FREIE UNIVERSITÄT BERLIN

MASTER'S THESIS

Structure from Motion Based on Monocular Image Sequences with a Virtual Stereo Camera

Author:
Johannes SAUER
Matr.Nr. 4473561

Supervisor & 1st Examiner:
Prof. Dr. Raúl
ROJAS GONZÁLEZ

Advisor:
Tobias LANGNER

2nd Examiner:
Prof. Dr. Daniel
GÖHRING

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

at the

Department of Mathematics and Computer Science

10th April, 2019

Selbstständigkeitserklärung

Name	Sauer
Vornamen	Johannes Theo
geb. am	08.02.1990
Matr.Nr.	4473561

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keiner anderen Universität als Prüfungsleistung eingereicht.

Datum

Unterschrift

FREIE UNIVERSITÄT BERLIN

Abstract

Department of Mathematics and Computer Science

Master of Science

Structure from Motion Based on Monocular Image Sequences with a Virtual Stereo Camera

by Johannes SAUER

Stereo matching (SM) is a well researched method for generating depth information from camera images. Efficient implementations of SM algorithms exist as part of widely used computer vision libraries, such as OpenCV. Typically, SM is being performed on pairs of images from a stereo camera in which intrinsic and extrinsic parameters are fixed and determined in advance by calibration. Furthermore, the images are usually taken at approximately the same time by triggering the shutters simultaneously. In this thesis a different approach is being pursued: stereo pairs are selected from a video sequence of a monocular camera, which is mounted on a moving vehicle. Two scenarios are covered: one where the camera is facing sidewards and one where it is facing forwards in relation to the driving direction. Extrinsic transformations between frames are computed by visual odometry. Images out of a series can each be rectified with the same reference image; the resulting image pairs are therefore effectually taken by a virtual stereo camera with variable baseline. Stereo matching and three-dimensional reconstruction can be applied to these images in the same way as to those of a binocular camera with fixed extrinsic calibration. Apart from the development of the virtual stereo principle itself, two main contributions have been developed in this thesis: Firstly, it has been shown that the fusion of disparity images (according to Hirschmüller [1]) taken at varying baselines improves quality in terms of density and error rate. Secondly, a new rectification procedure has been developed for the scenario of the forward facing camera; here the standard procedure developed for conventional stereo cameras is not applicable.

Acknowledgements

Thanks go to Prof. Dr. Rojas for supervising this thesis and Tobias Langner for his additional advice and feedback. Furthermore, I am grateful for the advice and support from my colleagues at TomTom, especially Bennet Fischer.

Contents

List of Figures

List of Tables

List of Abbreviations

DSO	Direct Sparse Odometry
LIDAR	Light Detection and Ranging
ROC	Receiver Operating Characteristic
SfM	Structure from Motion
SGBM	Semi-Global Block Matching
SLAM	Simultaneous Localization and Mapping
SM	Stereo Matching
VO	Visual Odometry
VS	Virtual Stereo

Chapter 1

Introduction

This thesis develops a new technique to solve the *Structure from Motion (SfM)* problem in the field of computer vision. A recent survey paper due to Özyeşil et al. [2] defines SfM as the recovery of three-dimensional structure of a stationary scene from a set of two-dimensional images via estimation of motion of the cameras corresponding to these images. The *Virtual Stereo (VS)* method, which is the main contribution of this thesis, fits this definition quite well, but at the same time it could be seen as an instance of a method for solving the *Simultaneous Localization and Mapping (SLAM)* problem. In [2], SLAM is defined as a form of SfM specific to robotics. It performs the reconstruction of three-dimensional structure on a video that is filmed by a (robotic) vehicle moving through the world. Additionally to a wide ranging recovery of the environment along the route of the vehicle, it is at the same time being localized in this recovered environment. There are usually also non-functional requirements for a SLAM implementation, e.g. that the processing happens on-line, meaning the processing proceeds immediately on images as they are being recorded, ideally with a low delay, so that the robot can react quickly to the resulting output data. Virtual Stereo, in principle, would fit the definition of SLAM, but it would be somewhat presumptuous to call it a fully fledged solution to SLAM. The reason for this lies in the fact that VS uses the localization data of another SLAM and *Visual Odometry (VO)* solution, namely *Direct Sparse Odometry (DSO)*, for reconstruction of three-dimensional structure; localization is not performed by VS itself. VS can rather be seen as a technique that uses motion information coming from DSO to augment its sparsely reconstructed structure by a denser one computed via *Stereo Matching (SM)*. Therefore, the title contains the more general formulation of SfM.

1.1 Motivation

The main motivation for the design of VS was the reuse of the preexisting techniques of visual odometry and stereo matching for the development of a SfM solution. In this approach the single parts are independent of each other, e.g. we could exchange DSO for a different VO implementation. This makes VS more modular and adaptable than other SfM methods which use bundle adjustment techniques that simultaneously determine motion and 3D reconstruction on longer sequences of images. Additionally, reusing highly optimized techniques as parts of VS enables a low latency implementation of SfM, e.g. it would be possible to use a stereo matching implementation on specialized hardware, such as a field programmable gate arrays. (It should be noted that the prototype that was implemented as part of this thesis is only a proof of the VS concept, but does not perform on-line on video sequences.) An advantage of other SfM methods comes from the use of longer sequences for structure recovery. Using a greater number of images can lead to a higher accuracy and greater density of the reconstructed structures, and in fact this can also be achieved in VS, namely by fusing the disparity images obtained via SM.

1.2 Contributions

Apart from the proof of concept of the VS principle on multiple real world scenarios, this thesis makes two further main contributions. Firstly, a new rectification procedure for stereo matching has been developed. The rectification of images taken with cameras that are displaced laterally to each other has been solved for a long time, e.g. a compact implementation in MATLAB code has been presented by Fusiello, Trucco, and Verri [3]. In this thesis we also explore the VS principle in a scenario where camera frames are displaced along the principal axis instead, i.e. one camera is behind the other. Rectification for this case can not be easily adapted from the conventional method, which assumes that cameras are laterally displaced, because the epipolar geometry of this new scenario does not allow whole images to be rectified. The novel rectification procedure that has been developed for this case rectifies selectable regions of images on which stereo matching proceeds subsequently. The resulting disparity images of these regions can finally be transfered back in the original perspective, so that we obtain disparity images of the same kind as with a conventional stereo rig.

The second main contribution is the application of disparity image fusion to the VS method. The fusion method, proposed by Hirschmüller in [1], combines disparity images based on the same reference image in order to obtain a fused result with improved density and accuracy. Even though the method has been applied in [1], the claims of improvement as opposed to ordinary disparity images were not underpinned by a numerical evaluation there. In this thesis, Hirschmüller's disparity fusion is evaluated numerically in the framework of a *Receiver Operating Characteristic (ROC)* relating the density with error rate of disparity images. Isolated from other parts of VS, which could introduce unrelated errors, the evaluation takes place on pre-rectified images of the Middlebury multi-view dataset of 2006 due to Hirschmüller, Scharstein, and Pal [4, 5]. We were able to confirm Hirschmüller's claims based on these data.

1.3 Related Work

SfM is a long-standing research topic in the area of computer vision. The survey paper [2] gives a good overview of current research in SfM and SLAM. An example for a state-of-the-art SLAM algorithm is *Large Scale Direct monocular SLAM (LSD-SLAM)* of Engel et al. [6]. It has commonalities with DSO (e.g. both algorithms compute photometric error directly on images), but it delivers a denser reconstruction of the scenery. As usual for SLAM and SfM algorithms, including also DSO, it jointly estimates camera poses and depth reconstruction. Algorihtms that separate these two steps into two independent parts, as in our VS method, seem to be out of the ordinary. One paper that used a similar approach is due to Sato et al. [7]. The method presented there has multiple similarities with ours: it estimates extrinsic camera transformations first and then computes disparity images, also with a multi-baseline approach, for reconstruction of the scenery. However, [7] is somewhat cursory and does not go deeply into the details of the method proposed there, as is natural for workshop papers. The lack of further research following this approach is reason to give it a more in-depth investigation which we did in this thesis.

1.4 Outline

Chapter 2 develops the basics of projective geometry and explains how these can be applied for the description of a camera in the framework of computer vision, namely in form of the pinhole camera model.

Chapter 3 continues the development of this model by extending it with another camera; the resulting epipolar geometry is the foundation for rectification. This chapter also treats standard rectification, stereo matching, and three-dimensional reconstruction as parts of the conventional stereo vision pipeline.

Essentially the same pipeline is applied in the Virtual Stereo method proposed in chapter 4. The first main difference between the conventional method, used for fixed stereo rigs, and VS is the source and type of extrinsic transformations: for a fixed rig it is static and needs to be computed only once, for VS it is obtained from VO and different for every pair of frames. The second difference is the time at which images are recorded: synchronously for the camera pair of a rig, and subsequently in time in the case of VS. Chapter 4 treats two scenarios to which we apply the VS method; in the first, where the camera is facing sidewards in relation to the vehicle on which it is mounted, no additional modifications need to be made to the stereo vision pipeline. The fusion of disparity images is demonstrated in this scenario, because images with this perspective lend themselves well to this additional refinement step. For the second scenario, where the camera is facing forwards in relation to the vehicle, the new rectification procedure is developed.

Chapter 5 evaluates the results of VS both qualitatively and quantitatively. In the evaluation of the front-facing scenario, scene reconstructions of a fixed rig were compared with those of VS. The reference images for stereo matching were identical for both methods, so that we could also compare disparity images of the two methods pixel-wise. Even though no ground truth was available, this comparison established that front-facing VS can deliver similar results to those of well-tried conventional methods. The second evaluation concerned the fusion of disparity images. This component of VS was tested in isolation on a data set recorded under laboratory conditions. The predicted improvements in accuracy and density, when compared to unfused disparity images, were confirmed via a ROC analysis.

The conclusion in chapter 6 gives an outlook of possible further developments and improvements of VS.

Chapter 2

Camera Geometry

The following chapter lays the groundwork for the Virtual Stereo method presented in chapter 4. These foundations do not differ from those on which conventional stereo matching with fixed extrinsics is based, and therefore they will be relevant both for the understanding of the chapter on stereo matching (chapter 3) as well as the new Virtual Stereo method, which also builds on conventional stereo matching.

One of the most fundamental necessities for stereo vision, and computer vision in general, is to describe how points in the three-dimensional world project onto a two-dimensional image plane. Projective geometry enables us to formulate this projection between two and three-dimensional spaces in a concise and comprehensive manner. The basic elements of projective geometry, necessary for the subsequent treatment of imaging and stereo matching, are presented in the first section of this chapter.

The second section introduces the most widely used projection model for cameras, the pinhole camera model. This model neglects distortion and other more complex optical effects, such as chromatic aberration, caused by imperfect optics. Instead of one or multiple lenses for a camera objective, it assumes an idealized point through which the optical rays pass. This makes it easy to formalize projection in terms of simple matrix operations inside the framework of projective geometry.

2.1 Projective Geometry

In [8], Stolfi presents the advantages of projective geometry for computer graphics and vision: parallel lines and planes do not need to be treated as special cases, as it is the case in Euclidean geometry, and often division operations can be avoided in formulas of homogeneous coordinates, as opposed to the standard Cartesian ones of Euclidean geometry. For the sake of brevity, only the most important elements of projective geometry will be introduced in this thesis. Therefore, *oriented projective geometry* , as presented in [8], will not be treated explicitely in this chapter. Only in later chapters and when it becomes necessary the orientation of the elements of the geometry will be taken into account.

2.1.1 Homogeneous Coordinates

The basic objects of projective geometry are points, lines, planes, and other subspaces of the ambient space, same as in Euclidean geometry. The objects of the Euclidean space represented by Cartesian coordinates in $\mathbb{R}^n$ correspond to objects with an additional dimension and an equivalence relationship in projective space. This corresponding projective space is denoted by $\mathbb{P}^n$.

While the representation of choice for objects in Euclidean space in this thesis are Cartesian coordinates, objects in projective space will be represented in homogeneous coordinates. Before continuing with the definition of homogeneous coordinates, it should

be noted that we will limit ourselves to spaces with $n = 2$ for simplicity's sake, and extend the notions developed therein to spaces with $n = 3$ later on, when it becomes necessary.

Definition 2.1 (Homogeneous Coordinates)**.** The homogeneous coordinates of a point with Cartesian coordinates $(x\ y)^T$ in the two-dimensional real plane $\mathbb{R}^2$ are $(x\ y\ 1)^T$ in $\mathbb{P}^2$. The tuple $(x, y\ 1)^T$ is equivalent to any other tuple of homogeneous coordinates $(\lambda x\ \lambda y\ \lambda)^T$ with $\lambda \in \mathbb{R}, \lambda \neq 0$. This equivalence relation is denoted by $(x\ y\ 1)^T \sim (\lambda x\ \lambda y\ \lambda)^T$.

The homogeneous coordinates of a line with equation $f(x, y) = ax + by + c = 0$ (again in $\mathbb{R}^2$) are $(a\ b\ c)^T$. As with points, these homogeneous line coordinates are equivalent to any other tuple $(\lambda a\ \lambda b\ \lambda c)^T$ with $\lambda \in \mathbb{R}, \lambda \neq 0$, which is denoted by the same equivalence relation $\sim$.

The points and lines of $\mathbb{P}^2$ are therefore the equivalence classes of $\sim$ on members of $\mathbb{R}^3 \setminus \{(0\ 0\ 0)^T\}$. We will write points and lines in homogeneous coordinates by picking one representative of the respective class. This is of course an abuse of notation, but usually it will be clear from context whether the vector of the point or line is meant to be in $\mathbb{R}^3$ or $\mathbb{P}^2$. If not, the distinction will be made explicit. The reason for excluding the zero vector will become obvious in section 2.1.2.

Representing some arbitrary homogeneous point $(x\ y\ w)^T$ in Cartesian coordinates is easily done by the mapping $(x\ y\ w)^T \mapsto (x/w\ y/w)^T$. It is clear that not all points in $\mathbb{P}^2$ can be mapped into $\mathbb{R}^2$. Namely those with $w = 0$ do not have corresponding points in $\mathbb{R}^2$. Similarly, the homogeneous coordinates $(0\ 0\ 1)^T$ can not fulfill any line equation in $\mathbb{R}^2$. This is not a fault, but one of the main advantages of projective geometry over Euclidean geometry, as many special cases are precluded thereby. We will return to these special points in section 2.1.2, but before that I will describe the relationships between lines and points in $\mathbb{P}^2$.

Corollary 2.1 (Line intersecting point)**.** A line $l = (a\ b\ c)^T$ intersects a point with Cartesian coordinates $(x\ y)^T$ if, and only if, $ax + by + c = 0$. Clearly this equality still holds when we multiply the left hand side with any $\lambda \in \mathbb{R}, \lambda \neq 0$. Therefore, when we use the homogeneous representation $p = (x\ y\ 1)^T$ of the point, this criterion becomes $\langle p, l \rangle = \langle l, p \rangle = 0$ with $\langle . \rangle$ being the inner (scalar) product.

Corollary 2.2 (Line meeting line)**.** Given two lines l_1, l_2 in homogeneous coordinates, the cross product $p = l_1 \times l_2$ is a vector that is orthogonal to both vectors representing l_1 and l_2. Therefore, $\langle p, l_1 \rangle = \langle p, l_2 \rangle = 0$. From corollary 2.1 it now follows that p is the point where l_1 and l_2 intersect.

Note that there is no special treatment for lines that coincide or are parallel to each other. We can still compute the intersection of those, even though we know that these points can not exist in Euclidean space. In section 2.1.2 we will take a closer look at this apparent problem.

Corollary 2.3 (Line joining points)**.** In a very similar fashion to the argument of corollary 2.2 let p_1, p_2 be two points in homogeneous coordinates and $l = p_1 \times p_2$ their cross product. Then $\langle l, p_1 \rangle = \langle l, p_2 \rangle = 0$, and from corollary 2.1 it follows that l is the line that joins p_1 and p_2.

2.1.2 Points and Lines at Infinity

In section 2.1.1 we saw that the points with $w = 0$ in $\mathbb{P}^2$ can not be mapped into the plane $\mathbb{R}^2$, and that the points of intersection of parallel lines can not lie in Euclidean space either. When we take two parallel lines $l_1 = (a\ b\ c)^T$ and $l_2 = (a'\ b'\ c')^T$, their

common slope is $s = -a/b = -a'/b'$. Therefore, in the point of intersection of these lines, $p = l_1 \times l_2$, the third coordinate is $ab' - ba' = 0$. We see that these two classes of points outside of $\mathbb{R}^2$ coincide. According to the slogan that "parallel lines meet at infinity" they are called *points at infinity*.

Definition 2.2 (Points at infinity)**.** Points of the shape $(x\ y\ 0)^T, x, y \in \mathbb{R}$ are said to lie *at infinity*.

In order to get a better understanding what it means for a point to be "at infinity" in projective space, it is helpful to look at how $\mathbb{R}^2$ and $\mathbb{P}^2$ relate exactly. Let us return to the definition of points in $\mathbb{P}^2$ (definition 2.1) and consider them as the equivalence classes under scaling of position vectors in $\mathbb{R}^3$. Then a point of $\mathbb{P}^2$ can be seen as a line through the origin $(0\ 0\ 0)^T \in \mathbb{R}^3$, where all differently scaled representatives of the point lie on this line. Considering the mapping of a point $(x\ y)^T$ in Cartesian coordinates to homogeneous coordinates $(x\ y\ 1)^T$, it becomes clear that the plane $\mathbb{R}^2$ is embedded into the space $\mathbb{R}^3$ of $\mathbb{P}^2$ as the plane with $z = 1$. The line through all representatives of the point $(x\ y\ 1)^T \in \mathbb{P}^2$ joins $(0\ 0\ 0)^T \in \mathbb{R}^3$ and the standard representative $(x\ y\ 1)^T \in \mathbb{R}^3$. For a point at infinity this means that all its representatives lie on a line on the plane with $z = 0$ passing through the origin of $\mathbb{R}^3$. Apart from having no correspondences in $\mathbb{R}^2$, points at infinity are no special cases in $\mathbb{P}^2$, and so the notions of intersection, joining, and meeting apply to them as they did to ordinary points.

Definition 2.3 (Line at infinity)**.** Given two points at infinity $p_1 = (x\ y\ 0)^T, p_2 = (x'\ y'\ 0)^T$, the line joining these two, $l_\infty := p_1 \times p_2 = (0\ 0\ xy' - yx')^T$, is called line at infinity. $l_\infty \sim (0\ 0\ 1)^T$ is uniquely determined in $\mathbb{P}^2$ and will usually be represented by $(0\ 0\ 1)^T$.

The vector $(0\ 0\ 0)^T$ was previously excluded from $\mathbb{P}^2$, and it should now be clear that it would not correspond to any point, as by corollary 2.1 it would have to lie on all lines.

So far, the points and line at infinity have little use, apart from avoiding undefined cases when intersecting parallel lines. Soon enough we will see their true usefulness though. As a small preview of this, consider the following method for constructing points at infinity. Taking an arbitrary point $p = (x\ y\ 1)^T \in \mathbb{P}^2$ and adding to it the vector $d = (d_1\ d_2\ 0)^T$, we move it into a direction on the plane. This direction stays the same when we add multiples of d to p; we obtain $p + \lambda d, \lambda \in \mathbb{R}$. In the limit process where $\lambda \to \infty$ this term will converge towards the point at infinity d, as the last coordinate converges to 0.

From a geometrical point of view, this means that the direction towards a point at infinity is invariant under translation; any starting point on the plane will move towards d. This is not true for other points on the plane. The direction towards them depends on the position of the starting point too. Therefore, points at infinity can be considered as directions. An analogy between points of infinity in $\mathbb{P}^3$ and fixed stars can be drawn, as the latter appear always (approximately) in the same direction relative to the earth.

2.2 The Pinhole Camera Model

As mentioned in the introduction to this chapter, the pinhole camera model is an abstraction of an actual camera where complex optical effects are neglected. Even though distortion effects can be considerable, they are also relatively easy to revert. In this thesis it is always being assumed that distortion effects of lenses have been reverted before any further processing of the images proceeds. Other effects, like chromatic aberration, are neglected, because all our recordings were made with lenses that minimize these effects to a suitable degree.

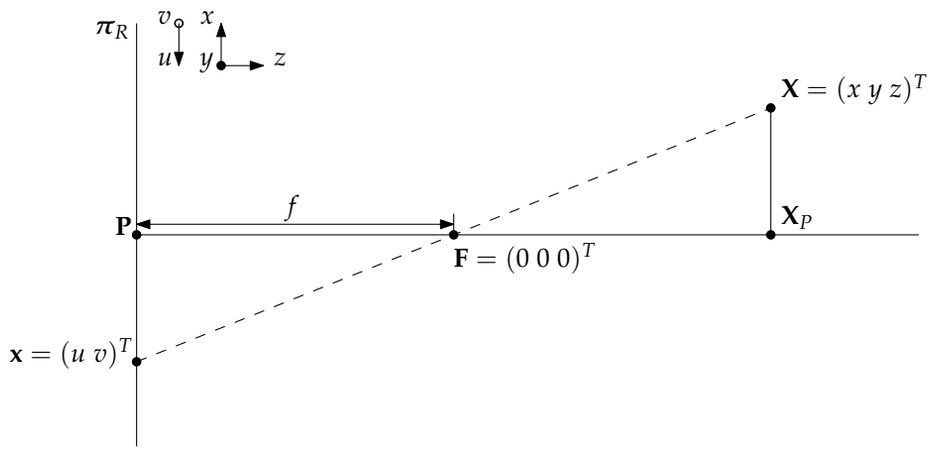


FIGURE 2.1: Projection through pinhole onto retinal plane

2.2.1 Derivation of the Intrinsic Camera Matrix K

The pinhole camera model describes how the light reflected from points in the three-dimensional world is projected onto the image sensor of a camera. Usually image sensors are planar, and, analogous to the retina of the eye, this plane of the sensor is called *retinal plane* $\boldsymbol{\pi}_R$ in the model. All rays of light that get captured by the sensor pass through a single point, the *focal point*. The focal point is an idealized formalization of the effect of a lens on rays of light. This thesis will only treat cameras that have fixed lenses, i.e. the lens and respectively the focal point will have a constant distance to the retinal plane, called *focal length*. One of the most desirable qualities of the pinhole camera model is its simplicity, and so almost all elments for describing the projection are already there.

First of all, we make a distinction between two different coordinate systems (also called coordinate frames); one for determining points in the world and one for points on the retinal plane. In the first derivation of the projection formula the orientation of the coordinate systems and their origins lie in a special configuration, so that we will arrive at an especially simple form of projection. This constraint says specifically that the camera coordinate frame coincides with the world coordinate frame. Later on, in section 2.2.2, a distinction between world and camera coordinates will become necessary to generalize the model introduced in this subsection.

The dimensions of the *world coordinate system* are denoted by the usual Cartesian coordinates x, y, and z. The z-axis is chosen so that it is normal to the retinal plane and its positive direction extends away from this plane. The origin of the world coordinate system is picked so that it coincides with the focal point $\mathbf{F}$. (Points in the world coordinate system will be written in bold uppercase letters.) The point where the z-axis intersects the retinal plane is called *principal point* $\mathbf{P}$; the axis extending from $\mathbf{P}$ through $\mathbf{F}$ is generally called *principal axis*. (In our special case this axis is identical to the z-axis, but this does not hold for general world coordinates.) The distance $|\mathbf{PF}|$ is the focal length f and so $\mathbf{P} = (0\ 0\ -f)^T$, in our special case. The x and y-axes are oriented in a right hand coordinate system relative to the z-axis. This means that with the thumb, index, and middle finger of a right hand, all stretched out orthogonally to each other, the thumb points into positive z, the index into positive x, and the middle finger into positive y-direction.

As the z-axis is normal to the retinal plane, it would seemingly be possible to describe points on the plane in x and y-coordinates of the world coordinate system. This is, however, only true for the special choice of coordinate system that was made. In

order to describe image coordinates independently of world coordinates, a second coordinate system, the *retinal* or *image coordinate system*, is used. Its dimensions are denoted by u and v; they extend on the retinal plane so that the positive u-direction is opposite to the positive x-direction, and the positive y-direction opposite to the positive u-direction. The reason for choosing opposite directions between world and retinal coordinates will become clear quite soon. For now the origin of the retinal coordinate system will coincide with **P**, but this constraint will be suspended also.

Definition 2.4. We call the coordinate system where

- the focal point has coordinates $(0\ 0\ 0)^T$ in the world,
- u- and x-axes, and respectively v- and y-axes extend into opposite directions,
- and the z-axis extends in direction of the normal of $\boldsymbol{\pi}_R$, towards the focal point

the *reference camera* coordinate system.

Sometimes we will let world coordinates refer to the corresponding retinal coordinates, or vice versa. This abuse of notation should not lead to confusion though, as the ambient coordinate system will be clear from context.

Finally, we can derive the projection equation for a point in the world $\mathbf{X} = (x\ y\ z)^T$ to its its retinal coordinates $\mathbf{x} = (u\ v)^T$. (Points on the retinal plane will be denoted by bold lowercase letters.) In fig. 2.1 the ray of light that passes from **X** through **F** onto the retinal plane at **x** is drawn as a dashed line. The diagram shows the x, z-plane of world coordinates with the y-axis extending towards the observer. Additionally to the elements introduced above, the *base point* $\mathbf{X}_P$ of **X** is shown in the diagram. From this perspective, we can see the relation between the coordinates u, x, and z by taking into consideration the two similar [1] triangles $\triangle\mathbf{xPF} \sim \triangle\mathbf{XX}_P\mathbf{F}$:

$$u = \frac{f}{z}x \tag{2.1}$$

By considering the y, z-plane we can derive the following formula by an entirely analogous consideration:

$$v = \frac{f}{z}y \tag{2.2}$$

Expressed together as vectors we obtain the following equation:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{z}\begin{pmatrix} x \\ y \end{pmatrix} \tag{2.3}$$

We now see that choosing opposite directions of coordinate systems avoids a change of signs between world and image coordinates. Another way of achieving the same is to virtually position the focal point behind the retinal plane while letting corresponding axes of world and retinal coordinates extend into the same directions. This can, however, lead to confusion, as in that model the rays of light would pass through the plane before meeting the focal point. We shall therefore continue with opposite directions of coordinate systems. Translating the equation into homogeneous coordinates will make it easy to derive projection as a matrix multipliation, as we shall see in a bit.

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f}{z}x \\ \frac{f}{z}y \\ 1 \end{pmatrix} = \frac{f}{z}\begin{pmatrix} x \\ y \\ \frac{z}{f} \end{pmatrix} \tag{2.4}$$

[1] The usage of $\sim$ for both equivalence of homogeneous coordinates and similarity of triangles is apt, as both are equivalence relationships up to proportionality between the members (sides and coordinates respectively).

The right hand side term of the previous equation has almost the shape of the world coordinate vector of $\mathbf{X}$, and in homogeneous coordinates it is easy to change this side to a linear transformation with the coordinates of $\mathbf{X}$ as argument:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{pmatrix} x \\ y \\ \frac{z}{f} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.5}$$

Just as homogeneous coordinates are equivalent up to scale, the same holds for linear transformations that operate on them. For representing image and world points, homogeneous and standard Cartesian coordinates will be used interchangeably when it is clear from context which ones are meant. The newly introduced *camera matrix* K can be, and usually is, expressed in the following form:

$$K := \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.6}$$

$$\Rightarrow \qquad \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim K \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.7}$$

The camera matrix has so far only one degree of freedom, because we have introduced multiple constraints, mostly on the situation of the coordinate systems. Now we will remove these constraints and see how the additional free parameters will effect the form of the matrix.

The first constraint to go is the coincidence of the origin of the two coordinate systems. Usually, the origin of the retinal coordinate system will be chosen to be in a corner of the image sensor, so that the coordinates of image points indicate the pixels in an image matrix. Consequently, we should allow the principal point to be at an arbitrary position, usually designated by the coordinates u_0, v_0. As it will be conventient later on to keep $\mathbf{F}$ at the origin of the world coordinate system, and because it is crucial for the functioning of the pinhole camera model to have the z-axis normal to $\boldsymbol{\pi}_R$, we can no longer abuse notation and let $\mathbf{P}$ denote the principal point in retinal coordinates. Therefore, $\mathbf{p} = (u_0 \ v_0)^T$ shall from now on denote the principal point in image coordinates, while $\mathbf{P} = (0 \ 0 \ -f)^T$ stays the same. Hence, eq. (2.3) now takes the following form:

$$\begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} = \frac{f}{z} \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.8}$$

$$\Leftrightarrow \qquad \begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{z} \begin{pmatrix} x - \frac{z}{f} u_0 \\ y - \frac{z}{f} v_0 \end{pmatrix} \tag{2.9}$$

Applying the same steps as before we obtain for K:

$$K := \begin{bmatrix} f & 0 & -u_0 & 0 \\ 0 & f & -v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.10}$$

$$\Rightarrow \qquad \begin{pmatrix} u_0 + u \\ v_0 + v \\ 1 \end{pmatrix} \sim K \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.11}$$

One further generalization of K shall be made so that the properties of CCD sensors get accurately captured in the model. The single cells of a CCD sensor array are assumed to be ordered in rows and columns orthogonal to each other. Usually, this assumption coincides close enough with the actual design of these sensors, and it is not necessary to introduce a parameter for skew between rows and columns. However, there is usually a slight difference between the height and width of the cells. Assuming that the extent of a cell is c_u, c_v, in units of the u- and v-axis respectively, we obtain the pixel coordinates from image coordinates by simply multiplying them by $1/c_u$, and $1/c_v$ respectively. One convenient way to incorporate this into K is to use two different focal lenghts: $f_u = f/c_u$ and $f_v = f/c_v$. The final formula for K then becomes:

$$K := \begin{bmatrix} f_u & 0 & -u_0 & 0 \\ 0 & f_v & -v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.12}$$

$$\Rightarrow \qquad \begin{pmatrix} (u_0 + u)/c_u \\ (v_0 + v)/c_v \\ 1 \end{pmatrix} \sim K \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.13}$$

2.2.2 Camera Extrinsics and the Projection Matrix P

For the derivation of the camera matrix K we have so far assumed the world coordinates to be the same as those of the camera frame; the camera frame was the reference frame as described in definition 2.4. As long as there is only one camera this poses no problem, but once we want to describe the relative positions and orientations of two cameras in the world frame it is obvious that we can no longer make this simplification; generally not both cameras can be in the same reference position in the world. The assumption of the reference camera frame for the camera position, however, is exactly what rendered the derivation of the projection so simple. Therefore, it is desirable to somehow to reduce general camera positions to the reference case, which is what we set out to accomplish in this section.

The first thing to note about a point $(x\ y\ z)^T$ in world coordinates is that we can rotate it around the origin of its coordinate system by left-multiplying a three-dimensional rotation matrix R with it. Represented in homogeneous coordinates, the point becomes $(x\ y\ z\ 1)^T$ and the homogeneous equivalent of R can be seen in the following equation (**0**, in bold, denotes the zero vector):

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = R \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2.14}$$

$$\Leftrightarrow \qquad \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \left[\begin{array}{c|c} R & \mathbf{0} \\ \hline \mathbf{0}^T & 1 \end{array}\right] \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.15}$$

So far the use of homogeneous coordinates does not seem to add anything, but when

we also want to translate the point by a vector $t = (t_1\ t_2\ t_3)^T$ the expressivity of homogeneous coordinates can be seen in action:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \left[\begin{array}{c|c} R & \mathbf{0} \\ \hline \mathbf{0}^T & 1 \end{array}\right] \left(\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ 0 \end{pmatrix} \right) \tag{2.16}$$

$$\Leftrightarrow \quad \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \left[\begin{array}{c|c} R & t \\ \hline \mathbf{0}^T & 1 \end{array}\right] \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.17}$$

We have just seen that in homogeneous coordinates it is possible to rotate and subsequently translate a vector in a single matrix multiplication. We can also multiply such a matrix to the right of the camera matrix in order to transform points inside of the world coordinate frame before projecting them onto the retinal plane. In this case the transformation matrix is called *extrinsic camera matrix*, as opposed to K, which is the *intrinsic camera matrix*.

This alone does not get us closer to representing the points in the reference camera frame. What we actually need for this is to transform the camera coordinate frame given by the rotation and translation of the camera in the world while leaving the points at their positions in the world. Luckily, the transformation of the coordinate frame, called *passive transformation*, and the transformation of the points in the world, called *active transformation*, are inverse to each other. Let us assume an arbitrary world coordinate frame, and a camera whose focal point is at t_C in this frame, and whose coordinate vectors are rotated by R_C relative to those of the world coordinate vectors. Then the rotation matrix R and the translation vector t that transform the world coordinates of points so that we can treat the camera as if it were in reference configuration are given by this equation:

$$\left[\begin{array}{c|c} R & t \\ \hline \mathbf{0}^T & 1 \end{array}\right] = \left[\begin{array}{c|c} R_C & t_C \\ \hline \mathbf{0}^T & 1 \end{array}\right]^{-1} \tag{2.18}$$

The right hand side can be simplified so that we obtain an explicit form for it:

$$\left[\begin{array}{c|c} R & t \\ \hline \mathbf{0}^T & 1 \end{array}\right] = \left[\begin{array}{c|c} R_C & t_C \\ \hline \mathbf{0}^T & 1 \end{array}\right]^{-1} \tag{2.19}$$

$$= \left(\left[\begin{array}{c|c} I & t_C \\ \hline \mathbf{0}^T & 1 \end{array}\right] \left[\begin{array}{c|c} R_C & \mathbf{0} \\ \hline \mathbf{0}^T & 1 \end{array}\right] \right)^{-1} \tag{2.20}$$

$$= \left[\begin{array}{c|c} R_C & \mathbf{0} \\ \hline \mathbf{0}^T & 1 \end{array}\right]^{-1} \left[\begin{array}{c|c} I & t_C \\ \hline \mathbf{0}^T & 1 \end{array}\right]^{-1} \tag{2.21}$$

$$= \left[\begin{array}{c|c} R_C^T & \mathbf{0} \\ \hline \mathbf{0}^T & 1 \end{array}\right] \left[\begin{array}{c|c} I & -t_C \\ \hline \mathbf{0}^T & 1 \end{array}\right] \tag{2.22}$$

$$= \left[\begin{array}{c|c} R_C^T & -R_C^T t_C \\ \hline \mathbf{0}^T & 1 \end{array}\right] \tag{2.23}$$

Now that we have explicit forms for the intrinsic and extrinsic parts of the complete camera projection matrix we can summarize them in this definition:

Definition 2.5. Given a pinhole camera with intrinsic parameters

- focal lengths f_u and f_v,

- and coordinates of the principal point u_0 and v_0,

as well as the extrinsic parameters

- rotation relative to the world coordinate frame R_C,
- and translation relative to the world coordinate frame t_C

a point $\mathbf{X}$ in homogeneous world coordinates projects to a point $\mathbf{x}$ on the image plane (also in homogeneous coordinates) by the equation

$$\mathbf{x} \sim \underbrace{\begin{bmatrix} f_u & 0 & -u_0 & 0 \\ 0 & f_v & -v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left[\begin{array}{c|c} R_C^T & -R_C^T t_C \\ \hline \mathbf{0}^T & 1 \end{array}\right]}_{P:=} \mathbf{X}$$

The combined intrinsic and extrinsic *projection matrix* is denoted by P. An alternative and simplified definition of the projection matrix that we will use in following chapters is this:

$$P := \begin{bmatrix} f_u & 0 & -u_0 \\ 0 & f_v & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{c|c} R_C^T & -R_C^T t_C \end{array}\right]$$

In following chapters we will often no longer make a distinction between equivalence of homogeneous entities (up to a scale factor) and actual equality of the entries in vectors and matrices. Instead we will write $=$ for both, and only in cases where ambiguities might arise we will distinguish the two relations explicitely again.

Chapter 3

Stereo Matching

Stereo matching is an algorithmic method for extracting depth information out of pairs of images. From this depth information the three-dimensional structure of the recorded scenery can subsequently be reconstructed. The commonly used hardware for acquiring these images consists of two or more CCD cameras that are mounted on a rigid base, so that their relative positions are fixed. The rotation and translation between pairs of cameras is known through extrinsic calibration in this case. Even though the Virtual Stereo method differs in both the fact that only one camera is used, and that the extrinsic positions of images are known through visual odometry, still mostly the same preprocessing and matching pipeline as in the traditional scenario can be applied. In this chapter the basics of stereo matching will be developed in line with the traditional method, but all parts presented here will remain relevant for the subsequent chapter on the Virtual Stereo method.

The first section introduces the correspondence problem. Stereo matching aims to solve this problem as well as possible, but to make this feasible various constraints and assumptions have to be introduced. Apart from presenting these limitations, this section also hints already at some principles that underly common stereo matching algorithms. One of these is treated in the third section.

The second section builds upon the introduction to the geometry of pinhole cameras of chapter 2 and develops epipolar geometry, which describes the geometry of two pinhole cameras. Based on this, image rectification will be introduced. Rectification is a preprocessing step that rotates images virtually and thereby facilitates the actual matching step greatly.

The third section finally explains stereo matching per se. Its goal is to compute so called disparities between features that appear in both images being matched. Therefore, disparities are introduced first. After that we continue with an explanation of reprojection, which is the reconstruction of three-dimensional points from a two-dimensional array of disparity values. Furthermore, the section gives an overview of a state-of-the-art algorithm for stereo matching, presented by Hirschmüller in [1], and its implementation, which is part of the implementation of Virtual Stereo.

3.1 The Correspondence Problem

The fundamental working principle of depth estimation via stereo matching is to detect image regions that occur in both images of a pair, and that correspond to the same point in the world, from where light has been reflected onto the sensors of the cameras [1]. Based on the known intrinsic camera parameters and the relative translation and rotation between the two cameras, it is possible to triangulate the position of the world point: First take the (ideally sharply localized) corresponding image regions of the two

[1] In the case of Virtual Stereo only one physical camera exists, but it acts as a pair by recording images from different perspectives and at different points in time.

cameras and project lines from there back through the respective focal points towards the world point. Ideally these lines will intersect in the original point. By determining the line intersecting the two image planes at the matching image regions, a triangle is formed.

We could now proceed to determine the lengths of sides and magnitudes of angles of this triangle in order to derive the position of the point in the world. However, before continuing with the reconstruction of world points, we will first consider the *correspondence problem*, which needs to be solved in order to determine which image regions in the pair correspond to the same point in the world. In section 3.3.2 a stereo matching algorithm will be treated in more detail, but already now we will introduce some parts that are commonly part of such algorithms to elucidate the difficulties underlying the correspondence problem.

First of all we have to define a metric for similarity between two regions. We want to localize matching regions as sharply as possible, so the natural choice for the size of image regions to compare is that of single pixels. The most simple measure of similarity for this case is the difference in intensity, if we assume grey level images. Smaller differences should reflect a greater similarity of pixels. Measures of this kind are usually called *matching cost measures*. Without introducing multiple simplifying constraints and assumptions this measure will not suffice to find the matching of pixels between two images. Some of the most important constraints and assumptions are:

Uniqueness constraint The correspondence between pixels should be one-to-one, i.e. when searching for a match of a pixel we will have to decide on a single corresponding pixel in the other image. When just single pixels are taken into account it will generally be impossible to enforce this constraint, as too many potential matches will minimize the metric of similarity.

Matching windows One way to make the computation of matching cost less ambiguous, is to define a measure that also includes surrounding pixels in a window of fixed size which is centered on the pixels that are being considered for a match. These window-based algorithms follow a *local* approach to the problem. *Global* algorithms, which define a global cost function instead, exist as well, but will not be discussed in this thesis, as the implementation of stereo matching used for Virtual Stereo is window-based. An overview of various cost measures can be found in the taxonomy paper by Scharstein, Szeliski, and Zabih [9].

Smoothness constraint To further reduce the amount of possible matches, a smoothness constraint is typically part of the window-based cost measure. This means that the variation of distances between world points that are imaged in a window are assumed to be limited. Even though this can strongly increase the matching quality, it also causes errors where this constraint does not apply, i.e. at sharp depth discontinuities, e.g. at borders of objects in the world.

Occlusion Some regions that are visible in one image will be covered or generally out of view in the other. Recognizing such regions as impossible to be matched is important, as otherwise they will cause wrongly matched image regions.

Lambertian surfaces This assumption is largely independent of the actual correspondence problem and more related to the physical properties of objects in the world: It is assumed that objects reflect light isotropically so that the intensities of imaged points does not depend on the angle under which they are viewed. This assumption does not hold for glossy or transparent surfaces which are therefore often hardly matchable.

Epipolar constraint Without taking the extrinsic parameters into account, corresponding image regions could have any relative position to each other. For each pixel the whole of the other image would have to be searched to find a match. Luckily, this problem can be easily simplified with known extrinsic parameters. In this case the search for a match can be limited to a single line through the other image.

3.2 Epipolar Geometry and Rectification

3.2.1 Epipoles and Epipolar Lines

After having treated the limitations of finding stereo correspondences, let us now have a closer look again at the triangle formed by some arbitrary point $\mathbf{X}$ in the three-dimensional world and its projections onto the retinal planes of two cameras. In order to properly distinguish the two cameras, all geometric entities, like points, lines and matrices, that are specific to a camera will have as index either 1 or 2. Starting with the projection matrices P_1, P_2, which we know through the intrinsic and extrinsic paramters, we obtain the projections of $\mathbf{X}$ onto the retinal planes: $\mathbf{x}_1 = P_1\mathbf{X}$ and $\mathbf{x}_2 = P_2\mathbf{X}$. The triangle we are talking about is then $\triangle\mathbf{x_1x_2X}$.

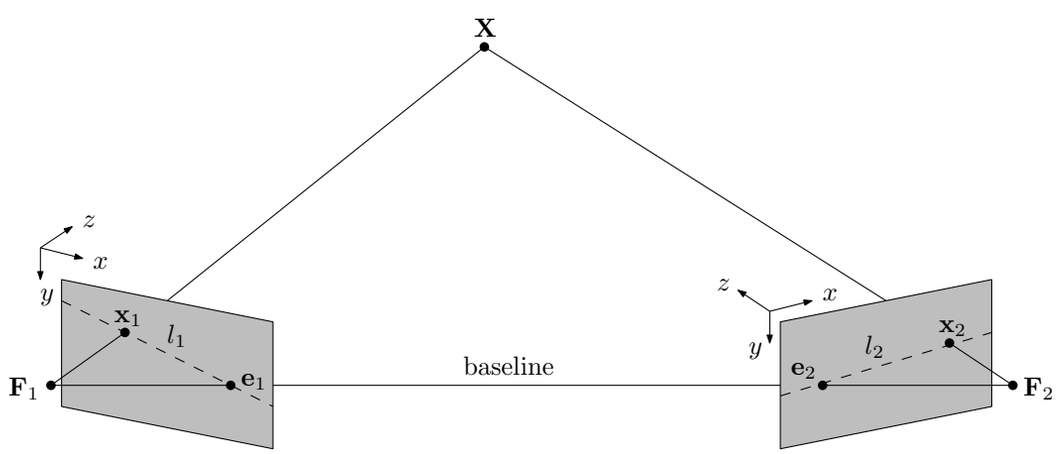


FIGURE 3.1: Epipolar geometry between two cameras; for better visual understanding the focal points are here behind the image planes.

Definition 3.1 (Epipolar Plane and Baseline)**.** The plane on which this triangle lies is called *epipolar plane* for a point $\mathbf{X}$. There are infinitely many epipolar planes, no matter which world point is coincident to them, they all contain the line $\overline{\mathbf{F}_1\mathbf{F}_2}$ between the two focal centers of the cameras. This line is called *baseline*.

Definition 3.2 (Epipoles)**.** The baseline can be seen also as a line of projection of $\mathbf{F}_1$ onto the image plane of camera 2, or vice-versa, of $\mathbf{F}_2$ onto the image plane of camera 1. The projected points $\mathbf{e}_1 := P_1\mathbf{F}_2$ and $\mathbf{e}_2 := P_2\mathbf{F}_1$ are called *epipoles*.

Definition 3.3 (Epipolar Lines)**.** The lines $l_1 := \overline{\mathbf{e}_1\mathbf{x}_1}$ and $l_2 := \overline{\mathbf{e}_2\mathbf{x}_2}$ are called *epipolar lines*. Clearly the epipolar lines are lines of intersection between the retinal planes and the epipolar plane.

Epipoles and epipolar lines obey the same rules as the points and lines in homogeneous coordinates that we have already seen in section 2.1.1. This means that we can calculate the epipolar lines as $l_i = \mathbf{e}_i \times \mathbf{x}_i$. Furthermore there can be also epipoles at infinity on a retinal plane. In the following sections these basics of homogeneous coordinates will become relevant again.

3.2.2 Derivation of the Fundamental Matrix F

Now that we know that we can limit the search for a corresponding point to an epipolar line we should have a closer look at how to compute this line. Specifically for a point $\mathbf{x}_1$, we want to compute the line l_2 on which the corresponding point $\mathbf{x}_2$ lies. This mapping is given by the fundamental matrix. The following derivation of this matrix is based mainly on chapter 9.2.2 of Hardley and Zisserman's book on geometry in computer vision [10].

As the first step let us compute a possible candidate as correspondence to $\mathbf{x}_1$: Let P_1^+ be the pseudo-inverse of P_1, i.e. $P_1 P_1^+ = I$. Now $\mathbf{x}_2' = P_2 P_1^+ \mathbf{x}_1$ is a mapping, firstly from $\mathbf{x}_1$ to a point in three-dimensional space, and secondly from this world point to the point $\mathbf{x}_2'$ on the image plane 2. From a naive perspective it looks as if we have already found the correspondence. This is not the case, because the point in the world through which this transfer happened is not $\mathbf{X}$. Clearly, any point on the line running from $\mathbf{F}_1$ through $\mathbf{X}$ to infinity will be mapped to $\mathbf{x}_1$ by P_1, so P_1^+ could map $\mathbf{x}_1$ back to any point along this line (which one is irrelevant for this derivation). Therefore it generally holds that $\mathbf{x}_2 \neq \mathbf{x}_2'$. This world point lies on the epipolar plane however, and therefore $\mathbf{x}_2'$ lies on the epipolar line l_2. This means that, given the epipole $\mathbf{e}_2$, we can compute $l_2 = \mathbf{e}_2 \times \mathbf{x}_2' = \mathbf{e}_2 \times (P_2 P_1^+ \mathbf{x}_1)$. The fundamental matrix will be the linear map $\mathbf{x}_1 \mapsto l_2$. In order to eliminate the cross product from the previous equation, we can use a skew symmetric matrix that expresses a cross product as a matrix multiplication:

$$a \times b = [a]_\times b \tag{3.1}$$

$$\text{where } [a]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{3.2}$$

The fundamental matrix is therefore $F = [\mathbf{e}_2]_\times P_2 P_1^+$. We have now all the parts necessary for deriving the fundamental matrix for a specific case of projection matrices. Let the projection matrices be the following:

$$P_1 = K_1 \left[I \mid \mathbf{0} \right], \quad P_2 = K_2 \left[R \mid t \right] \tag{3.3}$$

This means that the first camera is in a reference coordinate system. This assumption will lead to no loss of generality, because in the end only the relative rotation between cameras will be relevant for the resulting fundamental matrix. From this it follows that

$$P_1^+ = \left[\frac{K_1^{-1}}{\mathbf{0}^T} \right], \quad \mathbf{F}_1 = \left[\frac{\mathbf{0}}{1} \right] \tag{3.4}$$

Now the fundamental matrix is given as

$$F = [\mathbf{e}_2]_\times P_2 P_1^+ \tag{3.5}$$

$$= [P_2 \mathbf{F}_1]_\times P_2 P_1^+ \tag{3.6}$$

$$= [P_2 \mathbf{F}_1]_\times K_2 \left[R \mid t \right] \left[\frac{K_1^{-1}}{\mathbf{0}^T} \right] \tag{3.7}$$

$$= [P_2 \mathbf{F}_1]_\times K_2 R K_1^{-1} \tag{3.8}$$

3.2.3 Computing Standard Rectification Transforms

In the previous section we have seen how we can limit the search space for point correspondence to an epipolar line, which can be calculated by the fundamental matrix.

Rectification simplifies this search further by computing new intrinsic and extrinsic matrices for the two cameras. These new matrices correspond to a situation where the retinal planes are coplanar and parallel to the baseline. After computing these rectification matrices, the images can be remapped by a rectification transform that has the same effect as if the cameras were rotated according to the rectification matrices. The following derivation of rectification is based mainly on the paper by Fusiello et al. [3]. In the implementations for this thesis the OpenCV function `stereoRectify()` was used. The working principle of this function is the same as the one presented here. The choice of OpenCV as the main computer vision library for our implementations is explained in section 3.3.2.

In the standard rectification procedure described here, the baseline will run parallel to the u-axes of the rectified images. We assume that in this standard case the unrectified images were taken from positions that are already close to the rectified ones, i.e. the directions of the x-axes do not differ too much between cameras. This is the case for a standard horizontal stereo rig or a single camera that is moving sidewards. Rectification for the case where the camera is moving mainly forwards, meaning along its z-axis, will be treated in section 4.3.2.

We will see that after the rectification procedure described here, the epipolar lines will all be parallel to each other and horizontal in both images. This makes the search for correspondences computationally efficient, as epipolar lines will coincide with the rows of the image matrices. Memory cells will be accesssed linearly during search, and therefore the number of cache misses is strongly diminished.

For the derivation of the rectification transform we take again the same projection matrices P_1, P_2 as in the previous section. There we considered the extrinsic part of P_2 as a passive transformation from coordinate frame 1 to 2. This time we will take the point of view of active transformations. According to eq. (2.19) we obtain:

$$P_2 = K_2 \left[R \mid t \right] = K_2 \left[R_C^T \mid -R_C^T t_C \right] \tag{3.9}$$

Naturally, active and passive representations are the same for P_1. We now want to find a new active rotation matrix $\tilde{R}_C$ that puts both image planes into the same orientation in the world. Additionally the intrinsic camera matrices for both cameras need to be identical; otherwise the image planes would be parallel, but not coplanar, because the focal lengths could differ. A new camera matrix can be chosen arbitrarily, but ideally it is selected so that no areas of the images get lost due to a smaller focal length. We do not go into the details of computing a new intrinsic matrix here, but it should be mentioned that, apart from minimizing lost image area by selecting a suitable focal length, it is also required that $f_u = f_v = f$ in the new intrinsic matrix $\tilde{K}$. The two focal lengths need to be equal for reprojection to function as it is described in section 3.3.1. The focal points of both cameras need to stay in the same position so that the rectification transform can be applied. Following these constraints the new rectified projection matrices will be:

$$\tilde{P}_1 = \tilde{K} \left[\tilde{R}_C^T \mid \mathbf{0} \right], \quad \tilde{P}_2 = \tilde{K} \left[\tilde{R}_C^T \mid -\tilde{R}^T t_C \right] \tag{3.10}$$

The new rotation matrix $\tilde{R}_C$ can, like every rotation matrix, be seen as a orthonormal basis of $\mathbb{R}^3$. This follows directly from the fact that rotation matrices are orthogonal. Therefore we can specify it by its columns, which are the base vectors. Let $\tilde{R}_C = [r_1\ r_2\ r_3]$; then the base vectors should be chosen as follows:

- For the new vector corresponding to the x-axis we take the normed vector along the baseline from camera 1 to camera 2. This constraint guarantees that the image planes are parallel to the baseline. We pick $r_1 = -\frac{t_C}{||t_C||}$.

- The new y-axis needs to be orthogonal to the new x-axis. Additionally, it is chosen to be orthogonal to the z-vector of the rotation matrix of camera 1, this rotation matrix being the identity matrix. This latter constraint is chosen somewhat arbitrarily; in the case where the two cameras would be translated purely along the z-axis this would not be possible, as then the x- and z-axes would coincide. For standard rectification the z-axes of both cameras should be close to orthogonal to the baseline however, so that this constraint is sensible. Therefore we pick $r_2 = r_1 \times (0\ 0\ 1)^T$.
- The new z-axis is necessarily orthogonal to the new x- and y-axes. Therefore it has to be $r_3 = r_1 \times r_2$.

Let us now consider where the epipole $\mathbf{e}_2$ is after rectification. It gets projected along the baseline, which is the new x-axis, but the ray of projection does not intersect with the image planes, as they are parallel to the baseline. Therefore $\mathbf{e}_2 = (-1\ 0\ 0)^T$; the epipole is a point at infinity in negative x-direction. Epipole $\mathbf{e}_1 = (1\ 0\ 0)^T$ is also at infinity, but in positive x-direction. Normally there is no distinction between these two points in homogeneous coordinates, because they are equivalent up to multiplication by a scalar (-1 in this case), but it should be noted that later on in section 4.3.2 such a change of sign will not be considered an equivalence. This means that projected geometry will have an orientation in that case.

We will not concern us further with oriented projected geometry now. Let us instead look which form the fundamental matrix takes after rectification. Both cameras have the same intrinsic matrix $\tilde{K}$ then and there is no relative rotation between them. Therefore $F = [\mathbf{e}_2]_\times \tilde{K} I \tilde{K}^{-1} = [\mathbf{e}_2]_\times$. We can now derive the equation for an epipolar line corresponding to the point $\mathbf{x}_1 = (a\ b\ 1)^T$:

$$F\mathbf{x_1} = [\mathbf{e}_2]_\times \mathbf{x_1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} a \\ b \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ b \end{pmatrix} \tag{3.11}$$

The line equation is $f(u, v) = b - v = 0$. Therefore the epipolar line is horizontal in the image and at the same v-coordinate as $\mathbf{x}_1$, which is what we set out to achieve by rectification.

3.2.4 Rotating Images via a Homography

In the beginning of section 3.2.2 we have mentioned that it is generally impossible to map corresponding points onto each other based on known projection matrices alone. Luckily there is an exception to this, namely when the focal point of a camera is at the same position for both projection matrices.

Assume we want to find out how an image point $\mathbf{x}$ will be mapped to an image point $\mathbf{x}'$ after rotating and possibly changing the intrinsic parameters of a camera. Both points should be the images of the same point $\mathbf{X}$ in three-dimensional space. The following derivation of this remapping is taken from chapter 8.4.2 of [10]:

$$\mathbf{x} = K \left[I \mid \mathbf{0} \right] \mathbf{X} \tag{3.12}$$

$$\mathbf{x}' = K' \left[R \mid \mathbf{0} \right] \mathbf{X} = K' R K^{-1} K \left[I \mid \mathbf{0} \right] \mathbf{X} = K' R K^{-1} \mathbf{x} \tag{3.13}$$

The matrix $H = K' R K^{-1}$ that maps $\mathbf{x}$ onto $\mathbf{x}'$ is an instance of a so called homography. We will informally refer to a remapping by such a homography as "rotation" of images, even though this is not actually correct; not the two-dimensional images are rotated, but the image planes in three-dimensional space, and the remapping has the according effect. The first usage of image rotation, in the context of this thesis, is to apply the rectification transforms to grey level images. Homographies can not only be used for

rotation of images containing intensity values however. In section 4.2.1 they will also be part of the process for rotating images containing disparity values.

3.3 Computation of Disparities and 3D Reconstruction

3.3.1 Disparities and Reprojection for Rectified Cameras

Now that we know how to rectify images, and that in rectified images corresponding points lie on the same v-coordinate, we can define an especially simple output data type for a stereo matching algorithm. Assume we have a pair of rectified projection matrices $\tilde{P}_1, \tilde{P}_2$, and that the first one is in a reference coordinate system. This assumption again leads to no loss of generality. It just means that in the following derivation all steps happen modulo a passive transformation that puts the first camera in a reference frame. The inverse of this passive transformation can be applied later to the reprojected points in the world. Assume further on that we have already found two corresponding points $\mathbf{x}_1, \mathbf{x}_2$ in the respective images. As the images are rectified, the points will only differ in their u-coordinates.

Definition 3.4 (Disparity)**.** Given any two corresponding image points $\mathbf{x}_1 = (u_1\ v)^T$ and $\mathbf{x}_2 = (u_2\ v)^T$ (in inhomogeneous coordinates) the disparity function at the first point is defined as $d(u_1, v) = u_1 - u_2$.

It would also be possible to define disparity from the perspective of camera 2, but the reconstruction of world points from disparities is easier from a camera in a reference frame. Given the possibility of computing matches, which will be treated in the next section, we have now the possibility of generating a *disparity image*. This image is well-defined for every pixel of image 1, given that we are able to find correspondences for all pixels in it. This will not be the case in practice, and therefore a special value will mark these pixels as invalid, so that they will not be reprojected. The percentage of invalid pixels in a disparity image is a measure of quality that will be treated further in section 5.2.2.

Now that we have defined disparity, we should have a look at how to reconstruct world points from disparities. This process is termed reprojection. Parts of the following derivation are based on chapter 9.3.1 of [10].

First we have to consider how the two rectified projection matrices project world points. The matrices are

$$\tilde{P}_1 = \tilde{K}\left[I \mid \mathbf{0}\right] \tag{3.14}$$

$$\tilde{P}_2 = \tilde{K}\left[I \mid t\right] \tag{3.15}$$

and the projection of a world point $\mathbf{X} = (x\ y\ z\ 1)^T$ is

$$\mathbf{x}_1 \sim \tilde{P}_1\mathbf{X} = \tilde{K}\begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3.16}$$

$$\mathbf{x}_2 \sim \tilde{P}_2\mathbf{X} = \tilde{K}\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix} + t\right) \tag{3.17}$$

The difference between these two image point is precisely the disparity, for which we will now obtain a form that relates it to depth. Here it is important again to make the distinction between equivalence of homogeneous coordinates and true equality. The coordinates of $\mathbf{x}_1$ and $\mathbf{x}_2$ should be normalized, so that we can calculate a term for disparity

from their difference in the u-coordinate. For this we have to divide the right sides by z, because then $\mathbf{x}_1$ is normalized already:

$$\mathbf{x}_1 \sim \begin{pmatrix} fx/z - u_0 \\ fy/z - v_0 \\ 1 \end{pmatrix} = \underbrace{\begin{bmatrix} f & 0 & -u_0 \\ 0 & f & -v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{=\tilde{K}} \begin{pmatrix} x/z \\ y/z \\ 1 \end{pmatrix} \tag{3.18}$$

When we define the width of the baseline to be b, the translation vector is equal to $t = (b\ 0\ 0)^T$. This means that $\mathbf{x}_2$ is normal too (t is 0 in the z-coordinate, so it does not affect this coordinate in $\mathbf{x}_2$), and we get the following equation:

$$\mathbf{x}_1 = \mathbf{x}_2 + \tilde{K}t/z = \mathbf{x}_2 + 1/z \begin{bmatrix} f & 0 & -u_0 \\ 0 & f & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} b \\ 0 \\ 0 \end{pmatrix} = \mathbf{x}_2 + \begin{pmatrix} bf/z \\ 0 \\ 0 \end{pmatrix} \tag{3.19}$$

We have solved the problem of calculating the depth z of a world point from the disparity d of its pixel in the reference image:

$$d = \frac{bf}{z} \tag{3.20}$$

$$\Leftrightarrow \qquad z = \frac{bf}{d} \tag{3.21}$$

The x- and y-coordinates of $\mathbf{X}$ can now also be calculated from eq. (3.18) by substituting z for the right hand side of eq. (3.21):

$$x = b/d(u_1 - u_0) \tag{3.22}$$

$$y = b/d(v - v_0) \tag{3.23}$$

Note that we can only make this same substitution for both x and y, because we only have one focal length f. In the case where $f_u \neq f_v$ we would have ended up with a more complex formula that could not have been expressed as a matrix multiplication as in the following:

Definition 3.5 (Reprojection Matrix)**.** The *reprojection matrix* Q maps the vector $(u_1\ v\ d\ 1)^T$, which represents one pixel of the disparity image, to the world point $\mathbf{X}$:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & -u_0 \\ 0 & 1 & 0 & -v_0 \\ 0 & 0 & 0 & f \\ 0 & 0 & 1/b & 0 \end{bmatrix}}_{=:Q} \begin{pmatrix} u_1 \\ v \\ d \\ 1 \end{pmatrix} = \begin{pmatrix} u_1 - u_0 \\ v - v_0 \\ f \\ d/b \end{pmatrix} \sim \begin{pmatrix} b/d(u_1 - u_0) \\ b/d(v - v_0) \\ bf/d \\ 1 \end{pmatrix} = \mathbf{X} \tag{3.24}$$

At this point a note on the scale of reprojected points should be made. Disparities are usually discrete values representing distances as the multiple of the width of a pixel on the CCD sensor. Therefore all other units of distance, like focal length and baseline, have to be represented in the unit of pixels and naturally the reprojected points in the world will then also be in this unit. It is straightforward to multiply $1/b$ in Q with the factor px/m, the number of pixels per meter on the sensor. Consequently, the reprojected points will then be calculated in meters.

3.3.2 Semi-Global Block Matching

For the implementation of the Virtual Stereo method a matching algorithm had to be chosen. The functioning of the Virtual Stereo method is not dependent on any specific algorithm, therefore the choice was made based on technical considerations, namely whether the implementation of the algorithm is part of a library that is well documented and free to use, so that results can be reproduced. An additional requirement for this library was that it offers functionality for the geometric transformations of images which are necessary for rectification. OpenCV meets these requirements best, and has therefore been selected for the implementation of the Virtual Stereo method. Its most performant stereo matching function is *Semi-Global Block Matching (SGBM)*, which implements an algorithm described in [1] with some modifications.

The stereo matching algorithm is not the focus of this thesis, therefore only a rough overview of the algorithm, its OpenCV implementation, and where they differ will be given. Scharstein et al. present a taxonomy of stereo algorithms in [9]. This taxonomy categorizes these algorithms according to four steps that most of them perform. In the following paragraph Hirschmüller's algorithm and its implementation will be explained according to these steps:

Matching cost calculation Hirschmüller proposes to use mutual information, which is based on (joint) entropy of pixels, as a matching cost measure. The OpenCV implementation uses a simpler measure due to Birchfield and Tomasi [11]. Another difference in OpenCV is that the cost is calculated over square windows (called blocks) around a pixel instead of calculating the cost pixel-wise.

Cost aggregation In this step the matching cost for each pixel in the reference image is calculated over the range of permissible disparities. This means that at each pixel (or block) in the reference image, every disparity in the range relating the reference pixel to one in the second image, gets a matching cost assigned. All these costs are therefore aggregated along the epipolar lines. SGBM incorporates also a smoothness constraint into this step. In order to ensure that the variance of disparities around a possible match is kept small, matching costs are additionally aggregated along 16 lines surrounding each possible match. The fact that these lines run throughout the whole image is the reason for calling the method semi-global. Despite this additional computational effort, Hirschmüller's method is still performant due to dynamic programming. Nevertheless, the memory consumption and runtime are raised drastically by this. Hirschmüller calculates the complexity in memory and runtime of the entire cost aggregation for an image of width W, height H, and a range of disparities of size D to be in the complexity class $O(WHD)$. This computation is already very costly, but it does not yet factor in the search along the 16 directions. The number of directions is an additional factor in the runtime, therefore the OpenCV stereo matching implementation (the class `StereoSGBM`) used for Virtual Stereo aggregates matching costs along fewer lines. For the implementation of VS we used the parameter `MODE_SGBM` which limits the search directions to 5.

Disparity computation In this step the disparity with lowest cost among the aggregated ones is selected. In [1], disparity images are computed symmetrically for both images, meaning that there are two passes, alternating the role of reference image between the two. This enables a left-right-consistency check, which would help to enforce the uniqueness constraint, and lead to a better detection of occlusions. Due to the strong effect on runtime this has not been implemented for Virtual Stereo. A simpler way to attain the uniqueness constraint approximately is the uniqueness check that is part of the OpenCV function. There the best disparity

has to win against the second best by a relative margin of cost to be considered valid. This check will also be relevant as measure of confidence in a disparity later on in section 5.2.1.

Disparity refinement In [1], Hirschmüller proposes various techniques for fixing errors in the resulting disparity image. For some of these, OpenCV implements simpler versions, specifically speckle-filtering, which invalidates small outlier patches of disparities, and sub-pixel quadratic interpolation as smoothing between disparities. In Hirschmüller's paper more sophisticated interpolation, extrapolation, and plane fitting techniques are developed. These rely however also on occlusion images, and are therefore be infeasible for the OpenCV implementation, due to the aforementioned effect on runtime.

[1] presents some further extensions to the principle of semi-global matching that are not part of the OpenCV implementation SGBM. A notable one is the fusion of disparity images, which have been computed from pairs of images with varying baselines. This extension has been implemented and analyzed for Virtual Stereo. It is being treated in section 4.2.2.

Chapter 4

The Virtual Stereo Method

This chapter presents the main contribution of the thesis, the Virtual Stereo method. As the name suggests, it is a method that builds on the principle of the stereo matching methods presented in chapter 3. As opposed to traditional stereo methods, Virtual Stereo uses a monocular camera that is being moved through space. The frames used for matching are acquired consecutively in time. Unlike in standard stereo vision no physical binocular camera exists, instead the binocular camera is constructed *virtually* between frames recorded with a monocular camera, hence the name Virtual Stereo. This chapter presents the theoretical basis of the implementation of Virtual Stereo for two different scenarios. In chapter 5 the results of Virtual Stereo are compared to those of a conventional stereo matching method.

The first section of this chapter explains the working principle of the Virtual Stereo method: The extrinsic camera transformations are obtained directly from the recorded image sequence which is also the input data for stereo matching. The computation of extrinsics can, in principle, proceed live, directly after recording the images, instead of using a static calibration that has been computed in advance. (For the evaluation of prototypical implementation we processed a prerecorded video; the images were not processed in real-time there.) The method to compute extrinsics in this manner is called visual odometry; besides stereo matching it is the second building block of Virtual Stereo. Due to its importance, this section also gives an overview of the principles of visual odometry and the implementation that was used for Virtual Stereo.

The second section discusses the specifics of the scenario where the camera faces sidewards, away from the vehicle on which it is mounted. In this case it is possible to fuse disparity images that have been computed based on intensity images taken at varying baseline widths. The motivation for this multi-baseline fusion and the necessary (preprocessing) steps for implementing it are being treated.

The third section explores the second scenario; here the camera faces forwards. In this case the standard rectification procedure, used so far, is no longer applicable. A completely new rectification procedure for this scenario is derived. This procedure rectifies not whole images, but instead single regions individually. Even though this new rectification procedure produces results that look very different to the ones of the standard procedure, the procedures do not differ much in principle. In both cases, images, or respectively image regions, are rectified by homographies that are computed from camera intrinsics and a camera rotation. Consequently, stereo matching and reprojection work in the same way for front-facing Virtual Stereo as with the traditional method.

4.1 Basic Working Principle of Virtual Stereo

4.1.1 Extrinsic Camera Transformations from Visual Odometry

Chapter 3 introduced stereo matching as a method to reconstruct the three-dimensional shape of a scene from pairs of images using intrinsic and extrinsic calibration. For fixed

stereo rigs, both intrinsic and extrinsic calibration are known from a calibration procedure and assumed to be constant. In the case of the *Virtual Stereo (VS)* method only the intrinsic calibration is assumed to be constant and known from a prior calibration. The extrinsic transformation between two camera frames is obtained from the trajectory of a single camera moving through the world. The method used for computing this trajectory is *Visual Odometry (VO)*, which is described in more detail in section 4.1.2.

Before giving an overview of VO, the main difference between standard (fixed rig) stereo methods and the Virtual Stereo method should be explained: in standard methods the two frames with fixed relative poses correspond to two separate physical cameras which record pairs of images synchronously. In Virtual Stereo there is only one physical camera, and pairs of frames for stereo matching correspond to images that have been recorded subsequently in time. While the standard and virtual method have much in common, there are multiple deviations between these principles, which will be treated over the course of this chapter.

For any stereo matching method, a wider baseline is often desirable due to better depth reconstruction, but when the baseline becomes too wide the field of view of the two frames begin to overlap only at a greater depth, so that closer objects can not be matched. One main difference between traditional methods and VS is that with VS the baseline width is not constant; it depends both on the framerate of the camera and the speed of the vehicle on which it is mounted. This has not been a problem in our VS experiments. At a speed of $30km/h$ ($\approx 8.3m/s$) and a framerate of $30Hz$ the baseline for two directly consecutive frames in VS is approximately $\frac{8.3}{30}m \approx 27.6cm$, which is in the range of typical baseline widths for fixed rig stereo cameras. VS has as lower limit for the baseline width the distance between two directly consecutive frames. As any two images of a sequence can be matched, there is in principle no upper limit for the baseline width, but it should be considered that the accuracy of camera transformations and rotations computed with VO can degrade for greater distances. In our evaluation we limited the baseline width of VS usually to lenghts below $1m$, as the objects we recorded were relatively close to the camera (in the range of a few meters). With these relatively small variations of baseline widths, and without ground truth data for the trajectory, we were unable to make out the exact effect of VO accuracy on the accuracy of scene reconstruction with VS.

A key advantage of VS over standard methods is that the trajectory computed by VO serves a double purpose: not only does it deliver extrinsic calibration, but it also makes reprojections along this trajectory through the three-dimensional world possible. Any frame of a sequence can act as a reference frame for stereo matching, and, as the transformation between any two frames is known, the reprojected points of different reference frames can be aggregated in one global coordinate system. With a standard method it is also required to compute a camera trajectory for global reconstruction, but additionally a second camera and the extrinsic calibration between the two are needed. Especially the extrinsic calibration is a weakness of fixed stereo rigs. Our experience with those rigs in automotive use cases shows that the cameras easily shift their positions and orientations due to acceleration and especially vibrations. In effect, an extrinsic calibration that has been computed for a stationary vehicle can become invalid after just a few kilometers into a trip. Virtual Stereo does not require an extrinsic calibration, hence it avoids this problem. Having no extrinsic calibration has however also the downside that reconstruction with correct scaling in relation to the world (in meters) is no longer possible. This problem is treated in more detail in the next section.

4.1.2 Overview of Direct and Sparse Visual Odometry

The Virtual Stereo method is fundamentally a *Structure from Motion (SfM)* method. It computes the dense, three-dimensional structure of a scenery based on motion through it. We have treated the theory of structure reconstruction via stereo matching already in chapter 3. Before coming to the specifics of Virtual Stereo, the second building block, namely the computation of motion, specifically camera trajectories, should be given an overview. As with stereo matching, VO is not the focus of this thesis, and it will be treated only in the degree of detail necessary to understand its working principle and why we chose *Direct Sparse Odometry (DSO)*, developed by Engel, Koltun, and Cremers in [12], as the specific algorithm and implementation.

As already hinted, the goal of VO is to estimate the transformations of camera poses over a sequence of images. A change of pose relating two consecutive camera frames is modelled to be dependent on the change of light intensities between the images. This change of intensities is usually called *photometric error*. The main working principle of VO is to solve a optimization problem that chooses pose transformations which minimize the photometric error between images. [1] This optimization problem is, in a broader sense, related to the correspondence problem: for determining photometric error, matches between image regions are sought as well, but here the disparities together with the extrinsic transformation, and possibly other parameters, are the unknown values of a probabilistic model that aims to maximize the likelihood that the estimated parameters explain the measured photometric error best. The precise formulation of this model is one of the key challenges in designing VO algorithms. A commonly chosen model, which has also been used in the case of DSO, is *Maximum Likelihood Estimation (MLE)*. The choice for a model does not determine the solution completely yet, there are still many parts to defined, such as the precise error measure and the set of parameters.

One key factor for the choice of DSO in the implementation of Virtual Stereo was the fact that DSO uses a *sparse* set of small windows distributed over the images for the computation of disparities. *Dense* algorithms, on the other hand, compute disparities over the entirety of images. This means that these latter algorithms reconstruct sceneries at a density that is close to those of stereo matching algorithms operating on calibrated image pairs, so that they constitute already fully fledged SfM algorithms. There is, however, good reason for keeping dense reconstruction separate from the computation of odometry: for dense algorithms it is usually necessary to add a smoothness constraint (see section 3.1) to the function which is to be optimized. In effect, correlations between disparities are added to the corresponding linear system, which therefore becomes much more complex. In order to still solve it, additional approximations have to be used in the optimization. The authors of [12] suspect that this leads to a loss of large-scale accuracy in the trajectories computed by current dense algorithms. Especially in the automotive use-case of VO/SfM it is crucial to maintain accuracy over long distances, so that the complete scenery along a recorded video can be reconstructed consistently. In addition to the high accuracy, DSO is capable of performing in real time on widely available hardware. Virtual Stereo combines the precise, but sparsely reconstructed, odometry of DSO with an independent dense reconstruction via stereo matching.

A few words need to be said on the limitations of DSO. First of all it should be noted that we do not know the scale of the scenery it reconstructs. Even though the magnitudes of the translations computed via DSO stay consistent over long distances, their

[1] DSO is an instance of a *direct* VO algorithm that minimizes photometric error without computing an intermediate measure based on it. *Indirect* algorithms optimize measures that are derived from the photometric error instead.

unit is unknown. DSO itself does not provide a method to scale the coordinates of its result so that their unit is in meters, and therefore the dense reconstruction of VS has also an unknown scaling. This problem is typical for monocular VO/SfM algorithms. For the purpose of evaluation, we recorded one video with a front-facing stereo rig and processed it with a proprietary modification of DSO, called *stereo-DSO*. This version of DSO uses disparity images from the fixed rig to scale its own disparities for the estimation of the photometric error. Stereo-DSO was developed, mainly, because this consistent scaling of disparities leads to a greater robustness of results. For the use-case of VS it is additionally useful, because, with a known ratio of pixels to meters, both the disparities of the fixed rig (see section 3.3.1) and the trajectory can be scaled metrically. As a result we were able to obtain reprojected points with the coordinates in meters, both from the fixed stereo rig and from VS. This enabled us to quantitatively and qualitatively compare the results of both methods. The comparison of these two methods is being treated in more detail in chapter 5. Naturally this scenario is only sensible for evaluation; using a fixed stereo rig defeats the purpose of VS, because it is meant to work with just a monocular camera. For a real-world application of VS the scaling would need to be irrelevant or computed by some other means.

4.2 Virtual Stereo for a Side-Facing Camera

The first application of VS was based on the recordings of a vehicle-mounted camera that was facing sidewards in relation to the direction of the vehicle when driving straight ahead. In other words, the baseline of consecutive frames was close to parallel to the x-axis of the camera and in the direction of the vehicle when it was driving in a straight line. The z-axis was horizontal, so that objects at the side of the road, such as parking cars, building facades, trees, etc., were in view. See fig. 4.1 for the orientation of the frames.

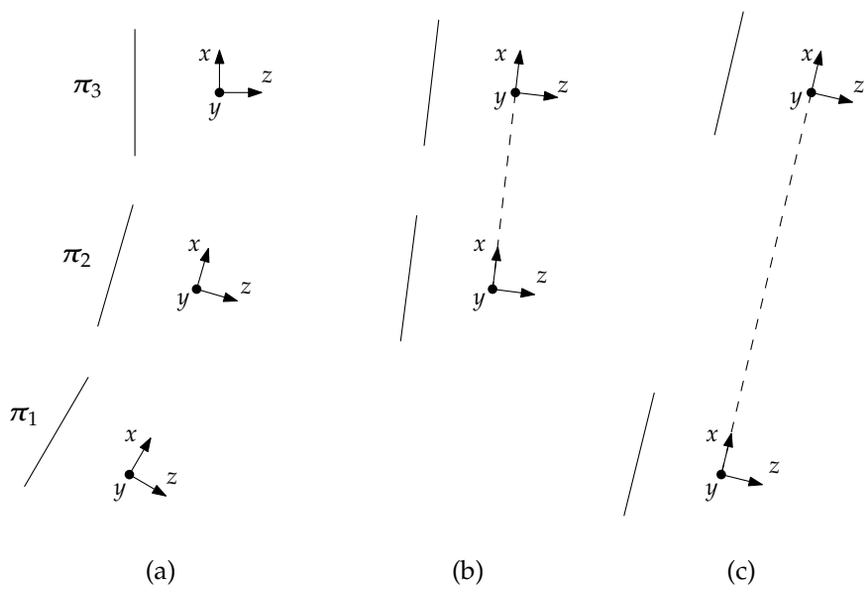


FIGURE 4.1: Orientation of frames in side-facing VS; the graphs show image planes and coordinate axes originating from the respective focal points; the perspective is top-down. Part (a) shows the original three consecutive frames (1 recorded first, then 2, then 3). Part (b) and (c) show differently rectified image planes (the dashed lines are the baselines): frame 3 rectified with frame 2 in (b), and frame 3 rectified with frame 1 in (c).

When driving in a curve, the z-axes of consecutive frames become either "cross-eyed" or they will diverge, but this effect is never very strong for standard four-wheeled road vehicles, as their turning radius is in the range of multiple meters. If we assume a typical turning radius of $5m$ and a baseline width of $30cm$, which can easily be attained between two directly consecutive frames when recording at $30Hz$, the angle between the two z-axes can be calculated as the angle at the apex of an isosceles triangle: $2\arcsin(\frac{0.3}{2*5}) \approx 3.4°$; clearly a very small divergence between the axes. Therefore the relative orientation between frames is, even in tight curves, very close to that of a fixed stereo rig. As a result, we were able to to rectify frames with the standard method described in section 3.2.3, using extrinsic transformations computed via DSO.

Apart from the advantages of VS over standard stereo algorithms for SfM, which have already been noted in section 4.1.2, there is another benefit to using VS in the side-facing scenario. With VS, extrinsics data between any pair of frames in a sequence is available. This means that we can pick any image as reference image and then rectify and match it with multiple other images instead of just one. The motivation for this is that multiple disparity images can be computed for this reference frame, which can then be fused together in order to obtain a denser disparity image, and thereby a denser three-dimensional scene reconstruction. An alternative to fusion of disparity images would be the fusion of reprojected points. We estimated this latter approach to be much more complex however, as point clouds can not be overlayed in the same way as disparity images. When overlaying disparity images, the disparity values at the same pixel location can be fused by averaging the depth along their shared ray of projection. Point clouds, on the other hand, are unstructured in this respect; their points are not related by rays of projection anymore. Before treating the fusion itself in section 4.2.2, disparity images first have to be aligned by rotation. This rotation is the topic of the following subsection.

4.2.1 Rotating Disparity Images via a Homography

When rectifying a reference image against multiple others, the rectifying transforms will generally be different for any of the pairs. Rectifying transforms are pure rotations; the focal points of the rectified images stay at the same position. Therefore, we do not have to concern us with correcting the effect of translations, however the rectified reference image will have a different extrinsic rotation for each other image it is being rectified with. The disparity images have the same orientation in space as the rectified reference image on which they are based. Consequently, for fusion being able to function pixel-wise on disparities, the disparity images have to be rotated [2] so that corresponding points in them are at the same location. We have already treated rotation of intensity images in section 3.2.4. The rotation of disparity images functions in a similar manner, however, when rotating a disparity image simply by a homography, as we did it for intensity images, disparity values will be mapped to different places on the image plane where the same value corresponds to a different ray of projection. Therefore we will adapt the disparity values to the new perspective by first reprojecting them as points, rotating these around the camera, and then calculate the new disparity values from their depth after rotation. The change of disparity values and the change of coordinates via homography can be calculated independently from each other. In our implementation first the values and then their indices will be changed.

Assuming a rectified intrinsic matrix $\tilde{K}$ and a rectified extrinsic rotation $\tilde{R}_C^T$ for the reference camera (as in eq. (3.10)), we describe in the following a procedure that rotates any disparity image back to the original orientation of the camera before rectification. The approach of simply reverting the rectification rotations of the disparity images was

[2] See section 3.2.4 for the explanation of the oversimplified term "image rotation".

chosen for the sake of simplicity. It is also possible to rotate them all to any other common extrinsic rotation. The camera intrinsics and image matrix size could, as with rectification, be changed in this process, but in this derivation we leave them the same. Retaining the old parameters here did not pose a problem for side-facing VS, as the rotations were usually quite small, and the amount of disparity pixels that got lost due to being remapped outside the image matrices were neglegible. When this method is used on disparity images from front-facing VS however, two different camera intrinsics will be used for reprojection and subsequent projection. The following four steps achieve the described rotation of disparity images:

1. Assuming a disparity image as a matrix with dimensions $m \times n$, every disparity is first reprojected by its reprojection matrix Q. We describe the whole process point-wise for arbitrary matrix indices $i \in [1..m], j \in [1..n]$. Any arbitrary value d_{ij} of the disparity image $D = [d_{ij}]_{i\in[1..m],j\in[1..n]}$ gets mapped as follows:

$$d_{ij} \mapsto (x\ y\ z)^T_{ij} \tag{4.1}$$

$$\text{where } Q \begin{pmatrix} d \\ i \\ j \\ 1 \end{pmatrix} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ w \end{pmatrix} \sim \begin{pmatrix} \tilde{x}/w \\ \tilde{y}/w \\ \tilde{z}/w \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{4.2}$$

   The resulting matrix $W = [(x\ y\ z)^T_{ij}]_{i\in[1..m],j\in[1..n]}$ of reprojected world points has dimensions $m \times n \times 3$ and preserves the neighborhood of the original disparities.

2. As the next step, the world points are rotated by an active rotation that corresponds to reverting the rectification rotation of the camera. The resulting array has the same dimensions as W and also keeps the points at the same indices as in the disparity image.

$$(x\ y\ z)^T_{ij} \mapsto (x'\ y'\ z')^T_{ij} \tag{4.3}$$

$$\text{where } \tilde{R}_C \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \tag{4.4}$$

   This step can also be combined with the previous one by reprojecting and simultaneously rotating the points via the matrix $\left[\begin{array}{c|c} \tilde{R}_C & \mathbf{0} \\ \hline \mathbf{0}^T & 1 \end{array}\right] Q$. The matrix we obtain after this step is $W' = [(x'\ y'\ z')^T_{ij}]_{i\in[1..m],j\in[1..n]}$.

3. From the now rotated points of W', we compute the new disparities via eq. (3.20):

$$(x'\ y'\ z')^T_{ij} \mapsto d'_{ij} \tag{4.5}$$

$$\text{where } d'_{ij} = \frac{bf}{z'_{ij}} \tag{4.6}$$

   These new disparities are now the ones of the disparity image in the orientation before rectification. The resulting matrix $D' = [d'_{ij}]_{i\in[1..m],j\in[1..n]}$ contains disparity values that correspond to depths according to the new perspective, but the indices of disparities in the array are still the same as in the original disparity image.

4. Additionally to the adaption of disparity values, the location of the disparities in the image has to be adjusted still. As with intensity images this is done by an

image rotation via homography:

$$d'_{ij} \mapsto d'_{kl} \tag{4.7}$$

$$\text{where } \tilde{K}\tilde{R}_C\tilde{K}^{-1}\begin{pmatrix} i \\ j \\ 1 \end{pmatrix} = \begin{pmatrix} k' \\ l' \\ w' \end{pmatrix} \sim \begin{pmatrix} k'/w' \\ l'/w' \\ 1 \end{pmatrix} = \begin{pmatrix} k \\ l \\ 1 \end{pmatrix} \tag{4.8}$$

In the implementation of this mapping the homography is actually applied in reverse: For each index in the target array one or multiple source indices in D' are computed. In the case that multiple values get mapped to the new location the value is determined by linear interpolation between the values of the source pixels.

A conceptually simpler way than this method would be to project the points in step 3 directly to their new coordinates with $\tilde{K}$, while computing the new disparities in the same way. Step 3 and 4 would then be combined. The problem of this alternative method is that it not only requires interpolation when multiple points are projected to the same coordinate, but that gaps where no point is being projected can form. Filling in these gaps would require extrapolation between disparities, which means that an additional numeric error is being introduced. Therefore we decided against using this method.

4.2.2 Fusion of Multi-Basline Disparity Images

After the preprocessing step of the previous section, the disparities of the rotated images are ready to be fused pixel-wise. We apply the fusion method proposed by Hirschmüller in section II.D of [1], which we will describe in this subsection. Hirschmüller does not describe a specific method for preprocessing disparity images there, but simply assumes that all images are projected onto a common plane that has the same distance to all optical centers (i.e. focal points). This is given after our rotation of disparity images; the common plane is the image plane of the unrectified reference camera.

We assume to have l different disparity images recorded at different baseline widths. For disparity $d^k_{\mathbf{p}}$ the index $k \in [1..l]$ denotes the image in which the disparity is to be found, and $\mathbf{p} \in [1..m] \times [1..n]$ is its pixel position (a single index is sufficient, rows and columns are not relevant in the following). As the depth given by a disparity depends on the baseline at which the image was recorded, the fusion works on disparities that have been normalized by their respective baselines t_k. The fusion computes the weighted average of disparities, but before the actual fusion step a simple outlier detection is being performed. Hirschmüller chooses to exclude all disparity values that differ more than one normalized pixel unit (i.e. $1/t_k$) from the median disparity.

The set of indices refering to valid disparities at a pixel $\mathbf{p}$ is then

$$V_{\mathbf{p}} = \left\{ k : \left| \frac{d^k_{\mathbf{p}}}{t_k} - \text{med}_{i\in[1..l]} \frac{d^i_{\mathbf{p}}}{t_i} \right| \leq \frac{1}{t_k} \right\} \tag{4.9}$$

As described in section 3.3.1 disparities can be set to a special value that marks them as invalid. These disparities are naturally also excluded from fusion. If $V_{\mathbf{p}} = \{\}$ the fused disparity at this position is also set to be invalid. In the fusion step the normalized disparities are weighted by their baselines to obtain fused disparities $d_{\mathbf{p}}$:

$$d_{\mathbf{p}} = \frac{\sum_{k\in V_{\mathbf{p}}} \frac{t_k\, d^k_{\mathbf{p}}}{t_k}}{\sum_{k\in V_{\mathbf{p}}} t_k} = \frac{\sum_{k\in V_{\mathbf{p}}} d^k_{\mathbf{p}}}{\sum_{k\in V_{\mathbf{p}}} t_k} \tag{4.10}$$

Choosing the baseline as a weight is a sensible choice. The differential of error in depth has been determined in [13] as $\delta z \approx -\frac{z^2 \delta d}{bf}$; it is antiproportional to the baseline width. Hirschmüller argues that the outlier filtering increases robustness, while the fusion itself increases accuracy of the resulting disparity image. We evaluate these claims in more detail in section 5.2.

Figure 4.2 shows the processing steps of disparity fusion for a scene that has been recorded by a side-facing camera. The resulting fused disparity image has, in most regions, more valid disparities than the disparity maps it has been computed from. As an example, the occlusion around the signpost to the left disappears.

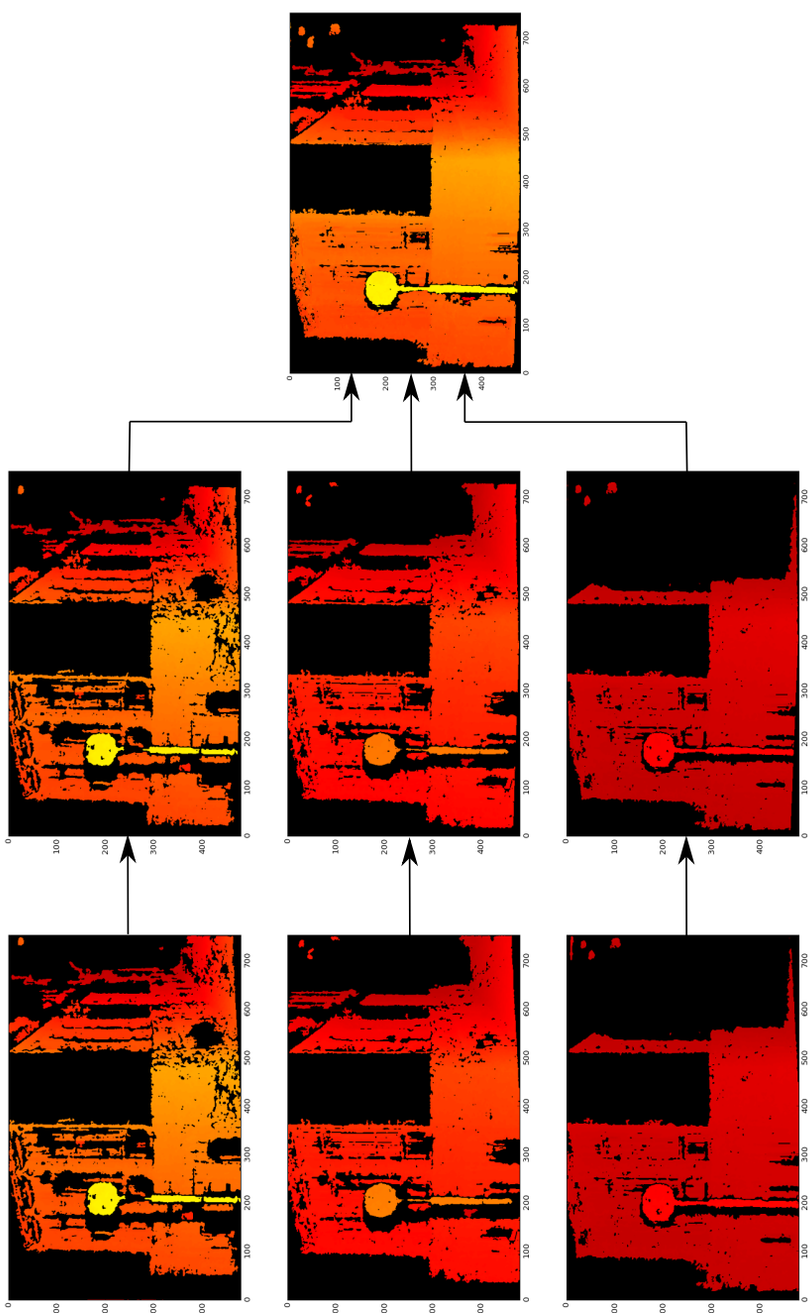


FIGURE 4.2: Fusion for side-facing VS disparity maps; the column on the left shows original disparity images; the one in the middle have been rotated according to section 4.2.1; the image on the right is the fusion computed by the method of section 4.2.2.

4.3 Virtual Stereo for a Front-Facing Camera

We applied the VS method in a second scenario with a different camera perspective. The camera was now facing forwards in relation to the vehicle. When driving straight ahead the camera was, in principle, translated purely along its z-axis. We have remarked already in section 3.2.3 that in this case the standard rectification procedure will break down. Even though this degenerate case will almost never happen in practice, as there are usually also smaller translations in x- and y-direction, tests with the standard procedure implemented in OpenCV (as the function `stereoRectify()`) returned no usable results. Instead of trying to adapt this recitification function for a front-facing camera, we decided to derive a completely new procedure.

The main reason to derive this new rectification procedure, namely *front-facing rectification*, is to have a procedure specifically for translation mainly along the z-axis. In this scenario the epipoles are on or close to the principal points of the images, and the epipolar lines are arranged radially around the epipoles. As with the standard rectification, we rotate the images so that the epipolar lines coincide with the rows of the image matrices, but naturally this is not possible for the whole image at once.

Instead, the procedure rectifies image regions that cover circular segments around the epipoles (see fig. 4.3, part (b) for an example of such an image region). Rectification and matching are performed per rectified image region. Unlike the standard recitification procedure, the front-facing rectification strongly distorts the images, especially in direction towards the epipole. We will explain this effect in more detail later, in section 4.3.2. The resulting disparity maps can be reprojected directly and independently, but they can also be rotated and possibly fused (in the case of overlaps between the regions) so that we obtain a disparity image in the original front-facing perspective without the distortion of the rectified image regions. Rotation and fusion of these segmented disparity images is performed by the procedures presented in section 4.2.1 and section 4.2.2.

As far as the author and his supervisor are aware, the front-facing rectification procedure is novel and has not been used for stereo matching yet. A natural reason that such a procedure was not developed until now, is that the scenario where one camera of the pair is placed behind the other can be easily achieved by the VS method, but hardly with a traditional fixed stereo rig. The field of view of the hind camera would then be mostly blocked by the front one.

The new rectification procedure presented here will rotate the camera perspectives by applying homographies, as with standard rectification. The main difference lies in the way of determining the new common image plane. We extend the notions of projective geometry introduced so far by those of vanishing points, lines, and planes. Epipolar points, lines, and planes will be considered as vanishing points, lines, and planes respectively. Based on this, the front-facing rectification procedure can then be derived easily.

4.3.1 Vanishing Points, Lines, and Planes

In section 2.1.1 we introduced points and lines in $\mathbb{P}^2$ in the form of homogeneous coordinates. Afterwards we also used representations of three-dimensional points, both in Cartesian coordinates of $\mathbb{R}^3$ or homogeneous coordinates of $\mathbb{P}^3$. The translation between these two representations happens analogous to the two-dimensional case. Even though image planes and epipolar planes have been quite important so far, we have never explicitely introduced planes in $\mathbb{P}^3$. We will make up for this omission now, but we will not develop the relations of join and meet between points, lines and planes in $\mathbb{P}^3$ as we did it in $\mathbb{P}^2$. It shall suffice to say that planes in $\mathbb{P}^3$ behave a lot like lines in

$\mathbb{P}^2$, also in relation to points. According to chapter 3.2.1 of [10], a plane $\boldsymbol{\pi}$ can be represented in homogeneous coordinates as $\boldsymbol{\pi} = (a\ b\ c\ d)^T$. The coordinates represent the plane equation $f(x_1, x_2, x_3, x_4) = ax_1 + bx_2 + cx_3 + dx_4 = 0$, which holds for any point $\mathbf{X} = (x_1\ x_2\ x_3\ x_4)^T$ on the plane. Concisely, this can be expressed as $\langle \boldsymbol{\pi}^T, \mathbf{X} \rangle = 0$. In section 2.1.2 we have introduced points and lines at infinity in $\mathbb{P}^2$. These transfer directly to points and planes at infinity in $\mathbb{P}^3$:

Definition 4.1 (Points and planes at infinity in $\mathbb{P}^3$)**.** Points in $\mathbb{P}^3$ of the form $(x_1\ x_2\ x_3\ 0)^T$ are said to lie *at infinity*. As is easily seen, they form a *plane at infinity* with the canonical representation $\boldsymbol{\pi}_\infty = (0\ 0\ 0\ 1)^T$.

We repeat just two basic results about the plane at infinity from the beginning of chapter 3.5 of [10] without proving them:

Corollary 4.1 (Parallel planes)**.** Two planes are parallel if, and only if, their line of intersection is on $\boldsymbol{\pi}_\infty$.

Corollary 4.2 (Parallel lines)**.** A line is parallel to another line, or to a plane, if the point of intersection is on $\boldsymbol{\pi}_\infty$.

In section 2.1.2 we have shown that points at infinity in $\mathbb{P}^2$ can be seen as directions. We constructed lines by a limit process where we started from a point and added multiples of a vector, representing a direction, onto it, until we arrived in the end at the direction vector itself. This point lay on the line at infinity. In $\mathbb{P}^3$ we can apply the same process to a point $\mathbf{X}$. We obtain $\lim_{\lambda \to \infty} \mathbf{X} + \lambda \mathbf{d} = \mathbf{d}$. The direction $\mathbf{d}$ is a point on the plane at infinity. This limit process forms a line in $\mathbb{P}^3$. We can project such lines in the world, which are usually called scene lines, onto the image plane of a camera:

Definition 4.2 (Vanishing point of a scene line)**.** Assuming a camera projection matrix $P = K[I|\mathbf{0}]$, the *vanishing point* of a scene line with direction $\mathbf{d}$ is $\mathbf{v} = K\mathbf{d}$, i.e. the vanishing point is the image of the point where the scene line meets $\boldsymbol{\pi}_\infty$. Scene lines with the same direction are parallel to each other and share the same vanishing point.

Vanishing points can also be constructed geometrically from a scene line, as the intersection of the line with the same direction that passes through the focal point of the camera. In definition 4.2 we assumed a reference camera. We should now consider directions and vanishing points for cameras in general positions and orientations in the world. First of all it is easy to see that the translation of a camera relative to the world coordinate origin has no influence on vanishing points: no matter how we translate a camera relative to a scene line, only the direction of the line has an influence on the vanishing point. Translating a camera relative to a line is equivalent to keeping the camera still and translating the line so that we get a parallel line with the same direction. Algebraically this is obvious from the fact that the last coordinate of directions, on which the translation would act, is 0.

The orientation of a camera, however, has an influence on vanishing points: directions are rotated by the extrinsic part of projection matrices just like any other point. Given two extrinsic camera rotations R_{C1}^T and R_{C2}^T, the direction $\mathbf{d}$ in the reference frame becomes

$$\mathbf{d}_1 = R_{C1}^T \mathbf{d} \tag{4.11}$$

$$\mathbf{d}_2 = R_{C2}^T \mathbf{d} \tag{4.12}$$

for cameras with the respective extrinsic rotations. The relative rotation R between the two directions is also the relative rotation between the cameras. We get it from eq. (4.11)

and eq. (4.12):

$$\mathbf{d}_1 = \underbrace{R_{C1}^T R_{C2}}_{=R} \mathbf{d}_2 \tag{4.13}$$

From directions we can calculate vanishing points, but we can also go the opposite way and get directions from known vanishing points:

$$\mathbf{v}_i = K\mathbf{d}_i \tag{4.14}$$

$$\Leftrightarrow \qquad \mathbf{d}_i = K^{-1}\mathbf{v}_i \tag{4.15}$$

In effect, we can calculate the relative rotation between two cameras simply from the vanishing points of the same scene line. In the computation of rectification rotations only this relative rotation is relevant. When it is encoded in vanishing points of the same scene line, the extrinsic part of the projection matrix is no longer needed explicitely. This result will be used in as part of the new rectification procedure, presented in section 4.3.2, for aligning the epipolar lines of the rectified image planes.

The last missing piece for the derivation of the front-facing rectification are vanishing lines. Unfortunately, lines in $\mathbb{P}^3$ are more difficult to represent in homogeneous coordinates than points and planes. Luckily we will be able to do without such representations for the results we need.

Definition 4.3 (Lines at infinity in $\mathbb{P}^3$)**.** *Lines at infinity* in $\mathbb{P}^3$ are the lines that join pairs of distinct points at infinity. Therefore they lie on the plane at infinity as well.

The relation between vanishing lines and lines at infinity is completely analogous to that between vanishing points and points at infinity. We can also construct them purely geometrically: from corollary 4.1 we know that all parallel planes intersect in the same line at infinity.

Definition 4.4 (Vanishing line of a plane)**.** The *vanishing line* of a plane is the image of the line at infinity that we get when intersecting the plane with π_∞. All parallel planes have the same vanishing line. The focal point of the camera lies on one of these parallel planes which we call *vanishing planes*. The line of intersection of the vanishing plane, to which the focal point is incident, and the image plane is the vanishing line of the vanishing planes.

We will not need to compute the coordinates of a line at infinity for rectification. Instead we will compute the normal of a plane (in Euclidean coordinates of a camera coordinate system) directly from its corresponding vanishing line. The following corollary is result 8.16 from [10]:

Corollary 4.3 (Plane normal from vanishing line)**.** The normal n of a plane π with vanishing line l on an image plane of a camera with intrinsics K is given by $n = K^T l$. The proof is straightforward: according to eq. (4.15) the line of projection of a point $\mathbf{x}$ on l has direction $\mathbf{d} = K^{-1}\mathbf{x}$. Clearly this line of projection is incident to the plane π, and therefore $\mathbf{d}$ and n are orthogonal to each other: $\mathbf{d}^T n = (K^{-1}\mathbf{x})^T n = \mathbf{x}K^{-T}n = 0$. For points on l we have $\mathbf{x}l = 0$, from this and the previous equation it follows that $l = K^{-T}n$, and therefore $n = K^T l$.

In the same way that vanishing points of a scene line tell us its direction relative to the cameras, vanishing lines tell us the orientation of a plane relative to the cameras. In the front-facing rectification procedure we use this result for rotating cameras so that their image planes are coplanar to such a vanishing plane.

4.3.2 Front-Facing Rectification Procedure

We have already mentioned the two main steps on which the rectification procedure is based: given the vanishing lines of the same plane in both images, we both rotate the image planes so that they are parallel to the vanishing plane and at the same time their x-axes point in direction of the respective vanishing points of a single scene line. This scene line is the baseline between the cameras. It intersects the image planes at the epipoles, which are close to the principal points in the front-facing scenario, because the two cameras perspectives are translated mostly in z-direction between each other. The vanishing lines we choose are two corresponding epipolar lines, so that the image planes are rotated to lie parallel to the vanishing plane, or respectively epipolar plane, of these lines. We are free to choose any one of these epipolar lines. The one in the other image will depend on the first; we will see shortly how it is derived from it. The angle at which these lines run through the epipole will determine the relative position of the image regions which are being rectified. As these epipolar lines determine the rectification rotation, we call them *rectifying epipolar lines*. See fig. 4.3 for a sketch of corresponding epipoles, rectifying epipolar lines, and the epipolar plane.

Summarizing, it can be said that we just need the two epipoles and two corresponding epipolar lines in the images to perform the recitification procedure. We assume camera 1 to be in front of camera 2. The projection matrices of the cameras are known from VO, therefore we can determine the epipoles $\mathbf{e}_1, \mathbf{e}_2$ directly from definition 3.2. We choose the rectifying epipolar line l_1 in the image of camera 1 freely. For determining the corresponding rectifying epipolar line l_2 in the other image, we first compute the fundamental matrix F via eq. (3.8) and then use result 9.5 from [10]:

Corollary 4.4 (Corresponding epipolar lines from F)**.** Let l_1 and l_2 be two corresponding epipolar lines (they lie on the same epipolar plane) and k any line in image 1 not passing through the epipole $\mathbf{e}_1$. Let $\mathbf{x} = [k]_\times l_1 = k \times l_1$, i.e. $\mathbf{x}$ is the point of intersection of the two lines. The epipolar line $l' = F\mathbf{x}$ in image 2 passes through the point corresponding to $\mathbf{x}$. We can choose k freely, so that $\mathbf{x}$ can be any point along l_1, and all corresponding points will lie on l'. Therefore l' is the epipolar line corresponding to l_1, i.e. $l' = l_2$. It is possible to interpret points in homogeneous coordinates as lines, as they share the same representation. If we do this for epipole $\mathbf{e}_1$, the inequality $\langle \mathbf{e}_1^T, \mathbf{e}_1 \rangle \neq 0$ means that the line $\mathbf{e}_1$ does not pass through the point $\mathbf{e}_1$. In effect, we can pick $k = \mathbf{e}_1$, and obtain $l_2 = F[\mathbf{e}_1]_\times l_1$.

Now we have all ingredients necessary for the actual computation of the rectifying rotations. For both images the camera matrix before the rectification rotation is K, and afterwards $\tilde{K}$. The front-facing rectification procedure works independently for both cameras, in each case the following steps are performed:

Definition 4.5 (Front-facing rectification procedure)**.**

1. Compute the plane normal corresponding to the rectifying epipolar line in Euclidean coordinates of the camera coordinate frame: $n = K^T l$ (corollary 4.3). Normalize this vector: $n \leftarrow \frac{n}{||n||}$.

2. Compute the direction of the baseline in the camera coordinate frame as $\mathbf{d} = K^{-1}\mathbf{e}$ (eq. (4.15)). Normalize this vector: $\mathbf{d} \leftarrow \frac{\mathbf{d}}{||\mathbf{d}||}$.

3. Choose three orthonormal base vectors as the columns of $\tilde{R}^T = [r_1\ r_2\ r_3]$:

   - The new z-axis is, as already explained, in direction of the normal of the vanishing plane. Therefore we pick $r_3 = n$.

- The new x-axis should be in direction of the baseline, which is orthogonal to the new z-axis as required. Therefore we pick $r_1 = \mathbf{d}$.
- The new y-axis needs to be orthogonal to the other two axes. Therefore we pick $r_2 = \mathbf{d} \times n$.

The construction of the rotation matrix should be explained in a bit more detail. It follows the same idea of the construction of the rotation matrix in section 3.2.3. The baseline is incident to the vanishing/epipolar plane corresponding to the rectifying epipolar line. This means that n and $\mathbf{d}$ are orthogonal to each other. The rectification should rotate the image plane so that its z-axis is in direction of the plane normal of the epipolar plane, because then they are parallel. This means that the (passive) rectifying rotation $\tilde{R}$ should fulfill $\tilde{R}n = (0\ 0\ 1)^T$. The (active) extrinsic rectification transform we constructed is its transpose $\tilde{R}^T$; based on the choice of base vectors it is clear that this condition is met.

For a better intuitive understanding of the procedure, it is worth to take a closer look at the rectifying epipolar line. Figure 4.3, part (b) shows such a line (dashed and and marked as l). First of all it should be noted that we have to consider its orientation which is given by its sign. Multiplying its representation in homogeneous coordinates by -1 is not an equivalence in this case, because this inverts the direction of the plane normal n computed from l in step 1. Hence the line with the opposite sign will rectify the image region on its opposite side. Furthermore, in the new, rectified image plane this epipolar line lies at infinity. This can be shown by a similar argument to the one of section 3.2.3 where the epipoles were shown to be at infinity: each rectifying epipolar line is an image of the vanishing plane used for rectification; this plane is parallel to the rectified image planes, therefore they intersect at infinity, and the rectifying epipolar lines are exactly the lines of intersection at infinity (corollary 4.1). The epipoles lie on the rectifying epipolar lines, hence they are also at infinity in the rectified image plane. Therefore, regions covering the epipoles can not be rectified with this procedure, but this is not a flaw in its design. In the front-facing scenario an object that does not move relative to the camera will be displaced by a smaller distance between images the closer it is to the epipoles. For objects that are very close to the epipoles the disparities can no longer be computed, as they would be below one pixel. Rectifying image regions covering the epipoles would therefore be useless anyways.

When the rectification rotation is applied via homography (see section 3.2.4), the rectified image regions closer to the epipole are interpolated from fewer pixels of the original, due to the distortion effect occuring when the epipole is at infinity (see fig. 4.4). In the rectification homography $H = \tilde{K}\tilde{R}'K^{-1}$ we picked the camera matrix $\tilde{K}$ by experimentation. The focal length was greater than in the orignal K so that the image region was downscaled and a wider region was mapped. For the same reason the image matrices after rectification were chosen to be wider in u-coordinates. The matching and reprojection of images that were rectified by the front-facing rectification procedure functions the same way as for images rectified by the standard rectification procedure. The resulting reconstructions are comparable in quality to those based on the images of a fixed stereo rig. In section 5.1 we evaluate this claim in more detail.

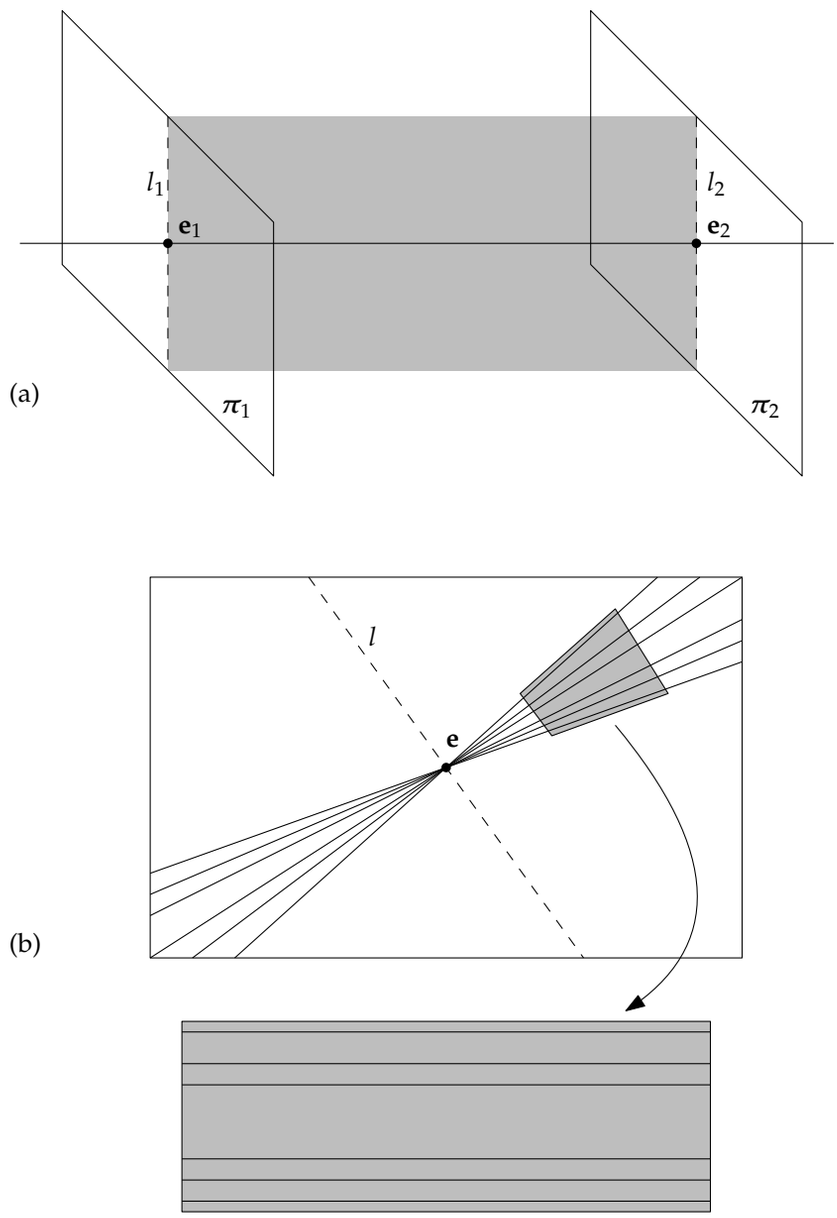


FIGURE 4.3: Rectification for front-facing VS. Part (a) shows the typical epipolar geometry between two image planes in the front-facing scenario. The grey plane is the epipolar plane of rectifying epipolar lines l_1 and l_2, the baseline is shown running through the epipoles $\mathbf{e}_1$ and $\mathbf{e}_2$. Part (b) shows an image region before and after rectification. The rectifying epipolar line l and epipole $\mathbf{e}$ are shown in the original perspective, before rectification. For better illustration some other epipolar lines running through the rectified image region are shown additionally.

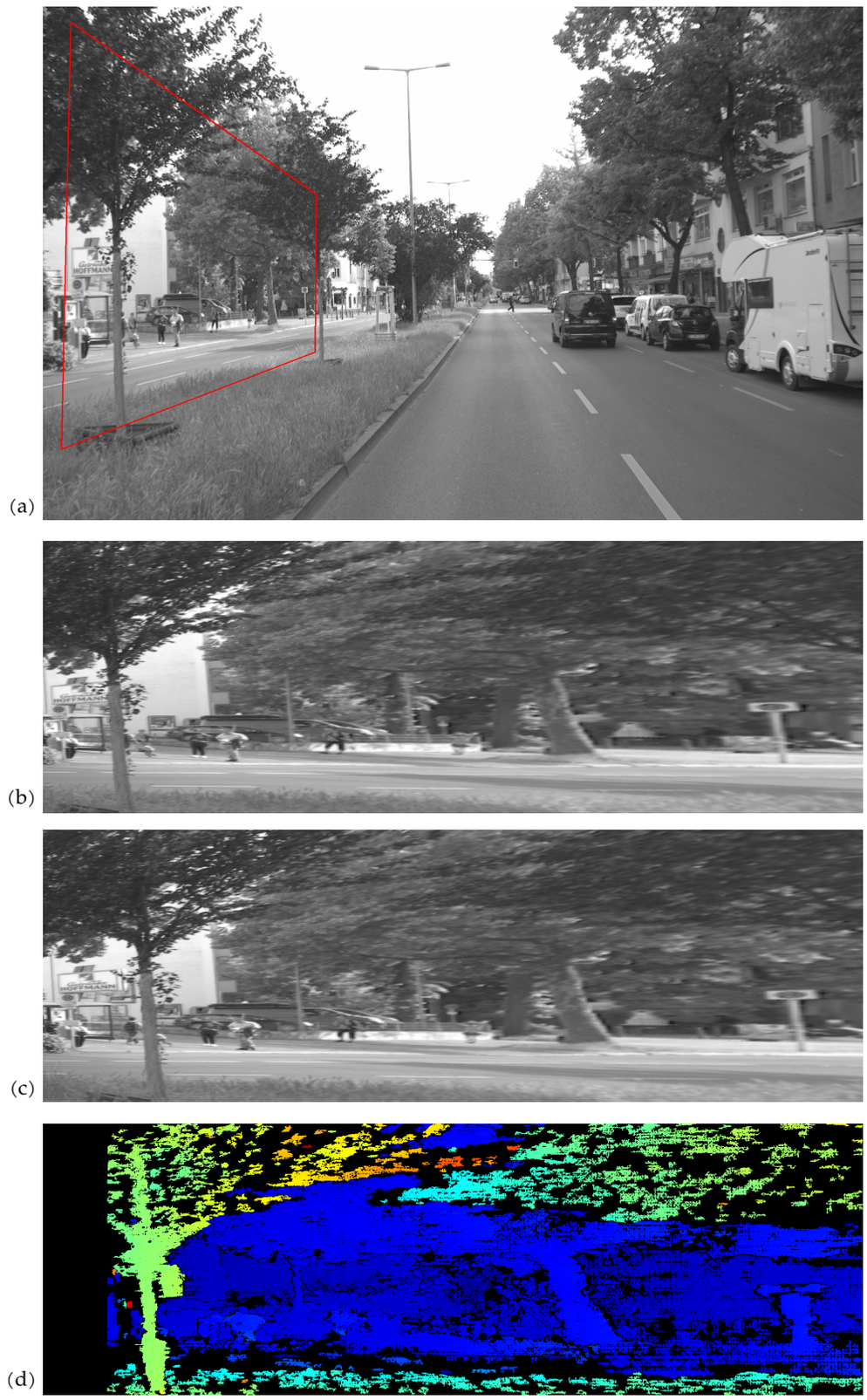


FIGURE 4.4: Processing of images in front-facing VS. Part (a): Original image with image region to be rectified marked in red; part (b): Rectified image region of (a); part (c): Rectified image region of previous frame (hind camera); part (d): Disparity image computed from (a) and (b), the colours represent the depths. From close to far: red, orange, yellow, green, blue.

Chapter 5

Evaluation of Virtual Stereo

The following chapter evaluates the Virtual Stereo method empirically to prove its applicability in the real world.

The first section describes one experiment carried out to show how the Vitual Stereo method with the front-facing rectification procedure can be applied to a video filmed in a typical automotive scenario. The hardware setup and parameters of the processing steps are specified. Subsequently, the results of the processing with VS are compared to those of a fixed stereo rig. Disparity images of both methods can be evaluated against each other, because one of the cameras played a double role. It recorded images as the reference camera of the fixed rig and the same image sequence was used as input data for VS.

In the second section the fusion of disparity images, as a part specific to the side-facing VS method, is evaluated. Side-facing VS was not evaluated as a whole, because we lack reliable data to compare our results against. On the upside, we were able to evaluate the fusion in isolation based on the Middlebury dataset of 2006, a pre-rectified dataset with precise calibration and ground truth. We show the improvement from simple disparity images with one baseline to fused images of multiple varying baselines. The analysis was based on a receiver operating characteristic specific to disparity images; it relates density and error rate of disparity images in a function graph.

5.1 Evaluation of Front-Facing Virtual Stereo

In this section we evaluate the results of the front-facing Virtual Stereo version on a video that has been recorded in a typical urban scenario. In order to offer the possibility to reproduce this experiment we first provide the specifics of the hardware setup and the parameters used for processing the data with DSO and rectification of image regions. In the evaluation itself we compared disparity images obtained by matching between the two cameras of the stereo rig and ones computed with the front-facing VS method described in section 4.3. After describing our evaluation method and the recorded scenario in more detail we present both quantitative and qualitative results of the evaluation.

5.1.1 Hardware Setup

This subsection presents the hardware setup used for recording the video. Two monocular cameras were recording as part of a calibrated stereo rig mounted on top of a common four-wheeled road vehicle. The intrinsic camera matrices were computed using OpenCV calibration functions. The extrinsic rotation and translation between the cameras of the pair were computed via a proprietary automatic calibration procedure that has been proven to provide results that are comparable in accuracy to those reached with common calibration procedures using calibration patterns. Before any further processing steps, the distortion effect of the lenses were reverted according to the widely used “plumb bob” distortion model proposed by Brown in [14]. The images were recorded

at $30Hz$, triggered by a signal transmitted synchronously to both cameras; the images of left and right camera were therefore recorded close together in time which is necessary for precise matching, especially in this scenario where the environment is moving relative to the cameras. If the cameras were triggered asynchronously, moving objects would be displaced between left and right image in relation to the global coordinate frame. Another property of the cameras, necessary for precise matching due to a similar reason, is global shutter. As opposed to rolling shutter the light intensity values of all image cells are acquired synchronously with global shutter; if this were not the case there would be a displacement of objects within a single image and different parts of the image would depict objects at different positions according to when they were recorded. Furthermore the exposure times of the images were regulated automatically for each camera according to the overall brightness. All further relevant properties of the hardware setup are gathered in table 5.1

Camera and sensor	
Make	Basler acA4096-30uc
Sensor	Sony IMX267
Sensor type	CMOS
Colour type	RGB (converted to greyscale)
Bit depth	12 Bit
Sensor size	$11.3mm \times 7.1mm$
Pixel size	$5.86\mu m \times 5.86\mu m$
Resolution	$1920px \times 1200px$
Shutter	global
Optics	
Make	Schneider Cinegon 1.9/10
f-number	1.9 (at nominal focal length)
Focal length (nominal/calibration)	$10.4mm/10.9mm$
Distortion model for calibration	plumb bob (polynomial model with 2 radial and 3 tangential parameters)
Recording parameters	
Baseline width	$49.57cm$
Exposure time	automatic
Trigger	synchronous hardware trigger
Image capture frequency	$30Hz$

TABLE 5.1: Hardware setup for front-facing VS recording.

5.1.2 Processing Setup

Before further processing by DSO and stereo matching, the images, which were recorded in RGB colour space, were converted to greyscale images with a depth of 8 Bit. As already mentioned in section 4.1.2, for obtaining the camera trajectories we used a proprietary modification of DSO called stereo-DSO. The main motivation to use this variation of DSO was to compute the trajectory data in the unit of meters, because only then we are able to compare the disparity images of Virtual Stereo and the ones computed from the fixed stereo rig. Additionally to the scaling in meters, stereo-DSO is usually more robust in computing trajectories, e.g. obvious divergences from the track when the vehicle stopped intermittently were a frequent problem with standard DSO that did not

Minimum disparity	1
Maximum disparity	256
Block Size	3
Uniqueness Ratio	15%
Window size for speckle filtering	200
Speckle range	2
Disparity smoothness parameter P1	$8 * \texttt{blockSize}^2$
Disparity smoothness parameter P2	$32 * \texttt{blockSize}^2$
Number of directions for DP solution of smoothness constraint	5

TABLE 5.2: Parameters for OpenCV stereo matcher of class `StereoSGBM` used in front-facing Virtual Stereo.

happen with stereo-DSO. However, we did not see significant improvements in accuracy from standard DSO to stereo-DSO; the standard method also provides trajectories usable for rectification and matching via Virtual Stereo.

Stereo-DSO, like standard DSO, can use additional intrinsic calibration parameters. These are three forms of photometric calibration, the first of which acounts for the non-linearity of sensor cell response to light intensity; the second, vignetting, corrects irregular attenuation of light by the lens; and the third scales intensities according to exposure time. Due to the lack of resources for photometric calibration, these parameters could not be used for the experiment and we left the images photometrically uncorrected. For all other parameters that exist both in standard and stereo-DSO, like number of keyframes and tracked points, the default values from the implementation of DSO were used.

The parameters to be defined for front-facing rectification, as described in section 4.3.2, are the intrinsic camera matrix $\tilde{K}$, the size of the rectified images, and the angle of the rectifying epipolar line for the reference image. These parameters strongly depend on the design and calibration of the camera used for the recordings, therefore we see no sense in giving the exact choice of values at which we arrived by experimentation, because they would only apply to our specific camera construction and calibration. With our choice of these parameters we aimed to maximize the areas of rectified image regions depicting the sides of the road. In order to rectify onto an image plane showing a side of the road we chose the rectifying epipolar line to be vertical. For depicting a region spanning from close to the outer border of the original image to the epipole near the center we scaled the image region down by choosing the focal length of $\tilde{K}$ to be greater than that of K. For the same reason we used wider image matrices for the result of the remapping via homography.

For the initialization of the stereo matcher instance (of the class `StereoSGBM`) we selected a fairly common set of parameters; it is gathered in table 5.2.

5.1.3 Comparison of Stereo Rig and Virtual Stereo Results

As basis for comparison to the front-facing VS results we could not rely on high precision ground truth data. The most accurate technique for gathering data in scenarios such as ours is via LIDAR (light detection and ranging), but due to the high cost and effort to calibrate and operate such systems, acquiring data with this precision was out of the scope of this thesis. Among the publicly available data sets for the evaluation of stereo matching algorithms the most fitting one for our scenario would have been the KITTI scene flow benchmark of Menze, Heipke, and Geiger [15, 16]. The multi-view extension of this data set includes sequences of twenty consecutive frames together with

high precision depth ground truth. Unfortunately, the low number of frames in these sequences makes processing with DSO unfeasible; often a higher number is already required for the intialization of a trajectory.

The best available option for evaluation of front-facing VS were our own recordings with cameras as part of the stereo rig described in section 5.1.1. All processing steps of VS were executed only on images of the left camera. The resulting disparity images were compared with those based on the same reference images, but matched against the synchronously recorded images of the right camera of the rig. Our evaluation proceeded according to the following three steps:

1. Directly after matching, the disparity images of VS were still in the distorted perspective (see part (d) of fig. 4.4). They were rotated (as desribed in section 4.2.1) into the perspective of the left camera after rectification with the right one, i.e. afterwards they were in the same perspective as the disparity images of the rig.

2. The baseline width of the VS data varies for each disparity image. We normalized all disparity images, both obtained from the rig and via VS, by dividing their values by the respective baseline of the pair (see also eq. (3.20)). In effect, all normalized disparities are scaled as if they were taken with the same camera intrinsics but at a baseline of one meter. The resulting disparities were calculated in a floating point type in order to minimize the loss of accuracy.

3. Now that both types of disparity images were in the same units, the error between them was ready to be calculated. We calculated it only on pixels where both images had valid disparity values. The alternative would have been to also evaluate errors where only one of both types of disparity images is valid. This would have been a misleading error measure however: there might be areas that are occluded between the images of the VS pair, but not between the images of the rig. Naturally it is unfair to count occluded, and therefore unmatchable, pixels into an error measure. On the other hand, VS might perform well on areas that can not be matched in the image pair of the rig. We computed three distinct error measures, all based on the differences between the disparity values of the two images: The mean absolute error, the median absolute error, and the error rate when counting all disparities with a difference of greather than 2 (in the normalized disparity unit) as an error.

The recording we evaluated took place along a route of some hundreds of meters through an inner city street. The route was mostly straight and includes one 180° curve. The vehicle on which the camera was mounted drove at varying speeds, up to ca. $30 km/h$. The recording took place on a sunny day, so that the scenery was generally well illuminated. The video recorded static objects, such as road surface, trees, parked cars, building facades, etc., and moving objects, such as bicyclists, driving cars and passengers. For evaluation we exclusively used image regions that depict objects which do not move independently from the rest of the scene. We selected the data for evaluation in this way, as the pairs of images to be matched in VS are recorded asynchronously and therefore independently moving objects are never reconstructed correctly (see also section 5.1.1 for similar problems with rolling shutter and asynchronous trigger). The evaluated image regions include complex structures, such as leaves of grass and trees, as well as scaffolding consisting of many segments. We selected the image regions so that there were only few glossy surfaces in them, because stereo matching generally performs poorly on such textures. We estimate all objects to be in a range of distances between 5 and $30m$ from the reference camera.

Error measures			Baseline	Disparities	Scene description
Mean $[px]$	Median $[px]$	Rate $[\%]$	$[cm]$		
4.15	2.66	48.99	10.7	176079	driving through tight
3.20	2.13	43.57	21.2	263192	curve; cars and tree in
2.82	1.79	38.95	32.2	234728	front of building facade
2.76	1.55	36.16	23.1	314909	slight swerve; building with
2.00	1.13	23.53	47.0	292841	scaffolding, signpost, poles
2.47	2.00	46.77	69.9	264520	streetsign (rather complex)
5.32	3.76	46.50	26.3	154787	driving straight ahead;
8.12	7.51	70.75	50.8	148250	trees, street, lamppost,
6.22	5.73	71.51	77.3	122143	and patches of grass

TABLE 5.3: Errors measured between disparity images of fixed stereo rig and front-facing VS for three different scenes. The error rate was computed with a tolerance of $2px$. The reference images were the same per scene, the match images of VS were for each scene the three frames following the reference image.

In table 5.3 the results for three different scenes are given. For each one, the basis of comparison was the disparity image of the rig with the left image as the reference image. The same reference image was used for VS, it was being matched against the three subsequent frames of the same camera, so that we got disparity images with widening baselines. The numeric results of the evaluation need to be considered with some precautions. First of all, it needs to be taken into account that both the DSO trajectory and the disparity images are subject to errors with unknown variances. Therefore, the computed error measures can only give some rough estimate that might be the basis for future evaluations. The effect of varying the baseline width is also hard to quantify. The set of valid disparities of the VS images differs with the width: wider baselines are often better at matching objects at greater distances while closer objects then appear no longer in both images and therefore are not being matched anymore. Equation (5) of [13] ($\delta z \approx -\frac{z^2 \delta d}{bf}$) states that disparity errors are proportional to the absolute depth error, but also proportional to the square of the depth itself. Therefore, the errors in images taken with wider baselines will be punished stronger, because there predominantly objects with greater depths have been matched. Ultimately, it can be said that the numerical evaluation remains inconclusive when it comes to pointing towards possible improvements of the VS method.

Luckily, the quality of the results can be easily judged by the naked eye. We compared the reprojected points of the two methods. In the range of up to ca. $5m$ both reconstructions were close to the intuitively estimated structure of objects, the objects reconstructed from both methods were usually very similar and the differences there were limited to a range of a few centimeters. We noticed that there are also structures, such as grass, that can frequently be matched via VS but not via the traditional method. Three exemplary scene comparisons are depicted in fig. 5.1. At greater distances the quality of reconstruction degrades quicker for VS than for the traditional method. We consider this not to be a grave disadvantage, as a range of a few meters is often enough to recontruct all relevant objects that are part of a road scenery, especially in cities.

FIGURE 5.1: Comparison of point clouds between front-facing VS and stereo rig. The points of the rig are green, the ones of VS blue; the street (in the lower image regions) was not in view of the rectified region of VS. The grassy ground regions surrounding the trees (in (a) and (b)) and the pole (in (c)) are reconstructed only by VS.

5.2 Evaluation of Disparity Image Fusion

For the evaluation of side-facing VS, no comparison to data from a fixed rig was possible, because all our recordings from this perspective were taken with a monocular camera. Judging by intuitive depth perception of the scenes we evaluated, the side-facing VS delivers results similar to those of front-facing VS of the previous section. Due to the similarity between the scenarios and results of front and side-facing VS, we will not go into the qualitative details of the reconstruction obtained in the side-facing case. There is however one part of side-facing VS that can be evaluated, even numerically, namely disparity fusion as described in section 4.2.2. Disparity fusion is not a novel technique, and in fact taken over identically from [12], but Hirschmüller does not evaluate the effect of disparity fusion on the development of error rate and density as compared to disparity images computed from a single pair of images. We made such an evaluation on the Middlebury multi-view dataset of 2006 acquired by Hirschmüller, Scharstein, and Pal and described in [4, 5]. Compared to our intended automotive scenario this dataset consists of images which depict smaller objects at a closer range to the camera. Regardless of this fact, we estimate that the complexity of the scenes is similar enough to the cityscapes evaluated in the previous subsection to expect similar effects there. We evaluated the effect of disparity fusion on the error rate and density of images by comparing fused and simple disparity images via a variety of the receiver operating characteristic. The next subsection introduces this method of analysis; it is followed by evaluation results on the aforementioned Middlebury dataset.

5.2.1 Receiver Operating Characteristic for Disparity Images

Receiver operating characteristics (ROC) usually apply to binary classifiers where they relate *true positive rates (TPR)* to *false positive rates (FPR)*. In the case of disparity images, no clear separation in two classes can be made when defining errors. One attempt would be to make the distinction between valid and invalid disparities; then any class difference between ground truth and test image would be counted as an error. This definition would, however, neglect the common case when disparities of both images are valid but differ by a big margin, clearly an error as well. Our definition of ROC for disparity images follows that of Hu and Mordohai from [17]. The TPR, or sensitivity, corresponds roughly to density in this definition. A higher density means that a greater area of valid ground truth pixels are coverered by valid test image disparities. The FPR, which is the inverse of specificity (1 - FPR), does not correspond directly to our definition of error rate. For disparity images, errors are measured only on pixels where both images exhibit valid disparities. As is usual for ROC evaluation, not just one ratio between these two proportions is calculated, instead they are plotted against each other in a curve whose data points are formed by varying a parameter of each of the algorithms. In [17] the varied parameter is the measure of *confidence* in a disparity. A high confidence means that the computed disparity is likely reconstructing its point in the world close to reality. The paper compares the performance of multiple confidence measures in terms of their ability to detect whether a disparity is valid and close to the ground truth. Comparing confidence measures is expedient for the design of matching algorithms, but in our case the confidence measures are already implicitly given by the definition of the matching and fusion algorithms. In these algorithms, a confidence threshold defines the minimum confidence in a disparity which is necessary for it to be considered valid. For both SGBM and disparity fusion, classification into valid and invalid disparities has already been mentioned. In section 3.3.2 an overview of the disparity computation via SGBM in OpenCV was given. The following measure decides whether a disparity computed with this algorithm is valid:

Definition 5.1 (Uniqueness ratio as confidence measure)**.** `StereoSGBM` uses a *uniqueness ratio* between the two best candidate disparities for a pixel as a threshold of confidence. Given the matching cost for the best disparity candidate as c and for the second best as c', a confidence threshold t classifies a disparity as valid if $c \leq c'(1 + t/100)$, i.e. the threshold is a percentage that spans a range from equality to half of the second best disparity cost as acceptable margins for validity.

For the disparity fusion defined in section 4.2.2, a disparity was marked as invalid if the set of valid disparities $V_{\mathbf{p}}$ was empty. Naturally, a higher number of disparity indices in this set should signify a greater confidence in the fused result. Additionally, the disparities were weighted by their baselines in the fusion, therefore disparities with wider baselines should also lead to a higher confidence when included in the calculation. For the ROC evaluation we defined the following confidence measure for fused disparities:

Definition 5.2 (Sum of baseline weights as confidence measure)**.** The *sum of valid baseline weights* of a fused disparity $d_{\mathbf{p}}$ is $\sum_{k \in V_{\mathbf{p}}} t_k$.

Given those measures, we compared a simple disparity image to one that had been obtained via fusion of multiple disparity images based on the same reference image but matched with different images at varying baselines. We started by including only disparities with the highest confidences in the respective measure, i.e. uniqueness ratio for the simple image and the sum of baseline weights for the fused image. For the simple disparity image the uniqueness ratio threshold was decreased by steps of 10% starting at 100% (highest confidence) and going down to 0% (lowest confidence). For the fused image the steps were given by the various sums of weights that occured in valid fused disparities, e.g. disparities of the highest confidence were fused from all baselines, the ones with second highest confidence did not include the shortest baseline, etc. In both cases, the step-wise lowering of the confidence threshold successively included more and more disparities into evaluation until all valid disparities of the test image were included. The sets of evaluated disparities and the density and error rate, which were evaluated on these sets, were defined as follows:

Definition 5.3 (Ground truth and set of evaluated test disparities)**.** Let D be the set of all valid pixels in the test image ($d_{\mathbf{p}}$ is the disparity value of a pixel $\mathbf{p} \in D$), $\mathrm{conf}(d_{\mathbf{p}})$ the confidence of $d_{\mathbf{p}}$, and G the set of all valid ground truth pixels ($g_{\mathbf{p}}$ is the disparity value of a pixel $\mathbf{p} \in G$). Define the function $\mathrm{conf}_t(S) := \{\mathbf{p} : \mathbf{p} \in S, \mathrm{conf}(d_{\mathbf{p}}) \geq t\}$, i.e. the set of all pixels in S with a confidence greater than or equal to t. Then the set of test pixels taken into consideration at threshold t is $D_t := \mathrm{conf}_t(D) \cap G$.

Even though most pixels in the ground truth images we used are valid, there are always some small areas that form an exception to this rule. It is self-evident that no statement can be made about test disparities where the ground truth is invalid. Therefore we excluded such pixels from the set D_t in definition 5.3. Errors are only evaluated on pixels where both the test image and the ground truth image are valid.

Definition 5.4 (Density and error rate)**.** The density at threshold t is $|D_t|/|G|$. All disparities differing by more than 1 from the ground truth were considered to be errors. Therefore the set of errors is $E_t := \{\mathbf{p} : \mathbf{p} \in D_t, |d_{\mathbf{p}} - g_{\mathbf{p}}| > 1\}$. The error rate at threshold t is $|E_t|/|D_t|$.

5.2.2 ROC Evaluation of Simple and Fused Disparity Images

The Middlebury multi-view dataset of 2006 [4, 5], on which we evaluated the difference between simple and fused disparity images, consists of 7 different images for each scene.

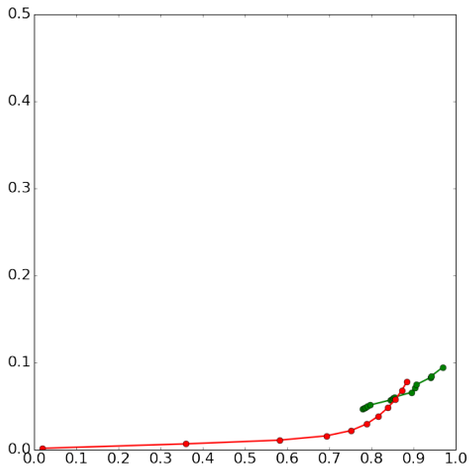
In our evaluation we used the full-resolution images with a height of $1110px$ and a width between $1240px$ and $1396px$. Same as the authors of the dataset, we use the indices $[0..6]$ to denote the pictures from the left to the right. They were taken at a regular distance of $40mm$ between neighbouring pairs. The ground truth data was provided only as two disparity images per set, computed between image 1 and 5. We only used one of these ground truth images; the one where image 1 was the reference. The baseline width of the ground truth is $160mm$, and in our comparison we normalized all test images to this baseline width. All images were undistorted and pre-rectified, and the calibrated camera intrinsics had a focal length of $3740px$. No other calibration data is provided by the authors of the dataset.

For the simple disparity image we matched image 1 against 5 with an instance of `StereoSGBM` using mostly the same parameters as in section 5.1.2. For the fused disparity image we matched image 1 against all other six images and subsequently fused them by the method presented in section 4.2.2. Here the same matcher as for the simple disparity image was used, with one difference: whereas in the simple case we varied the uniqueness ratio as the variable confidence measure threshold, in the multi-baseline case it had to be fixed to one value. We tried different values in in the recommended range from 5% to 15% but saw barely any differences in the results; therefore we fixed this value to 15%. Another difference to the matcher of table 5.2 is that we turned of speckle filtering for the ROC analysis. The reason for this is that we wanted to limit post-processing as much as possible, in order to have a better comparability between the two disparity images. All parameters of the matcher are presented in table 5.4. We

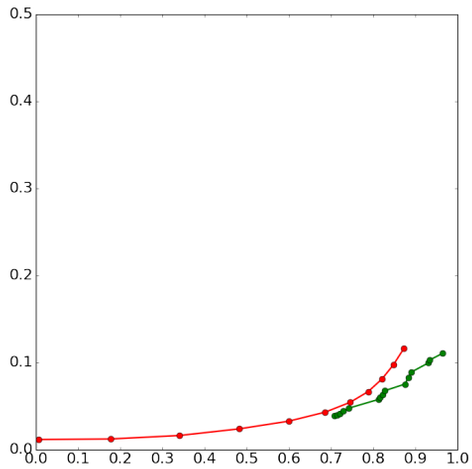
Minimum disparity	1
Maximum disparity	256
Block Size	3
Uniqueness Ratio	15% (multi-baseline) / 0%-100% (simple)
Speckle filtering	none
Disparity smoothness parameter P1	$8 * \texttt{blockSize}^2$
Disparity smoothness parameter P2	$32 * \texttt{blockSize}^2$
Number of directions for DP solution of smoothness constraint	5

TABLE 5.4: Parameters for OpenCV stereo matcher of class `StereoSGBM` used in ROC analysis.

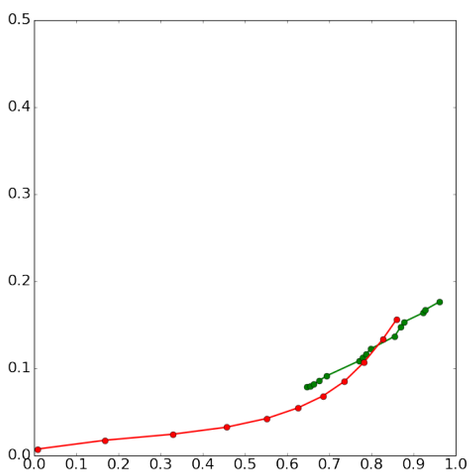
did not perform the evaluation on all scenes of the dataset, instead we chose scenes that offered a variety of textures so that our selection should be representative for a wider range of similar scenes of the whole Middlebury dataset. In total we evaluated six scenes (Aloe, Baby3, Bowling2, Flowerpots, Midd1, and Wood1). The scenes differ strongly in the difficulty they pose to the matcher as can be seen in the variance of the slopes in fig. 5.2. The fused images generally reach higher densities than the simple disparity images, but it should be remarked that the simple ones are often better in terms of error rates at lower densities. Only in the scene Baby3, the fused image had an overall lower error rate in comparison. We conclude that disparity fusion is sensible, especially when a higher density shall be reached. In this case the confidence threshold must be set to be more permissive, which leads to fewer disparities being filtered out. Compared to single baseline matching, the fusion method retains a lower error rate at these higher densities.



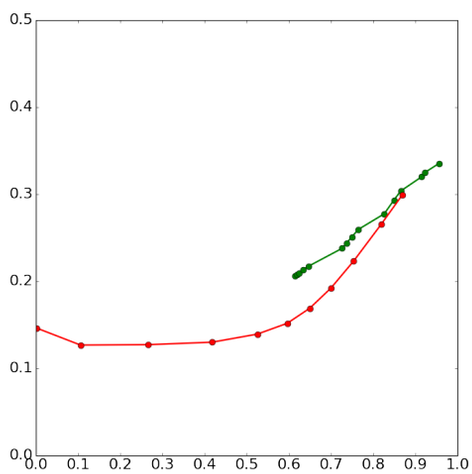
(A) Aloe



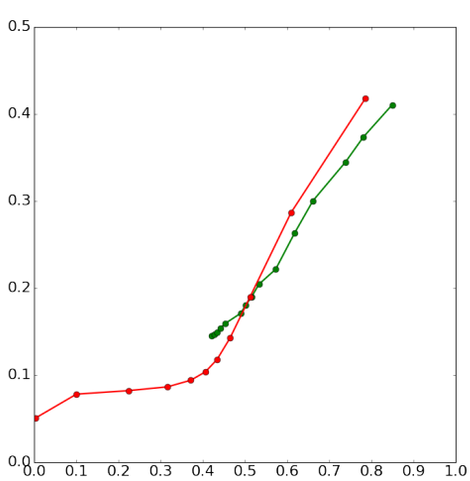
(B) Baby3



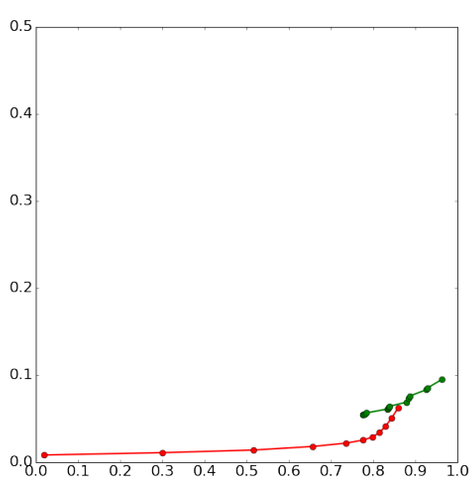
(C) Bowling2



(D) Flowerpots



(E) Midd1



(F) Wood1

FIGURE 5.2: ROC curves of evaluated scenes; the horizontal axes show the density, the vertical the error rate; the curve for the simple disparity image is in red, the one for the fused image is in green.

Chapter 6

Conclusion & Outlook

The proposed Virtual Stereo method has been shown to work on real-world scenarios, but as usual, there is much room for improvements. First of all, the choice of image regions to be rectified in front-facing VS could be rendered more convenient. Instead of choosing a rectifying epipolar line and a new intrinsic matrix, simply defining the regions directly and computing the matrix and rectifying line based on this would make the implementation much more accessible. Also in this scenario, a simple image rotation could align the epipoles so that the are coincident with the principal point. If this is not the case, the rectified regions vary based on the extrinsic transformation between the frames.

Apart from these points, which add usability but no deviations from the proposed method, some other approaches to VS came up during its development.

For instance, an alternative method to determine the fundamental matrix and epipoles, used for front-facing rectification, by other means than a fully fledged and computationally expensive VO algorithm. One possibility would be to compute the fundamental matrix based on point correspondences and the epipoles from optical flow (as the center of expansion of the flow).

Another proposal that came up was to use a fisheye camera model for the front-facing rectification. The motivation is to mitigate the strong distortion effect towards the epipole.

There are still other improvements possible without introducing new elements to VS. The disparity fusion has so far been only applied to the side-facing scenario, but it would lend itself just as well to an application to the front-facing one.

We hope that some of these ideas will be explored further on in future research.

Bibliography

[1] Heiko Hirschmüller. “Stereo Processing by Semiglobal Matching and Mutual Information”. In: *Transactions on Pattern Analysis and Machine Intelligence* 30.2 (2008), pp. 328–341.

[2] Onur Özyeşil et al. “A Survey of Structure from Motion”. In: *Acta Numerica* 26 (2017), pp. 305–364.

[3] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. “A compact Algorithm for Rectification of Stereo Pairs”. In: *Machine Vision and Applications* 12.1 (2000), pp. 16–22.

[4] Daniel Scharstein and Chris Pal. “Learning Conditional Random Fields for Stereo”. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8.

[5] Heiko Hirschmüller and Daniel Scharstein. “Evaluation of Cost Functions for Stereo Matching”. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8.

[6] Jakob Engel, Thomas Schöps, and Daniel Cremers. “LSD-SLAM: Large-Scale Direct Monocular SLAM”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 834–849.

[7] T. Sato et al. “Dense 3-D Reconstruction of an Outdoor Scene by Hundreds-baseline Stereo Using a Hand-held Video Camera”. In: *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV)*. 2001, pp. 57–61.

[8] Jorge Stolfi. “Oriented projective geometry”. In: *Proceedings of the third Annual Symposium on Computational Geometry*. ACM. 1987, pp. 76–85.

[9] Daniel Scharstein, Richard Szeliski, and Ramin Zabih. “A taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms”. In: *Workshop on Stereo and Multi-Baseline Vision*. IEEE, Dec. 2001, pp. 131–140.

[10] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Second Edition. Cambridge University Press, 2003.

[11] Stan Birchfield and Carlo Tomasi. “A Pixel Dissimilarity Measure that is Insensitive to Image Sampling”. In: *Transactions on Pattern Analysis and Machine Intelligence* 20.4 (Apr. 1998), pp. 401–406.

[12] Jakob Engel, Vladlen Koltun, and Daniel Cremers. “Direct Sparse Odometry”. In: *Transactions on Pattern Analysis and Machine Intelligence* 40.3 (2018), pp. 611–625.

[13] Chienchung Chang and Shankar Chatterjee. “Quantization Error Analysis in Stereo Vision”. In: *Conference Record of the Twenty-Sixth Asilomar Conference on Signals, Systems & Computers*. IEEE. 1992, pp. 1037–1041.

[14] Duane C. Brown. “Decentering Distortion of Lenses”. In: *Photometric Engineering and Remote Sensing* 32.3 (1966), pp. 444–462.

[15] Moritz Menze, Christian Heipke, and Andreas Geiger. “Joint 3D Estimation of Vehicles and Scene Flow”. In: *ISPRS Workshop on Image Sequence Analysis (ISA)*. 2015.

[16] Moritz Menze, Christian Heipke, and Andreas Geiger. "Object Scene Flow". In: *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)* (2018).

[17] Xiaoyan Hu and Philippos Mordohai. "A Quantitative Evaluation of Confidence Measures for Stereo Vision". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2121–2133.