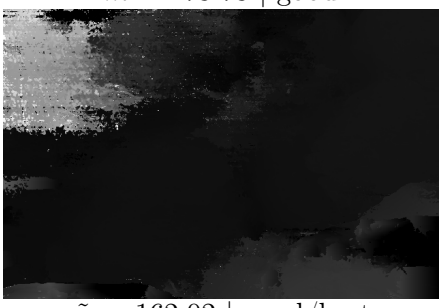



$\tilde{m} = 200.05$ | fail $\tilde{m} = 177.53$ | ok $\tilde{m} = 172.85$ | good/best

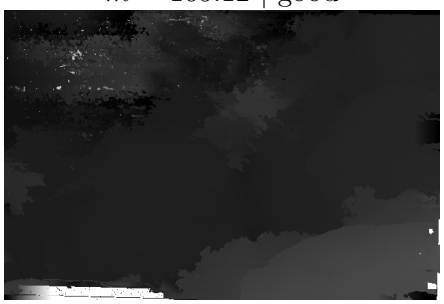
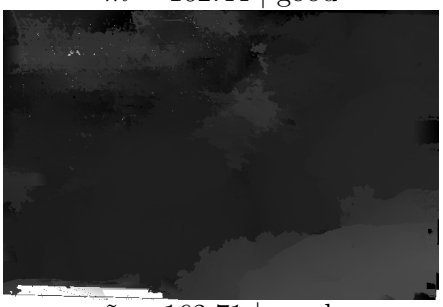
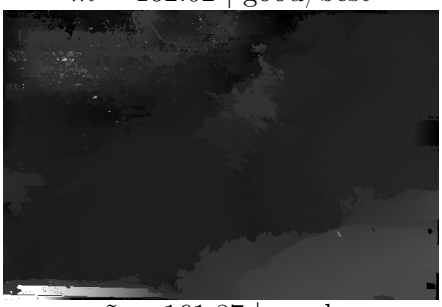




$\tilde{m} = 178.74$ | ok $\tilde{m} = 173.73$ | good

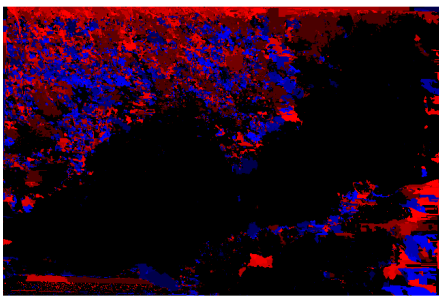
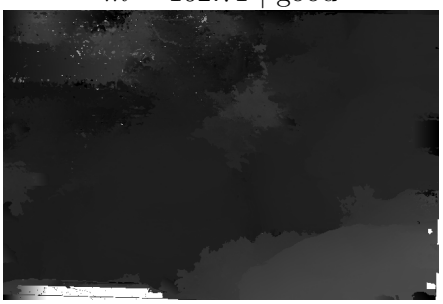
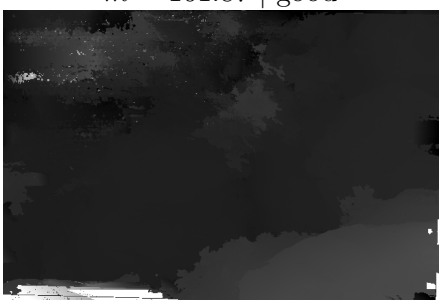




$\tilde{m} = 165.12$ | good $\tilde{m} = 162.44$ | good $\tilde{m} = 162.02$ | good/best

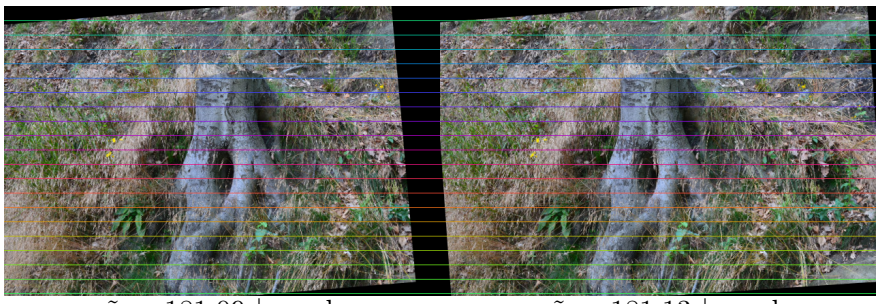




$\tilde{m} = 162.71$ | good $\tilde{m} = 161.87$ | good

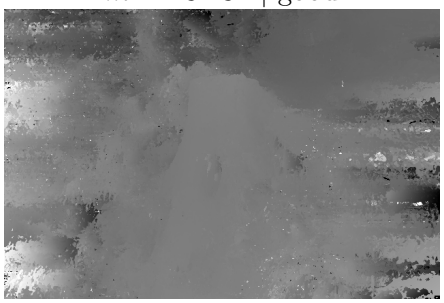
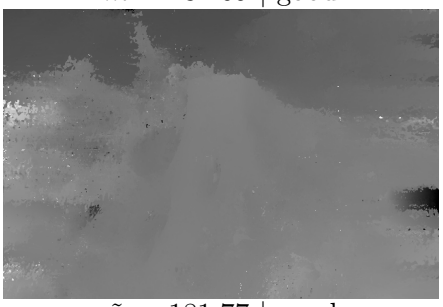
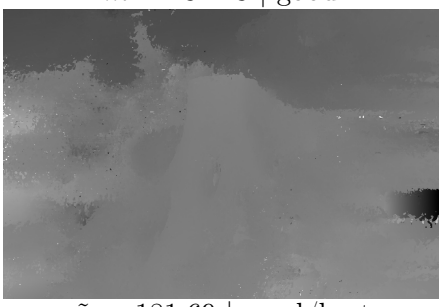




root

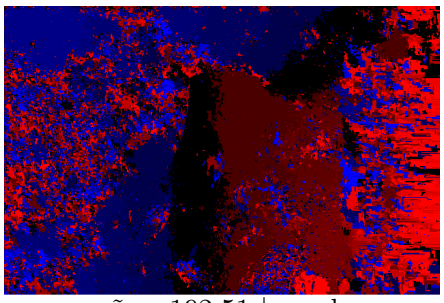
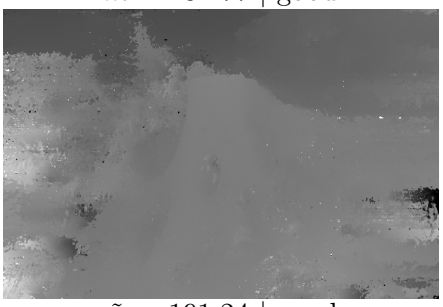
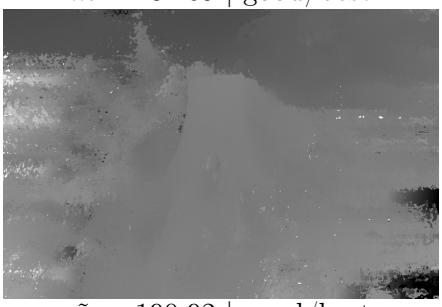




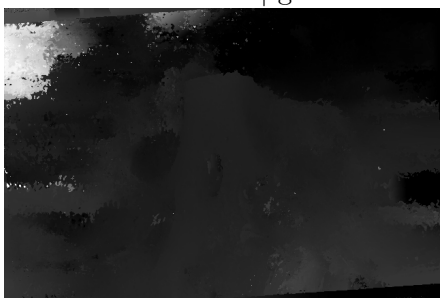
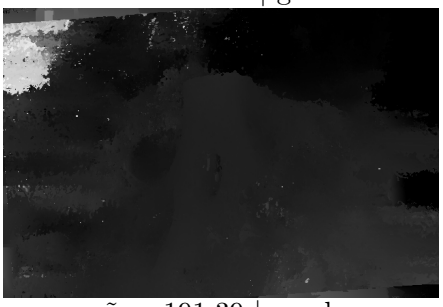
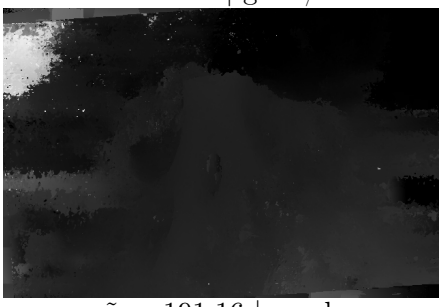
$\tilde{m} = 184.32$ | good $\tilde{m} = 181.09$ | good $\tilde{m} = 181.13$ | good

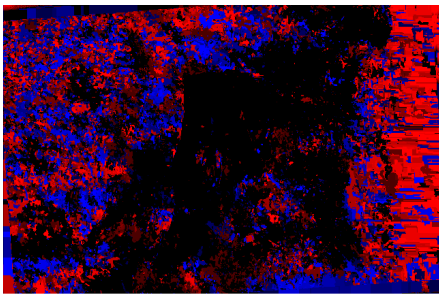
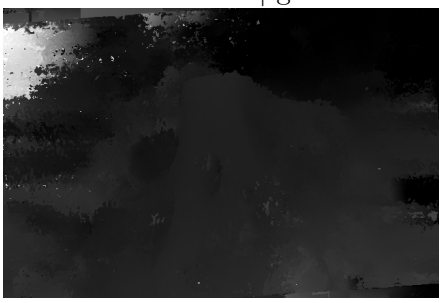
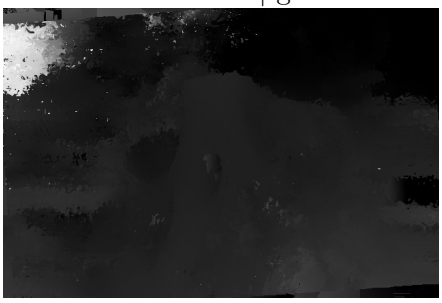
$\tilde{m} = 181.77$ | good $\tilde{m} = 181.69$ | good/best

$\tilde{m} = 192.51$ | good $\tilde{m} = 191.24$ | good $\tilde{m} = 190.92$ | good/best

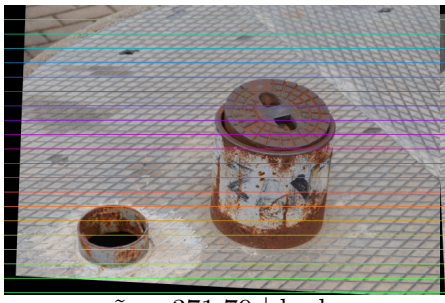
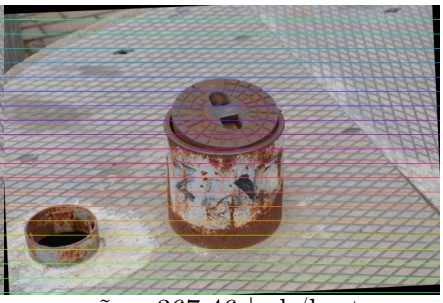




$\tilde{m} = 191.39$ | good $\tilde{m} = 191.16$ | good

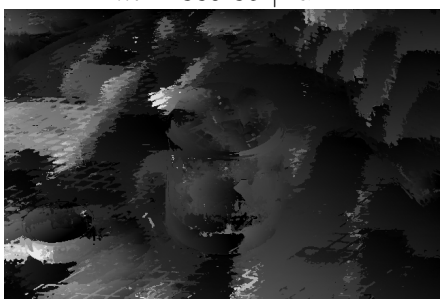
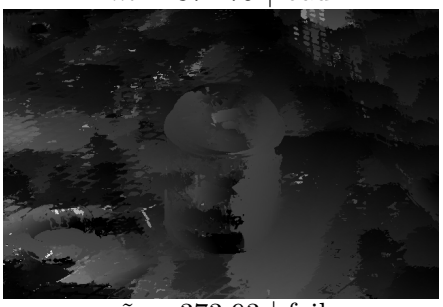
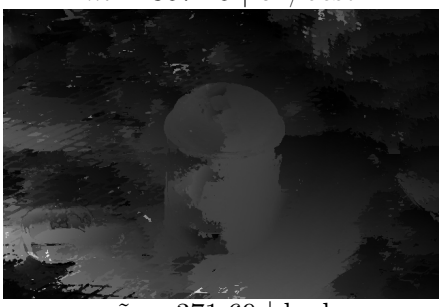




sewerCover

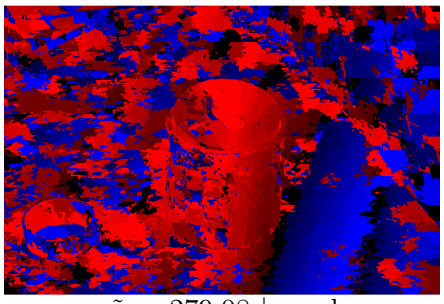
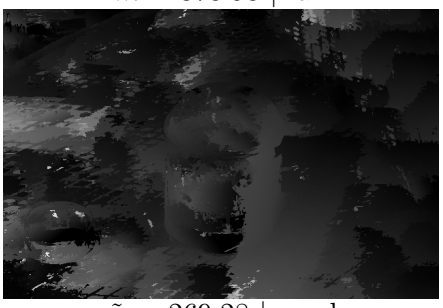
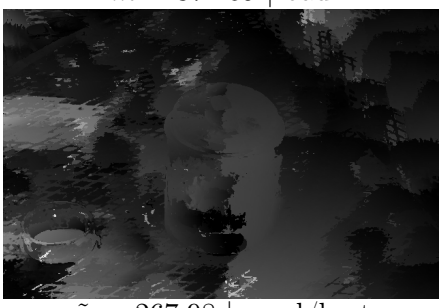


$\tilde{m} = 389.85$ | fail $\tilde{m} = 371.79$ | bad $\tilde{m} = 367.46$ | ok/best

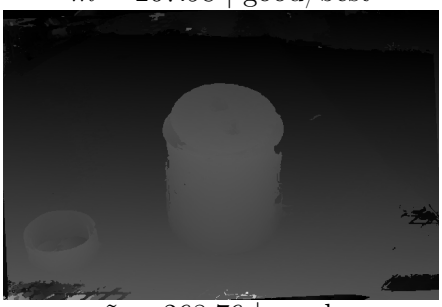




$\tilde{m} = 373.93$ | fail $\tilde{m} = 371.69$ | bad

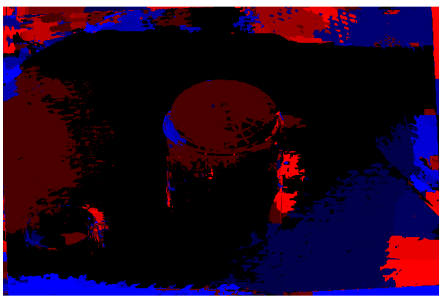




$\tilde{m} = 279.08$ | good $\tilde{m} = 269.28$ | good $\tilde{m} = 267.98$ | good/best

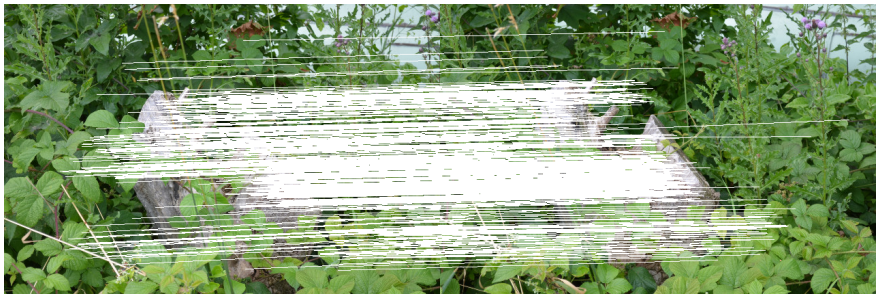




$\tilde{m} = 269.63$ | good $\tilde{m} = 268.76$ | good



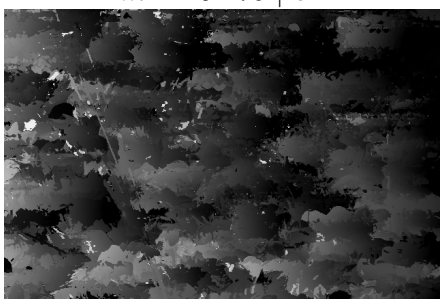
stub

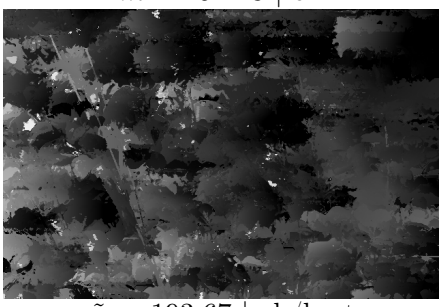


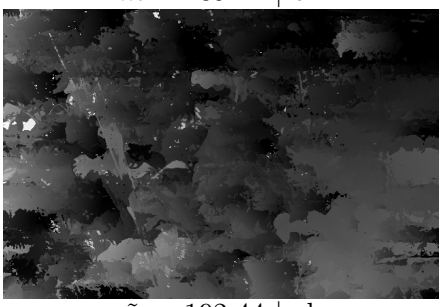
$\tilde{m} = 194.76$ | ok

$\tilde{m} = 192.28$ | ok

$\tilde{m} = 189.24$ | ok

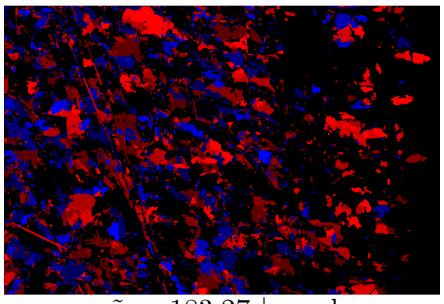


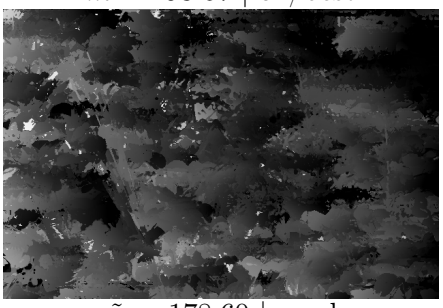


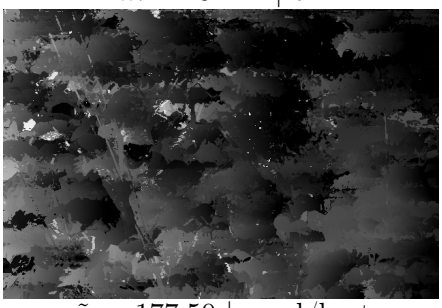


$\tilde{m} = 193.67$ | ok/best

$\tilde{m} = 192.44$ | ok



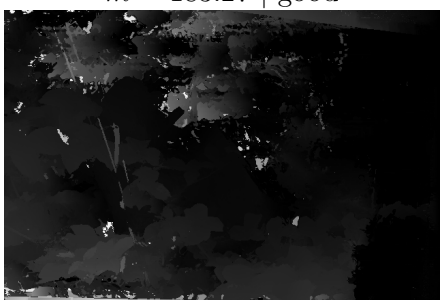


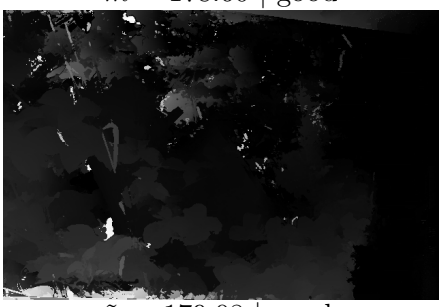


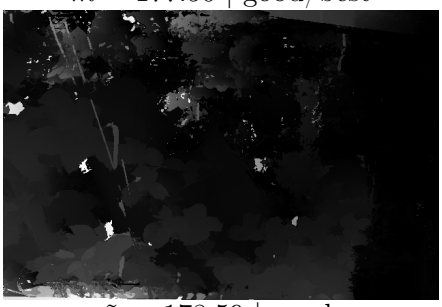
$\tilde{m} = 183.27$ | good

$\tilde{m} = 178.60$ | good

$\tilde{m} = 177.50$ | good/best

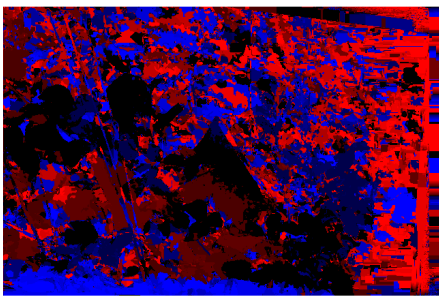


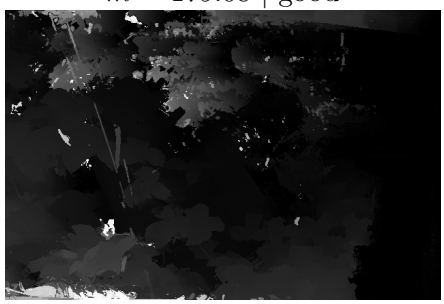


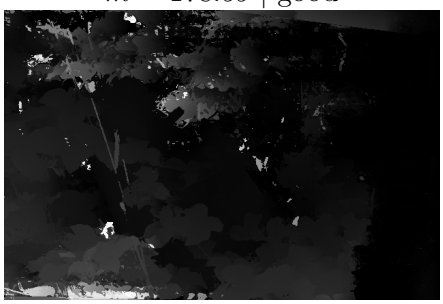


$\tilde{m} = 179.08$ | good

$\tilde{m} = 178.59$ | good

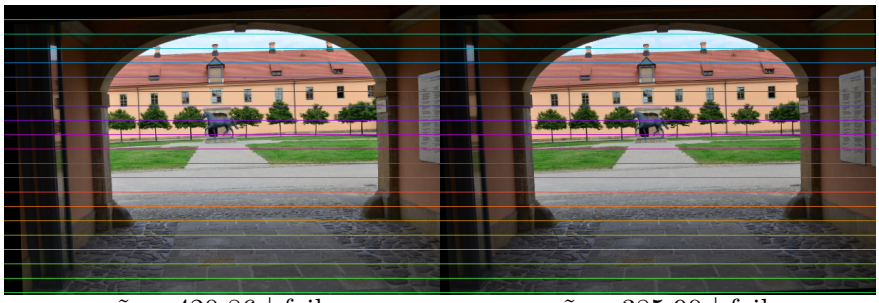






stud



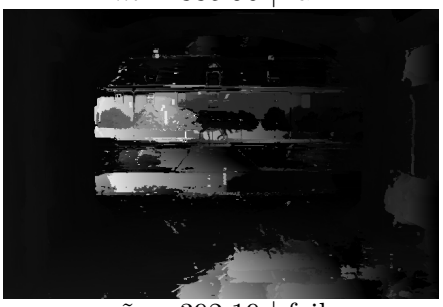


$\tilde{m} = 448.74$ | fail

$\tilde{m} = 420.86$ | fail

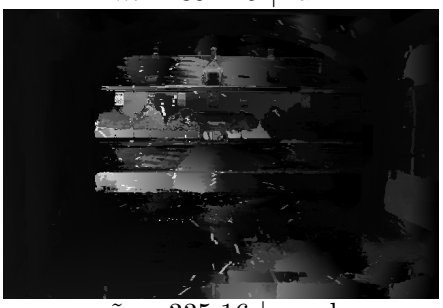
$\tilde{m} = 385.90$ | fail





$\tilde{m} = 422.95$ | fail

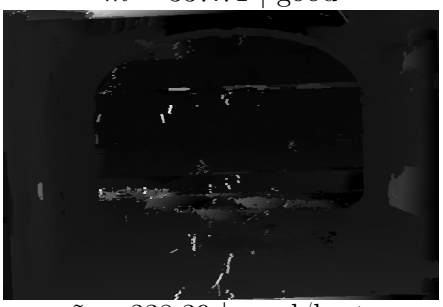
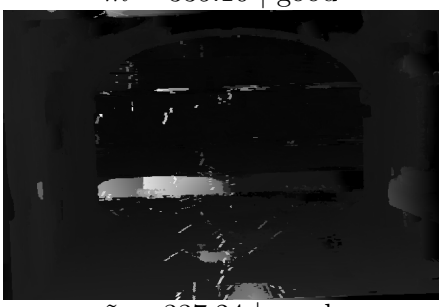
$\tilde{m} = 392.19$ | fail

$\tilde{m} = 344.10$ | good

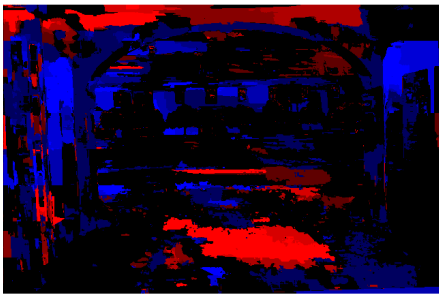
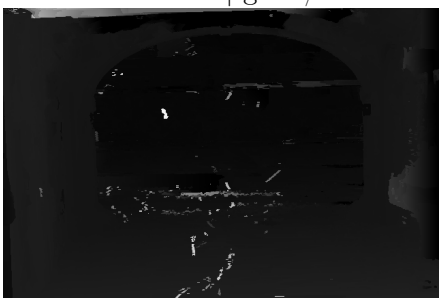
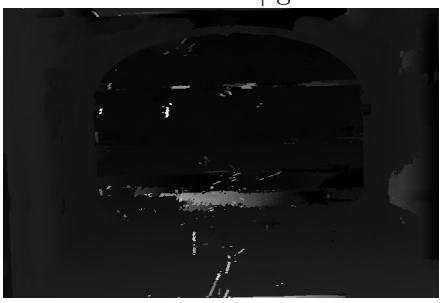
$\tilde{m} = 337.71$ | good

$\tilde{m} = 335.16$ | good

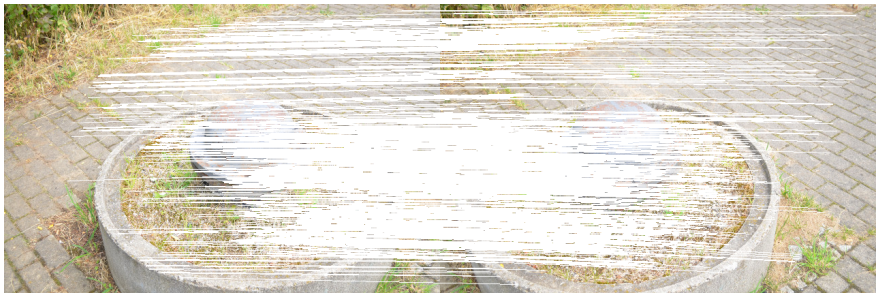




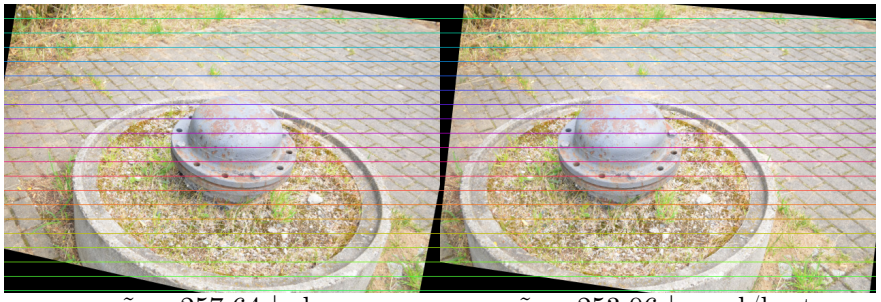
$\tilde{m} = 338.39$ | good/best

$\tilde{m} = 337.24$ | good

thing

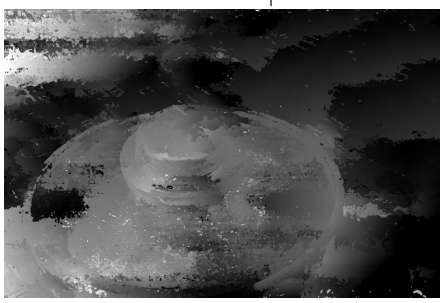


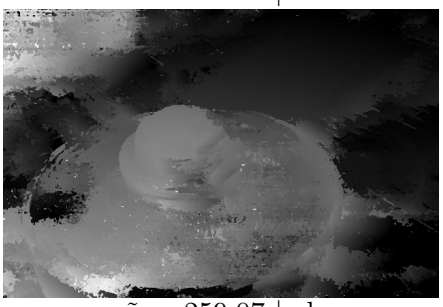


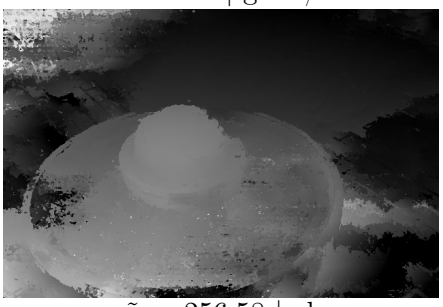
$\tilde{m} = 261.55$ | bad

$\tilde{m} = 257.64$ | ok

$\tilde{m} = 253.06$ | good/best

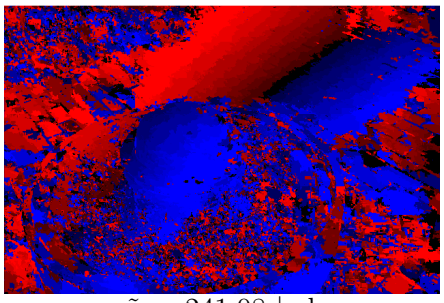


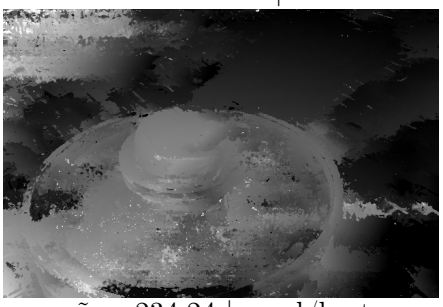


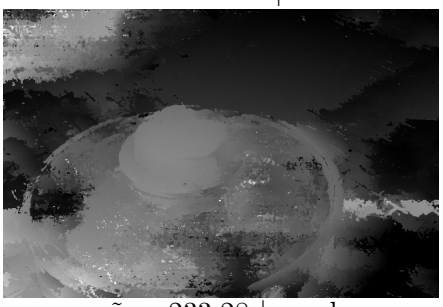


$\tilde{m} = 259.07$ | ok

$\tilde{m} = 256.58$ | ok



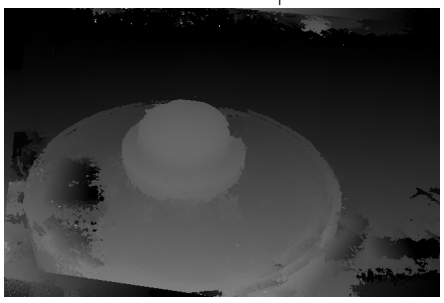


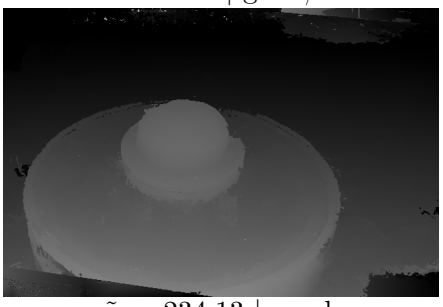


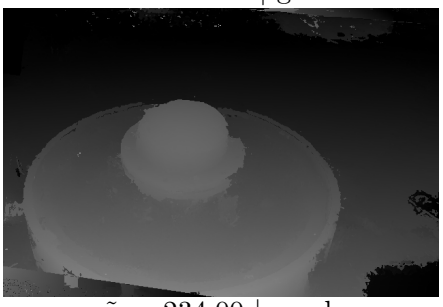
$\tilde{m} = 241.08$ | ok

$\tilde{m} = 234.24$ | good/best

$\tilde{m} = 233.28$ | good

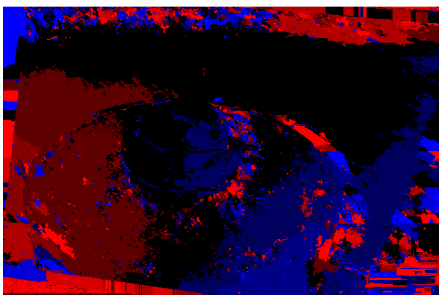


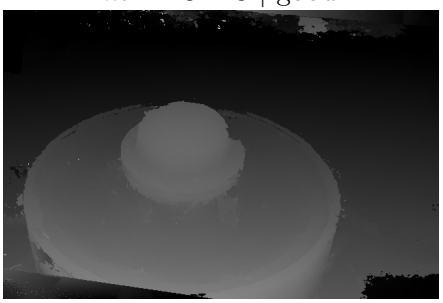


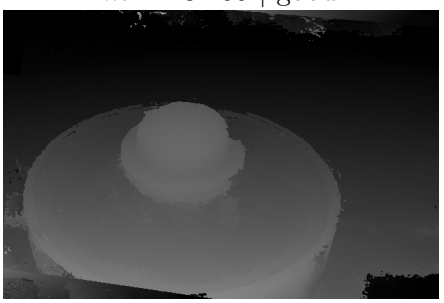


$\tilde{m} = 234.13$ | good

$\tilde{m} = 234.00$ | good







Bibliography

[Bar+09] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing". In: *ACM SIGGRAPH 2009 Papers*. SIGGRAPH '09. New Orleans, Louisiana: ACM, 2009, 24:1–24:11. ISBN: 978-1-60558-726-4. DOI: 10.1145/1576246.1531330.

[BK08] Gary Bradski and Adrian Kaehler. *Learning OpenCV. Computer Vision with the OpenCV Library*. First. O'Reilly, Sept. 2008. ISBN: 978-0-596-51613-0.

[Bou13] Jean-Yves Bouget. *Camera Calibration Toolbox for Matlab*. Dec. 2, 2013. URL: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html (visited on 11/13/2014).

[BRR11] Michael Bleyer, Christoph Rhemann, and Carsten Rother. "PatchMatch Stereo - Stereo Matching with Slanted Support Windows". In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2011, pp. 14.1–14.11. ISBN: 1-901725-43-X. DOI: 10.5244/C.25.14.

[Cap12] Ugo Capeto. *Stereo Matching - Global Methods*. Jan. 4, 2012. URL: http://3dstereophoto.blogspot.de/2012/01/stereo-matching-global-methods.html (visited on 03/10/2015).

[CM02] D. Comaniciu and P. Meer. "Mean shift: a robust approach toward feature space analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.5 (May 2002), pp. 603–619. ISSN: 0162-8828. DOI: 10.1109/34.1000236.

[Com15] Computer Vision Lab Dresden. *Welcome to the Computer Vision Lab Dresden*. Technische Universität Dresden. 2015. URL: http://cvlab-dresden.de (visited on 03/12/2015).

[CP11] Antonin Chambolle and Thomas Pock. “A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging”. In: *Journal of Mathematical Imaging and Vision* 40.1 (2011), pp. 120–145. ISSN: 0924-9907. DOI: 10.1007/s10851-010-0251-1.

[Daw14] Kenneth Dawson-Howe. *A Practical Introduction to Computer Vision with OpenCV*. First. John Wiley & Sons Ltd., 2014. ISBN: 978-1-118-84845-6.

[FB81] Martin A. Fischler and Robert C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Communications of the ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692.

[FI11] Andrea Fusiello and Luca Irsara. “Quasi-Euclidean epipolar rectification of uncalibrated images”. English. In: *Machine Vision and Applications* 22.4 (2011), pp. 663–670. ISSN: 0932-8092. DOI: 10.1007/s00138-010-0270-3.

[Geu+01] J.-M. Geusebroek, R. Van den Boomgaard, A.W.M. Smeulders, and H. Geerts. “Color invariance”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.12 (Dec. 2001), pp. 1338–1350. ISSN: 0162-8828. DOI: 10.1109/34.977559.

[Har99] Richard I. Hartley. “Theory and Practice of Projective Rectification”. In: *International Journal of Computer Vision* 35.2 (Nov. 1999), pp. 115–127. ISSN: 0920-5691. DOI: 10.1023/A:1008115206617.

[Hei+13] P. Heise, S. Klose, B. Jensen, and A. Knoll. “PM-Huber: PatchMatch with Huber Regularization for Stereo Matching”. In: *IEEE International Conference on Computer Vision (ICCV), 2013*. Dec. 2013, pp. 2360–2367. DOI: 10.1109/ICCV.2013.293.

[Hir08] H. Hirschmuller. “Stereo Processing by Semiglobal Matching and Mutual Information”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 30.2 (Feb. 2008), pp. 328–341. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2007.1166.

[Hos+09] A. Hosni, M. Bleyer, M. Gelautz, and C. Rhemann. “Local stereo matching using geodesic support weights”. In: *16th IEEE International Conference on Image Processing (ICIP), 2009*. Nov. 2009, pp. 2093–2096. DOI: 10.1109/ICIP.2009.5414478.

[Hub73] Peter J. Huber.
"Robust Regression: Asymptotics, Conjectures and Monte Carlo".
In: *The Annals of Statistics* 1.5 (Sept. 1973), pp. 799–821.
DOI: 10.1214/aos/1176342503.

[HZ04] Richard I. Hartley and Andrew Zisserman.
Multiple View Geometry in Computer Vision. Second.
Cambridge University Press, 2004. ISBN: 978-0-521-54051-3.

[Jia+14] Jianbo Jiao et al. "Cost-volume filtering-based stereo matching with improved matching cost and secondary refinement".
In: *IEEE International Conference on Multimedia and Expo (ICME), 2014*.
July 2014, pp. 1–6. DOI: 10.1109/ICME.2014.6890201.

[Khr11] Khronos OpenCL Working Group. *The OpenCL Specification*.
Ed. by Aaftab Munshi. Version 1.1. June 1, 2011.
URL: https://www.khronos.org/registry/cl/specs/opencl-1.1.pdf (visited on 01/22/2015).

[Lon81] H. C. Longuet-Higgins.
"A computer algorithm for reconstructing a scene from two projections".
In: *Nature* 293.5828 (Sept. 10, 1981), pp. 133–135. ISSN: 0028-0836.
DOI: 10.1038/293133a0.

[LZ99] Charles T. Loop and Zhengyou Zhang.
"Computing Rectifying Homographies for Stereo Vision."
In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 1. IEEE Computer Society, 1999, pp. 125–131. ISBN: 0-7695-0149-4. DOI: 10.1109/CVPR.1999.786928.

[Mar63] Donald W. Marquardt.
"An Algorithm for Least-Squares Estimation of Nonlinear Parameters".
In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441. ISSN: 0368-4245. DOI: 10.1137/0111030.

[Mon11] Pascal Monasse. "Quasi-Euclidean Epipolar Rectification".
In: *Image Processing On Line* 1 (2011).
DOI: 10.5201/ipol.2011.m_qer.
URL: http://dx.doi.org/10.5201/ipol.2011.m_qer (visited on 02/28/2015).

[MSK14] Veldandi Muninder, Ukil Soumik, and Govindarao Krishna.
"Robust segment-based Stereo using Cost Aggregation".
In: *Proceedings of the British Machine Vision Conference*.
BMVA Press, 2014.

[MY09] Jean-Michel Morel and Guoshen Yu.
"ASIFT: A New Framework for Fully Affine Invariant Image Comparison".
In: *SIAM Journal on Imaging Sciences* 2.2 (Apr. 2009), pp. 438–469.
ISSN: 1936-4954. DOI: 10.1137/080732730.

[Nis04] David Nistér.
"An Efficient Solution to the Five-Point Relative Pose Problem".
In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.6 (June 2004), pp. 756–777. ISSN: 0162-8828.
DOI: 10.1109/TPAMI.2004.17.

[Ode15] Marlene Odenbach. *TU Dresden: University of Excellence.*
Ed. by Cindy Ullmann. PowerPoint slides.
Technische Universität Dresden, Feb. 20, 2015.
URL: http://tu-dresden.de/die_tu_dresden/portrait/zahlen_und_fakten/daten/150223_TUD_Excellence_en.pptx (visited on 03/12/2015).

[PKV99] M. Pollefeys, R. Koch, and L. Van Gool.
"A simple and efficient rectification method for general motion".
In: *The Proceedings of the Seventh IEEE International Conference on Computer Vision.* Vol. 1. 1999, pp. 496–501.
DOI: 10.1109/ICCV.1999.791262.

[Rhe+11] C. Rhemann et al.
"Fast cost-volume filtering for visual correspondence and beyond".
In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on.* June 2011, pp. 3017–3024.
DOI: 10.1109/CVPR.2011.5995372.

[Ric+10] Christian Richardt et al. "Real-Time Spatiotemporal Stereo Matching Using the Dual-Cross-Bilateral Grid". In: *Computer Vision – ECCV 2010.*
Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Vol. 6313.
Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 510–523. ISBN: 978-3-642-15557-4.
DOI: 10.1007/978-3-642-15558-1_37.

[Rot14] Carsten Rother. *Computer Vision I - Introduction.* lecture slides.
Oct. 21, 2014.
URL: http://wwwpub.zih.tu-dresden.de/~cvweb/teaching/Courses/WS_2014_15/CV1/VL1.pdf (visited on 03/03/2015).

[RTV13] Richard Rzeszutek, Dong Tian, and Anthony Vetro. “Disparity estimation of misaligned images in a scanline optimization framework”.
In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. May 2013, pp. 1523–1527.
DOI: 10.1109/ICASSP.2013.6637906.

[San12] Thiago Santos. *Heavy shearing effects using Hartley's rectification*.
July 23, 2012.
URL: http://answers.opencv.org/question/418/heavy-shearing-effects-using-hartleys-rectification/?answer=677#post-id-677 (visited on 03/04/2015).

[Sch05] Oliver Schreer. *Stereoanalyse und Bildsynthese*. German.
Springer-Verlag Berlin Heidelberg, 2005. ISBN: 3-540-23439-X.

[Sim12] Kyle Simek. *Dissecting the Camera Matrix, Part 2: The Extrinsic Matrix*.
Aug. 22, 2012.
URL: http://ksimek.github.io/2012/08/22/extrinsic/ (visited on 10/17/2014).

[SS02] Daniel Scharstein and Richard Szeliski. “A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms”.
In: *International Journal of Computer Vision* 47.1-3 (2002), pp. 7–42.
ISSN: 0920-5691. DOI: 10.1023/A:1014573219977.

[TMN14] T. Taniai, Y. Matsushita, and T. Naemura. “Graph Cut Based Continuous Stereo Matching Using Locally Shared Labels”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014*. June 2014, pp. 1613–1620. DOI: 10.1109/CVPR.2014.209.

[Tre98] Götz Trenkler. “82.15 Four Square Roots of the Vector Cross Product”.
In: *The Mathematical Gazette* 82 (Mar. 1998), pp. 100–102. URL: http://www.jstor.org/stable/3620167 (visited on 11/05/2014).

[Tsa87] Roger Y. Tsai. “A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses”.
In: *IEEE Journal of Robotics and Automation* 3.4 (Aug. 1987), pp. 323–344.
ISSN: 0882-4967. DOI: 10.1109/jra.1987.1087109.

[Wei02] Eric W. Weisstein. *Matrix Rank*.
From MathWorld–A Wolfram Web Resource. Nov. 16, 2002.
URL: http://mathworld.wolfram.com/MatrixRank.html (visited on 02/28/2015).

[Wes14] Bettina Weser. *3D Scene Understanding*. Oct. 7, 2014. URL: http://www.inf.tu-dresden.de/index.php?node_id=3533&ln=de (visited on 03/03/2015).

[YK06] Kuk-Jin Yoon and In So Kweon.
"Adaptive support-weight approach for correspondence search".
In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.4 (Apr. 2006), pp. 650–656. ISSN: 0162-8828.
DOI: 10.1109/TPAMI.2006.70.

[Zha+12] Shihui Zhang et al.
"A Dense Stereo Matching Algorithm Based on Triangulation".
In: *Journal of Computational Information Systems* 8.1 (2012), pp. 283–292.
ISSN: 1553-9105.

Selbstständigkeitserklärung gem. § 22 Absatz 5 BPO

Hiermit versichere ich, Dominik Wetzel, dass ich die vorliegende Master-Thesis mit dem Titel

Enhanced depth estimation with "freehand stereo" using PatchMatch Stereo

selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen wurden, sowie Bilder, Tabellen und Listings, die anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Zwickau, den 12. März 2015

Ort, Datum

Unterschrift