# Visual Localization, Mapping and Reconstruction Using Edges

Vom Promotionsausschuss der
Technischen Universität Hamburg-Harburg
zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

von

Jonas Witt

aus Hamburg

2016

1. Gutachter: Prof. Dr.-Ing. Uwe Weltin
2. Gutachter: Prof. Dr.-Ing. Rolf-Rainer Grigat

Tag der mündlichen Prüfung: 10. Juli 2015

# ABSTRACT

Visual navigation is one of the fundamental problems in robotics. The last decade specifically has seen many important contributions in this field. As of today, feature point based approaches are by far the most popular. While successful in a host of applications, untextured environments can be highly problematic for these methods, since the number of reliable feature points is often low in these scenarios. Nevertheless, edges may still be abundantly available, however, remain unused. In this dissertation, we propose complementary edge-based methods for visual localization, mapping and dense reconstruction that can still operate in theoretically minimal scene configurations.

Starting from sparse stereo edge matching, we propose two techniques with different performance/efficiency trade-offs that are both targeted at real-time operation. Besides a comparison to popular dense stereo techniques, we also compare the algorithms to our efficient adaptation of a line segment based stereo approach.

Moving on to stereo visual odometry, we propose a line segment based reprojection optimization that is able to prevail in untextured environments where a proven state-of-the-art feature point based method fails. We argue that our approach can even cope with the theoretically minimal case, consisting merely of two nonparallel line segments. We then extend this approach to a full line segment based simultaneous localization and mapping solution. Using bundle adjustment we are able to build consistent line segment maps that have a high geometric expressiveness with respect to the underlying scene geometry. Especially our long line segment tracks are notable. These are made possible by being completely independent of photometric influences, and additionally our line segment end point estimation approach. We show that we are even able to close loops with viewpoint changes of 180°.

Finally, based on our line segment maps, we propose an efficient method for dense surface reconstruction. Without using restricting assumptions about the scene geometry, we show real-time suitable processing times that make our reconstruction approach highly applicable to robotic exploration use cases in structured environments.

# Acknowledgements

First of all, I would like to thank my supervisor Uwe Weltin for his continued trust and the scientific freedom that I enjoyed during my studies. I learned a lot during the time at his institute and am very grateful that he gave me the opportunity to pursue my research interests.

I also would like to thank the Nordmetall federation of mechanical and electrical industry for sponsoring my doctoral studies and all the colleagues and students I worked with during my time at the Hamburg University of Technology (TUHH) for making it an enjoyable journey. My special thanks go to my collaborators Gerhard Mentges, Ole Falkenberg, Ulf Pilz and Herbert Werner. Especially Gerhard Mentges contributed significantly to this thesis. Sincere thanks go to all of my resilient proof readers Elmar Mair, Jeff Johnson and Shilpa Gulati.

Finally, I wish to thank my mother Jutta and sister Viola for their support and encouragement. I feel especially indebted to my father Lothar who introduced me to electronics and engineering already in my early childhood and took a great part in starting my fascination for robotics. I also want to express my deep gratitude to my wife Miriam for her patience, love and support.

# Contents

# Chapter 1

# Introduction

How can a robot visually find its way through the world and back home? This is one of the core questions in robotics and it has received considerable research attention in the last decades. In the pursuit of an answer, several subproblems have to be addressed. One is the recovery of relative motion based on visual cues. This is often referred to as visual odometry and yields incremental motion estimations between consecutive images. What visual cues to consider, how to detect them efficiently, and how to reliably find them again are important details in this respect. However, in order to answer the original question, we also need to be able to remember or "map" the places where we have been and recognize familiar ones once we return. The place recognition problem is fundamentally similar to the visual odometry one, in that we seek to find correspondences between two different views and eventually infer their relative pose. Nevertheless, the practical challenges are quite different. While visual odometry addresses the case of small motion between two consecutive, and thus visually similar images with high frequency, place recognition has to be able to tell whether a view can be related to *any* of the previously captured ones. Accordingly, the spatial and temporal disparity can be significant. Finally, in order to achieve true autonomy, a robot needs to be able to perceive its surroundings to a level that allows it to reason about where to go next. In the most basic form, this only requires a local reconstruction of traversable space. Nonetheless, the more information can be extracted, the more intelligent a robot can act.

As complex as this is for robots, visual perception of the environment is a capability that humans are usually not actively aware of. While it comes to us

naturally, the deduction of useful information from the raw input stimulus of our eyes is non-trivial. It is the unstructured nature of visual input that requires our brain to devote large sections to give meaning to an observation (Zeki et al., 1991). Mimicking the structure of the visual cortex, many computer vision algorithms similarly strive to make useful information more explicit by first applying low-level filters to acquire features which are then processed by higher level stages. However, since no universally adequate approach exists, the feature extraction stage is highly application dependent.

Von der Heydt et al. (2000) have found that edges – particularly structural ones – play an important role in early primate vision. And indeed, humans can easily deduce the three-dimensional structure even from just the contours of textureless objects and environments. In contrast, most computer vision algorithms struggle with these cases or make restrictive assumptions about the geometry. For example, one very popular simplifying assumption is dubbed "Manhattan world" and restricts all scene geometry to be perfectly orthogonal to each other. Obviously, such assumptions limit the scope of respective methods to environments where this is actually the case. This thesis strives to find generally applicable solutions in the context of robotic environment perception that are able to cope with extreme scene sparsity (absence of texture and few geometrical features) by utilizing edges as features. More specifically, new techniques for stereo edge matching, edge-based localization, mapping and dense reconstruction are proposed.

## 1.1 Why Edges?

Edges surround us in our everyday life. Whether they form the silhouette of a chair, the road markings on our way to work or the complex texture in the bark of a tree, the word *edge* will be used to describe all perceptual transitions between image regions of different brightness. Consequently, this informal definition implies that such edges do not necessarily correspond to structural edges. Nevertheless, structural edges often give rise to brightness changes which are perceivable as edges by the above definition. Figure 1.1 depicts examples of several different types of edges.

In a digital image, the information is encoded in a regular pixel grid. Edges


Figure 1.1: All marked areas show different edge types. 1. contour or structural edge, 2. edges on a planar surface, 3. edges as a result of a mixture of complex texture and three-dimensional structure, 4. soft edges that describe a smooth brightness or even color transition.

can be detected by finding the largest differences between neighboring pixels. Mathematically, this operation corresponds to the search for local maxima in the first derivative (spatial gradient) of an image (see the seminal works by Canny (1986) and Marr and Hildreth (1980)). As homogeneous regions have small derivatives, it is obvious that the gradient reveals locations of interest. Accordingly, our information about the observed scene has already become more explicit than in the original pixel representation. This becomes visually apparent if one compares the dense pixel grid in the left image of Figure 1.2 to its edge point representation on the right. However, it is not obvious whether we can retain the full information content with this transformation. This was thoroughly investigated by Elder (1999). He showed that from edge representations that include location, intensity change, direction and blur/scale, one can almost flawlessly restore the original pixel information. Yet, this requires the costly computation of all edge scales, which is often omitted in real-time applications (also see (Lindeberg, 1998) for an edge scale disquisition). Instead, small and efficient edge filters that only detect sharp intensity changes are used most often. This approach loses some lower frequency information like shading or soft shadows for the benefit of computational efficiency. However, contour edges are never soft and thus always appear

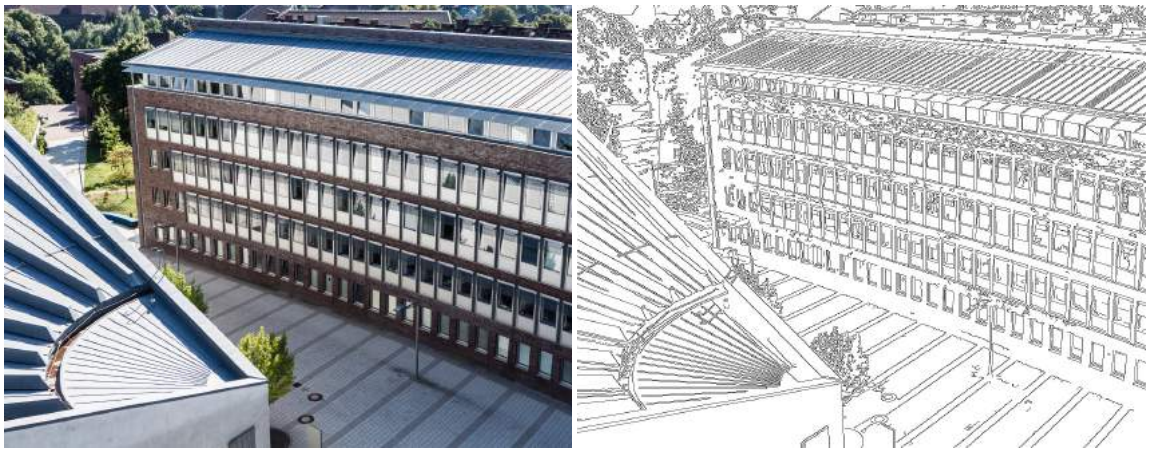


Figure 1.2: Edges capture important image information and make it explicit. If all edge scales are incorporated, Elder (1999) even showed that edges can be a complete representation of an image from which the original pixel information can be restored. Photograph: Lina P. A. Nguyen.[1]

at the lowest edge scale. Accordingly, this incomplete representation can be seen as a natural reduction of the original image to the potentially structurally relevant information.

## 1.2 Simultaneous Localization and Mapping

The ability to navigate is a core functionality in many robotic applications. Be it a state-of-the-art vacuum cleaning robot, an autonomous drone exploring disaster zones, a delivery robot or an automated car: navigation capabilities are required in an increasing number of applications. For this reason, navigation has received strong research interest and is currently being commercialized in various ways.

The most common navigation problem is to keep track of the robot pose in an initially unknown world. Without simplifying assumptions, this problem can be arbitrarily complex. Ideally, a system should be able to recognize features and places independent of environmental influences. Dramatic changes in appearance due to lighting, partial scene reconfiguration or even seasonal changes in the environment are challenging in this respect. In addition, a system must be able to cope with dynamic objects in its field of view and work equally well in textured and untextured environments. Under all these conditions, we want to

[1] `http://www.lina-nguyen.de`

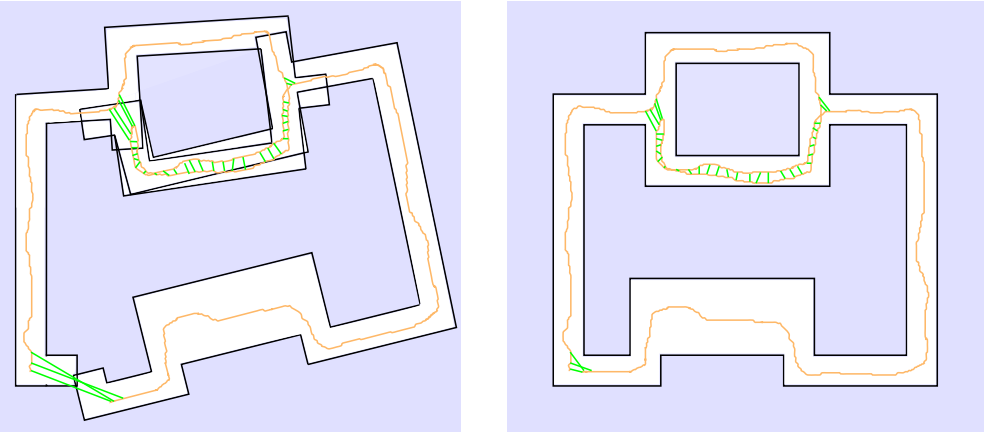


Figure 1.3: Left: 2D SLAM problem solution before optimizing with loop-closure constraints (straight green lines). The reconstructed robot trajectory (orange) as well as the reconstructed map (black outline) suffers from large accumulated errors. Right: the added loop-closures allow to eliminate large portions of the accumulated errors by introducing associations between initially "far away" locations.

be able to localize with respect to the map that we build and update at the same time. However, to the knowledge of the author, such a complete general purpose solution has not been proposed yet.

The underlying state estimation problem is called Simultaneous Localization And Mapping (SLAM). This name already suggests the chicken-and-egg problem that arises by the need to localize with respect to a map while we do not yet have a map. Hence, the map geometry and robot location are strongly interdependent since the map is built from the robot's observations. This becomes especially apparent when a so-called loop-closure is found during exploration. A loop-closure is the recognition of a place that was already mapped. Figure 1.3 shows a 2D example of the trajectory and map before the addition of loop-closure constraints to the optimization problem and after. The error (and uncertainty) that is naturally accumulated over time during exploration (usually proportional to the travel distance) can be significantly reduced by successful loop-closures. This is due to the graph nature of the navigation problem that is exploited. Loop-closures add short-cuts in the graph to a certain robot pose. Accordingly, the travel distance along the graph to that point decreases instantaneously as the loop-closure constraint is introduced – and so does the uncertainty of that location.

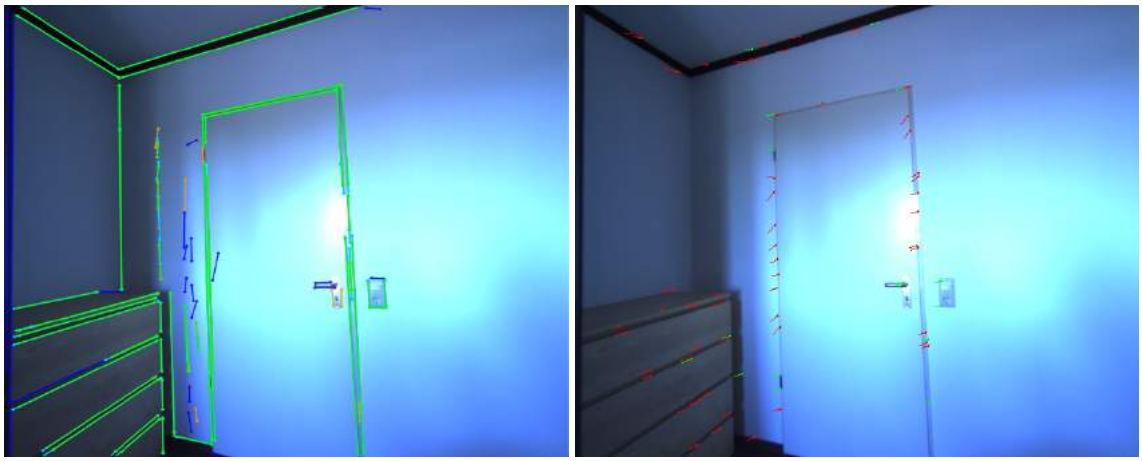


Figure 1.4: These images show successfully tracked edges (green lines in left image) in comparison to feature points (green point trajectories in right image). The restriction to corners and blob-like features discards large portions of useful information. This can lead to erratic motion estimates in sparsely textured scenes.

### 1.2.1 Passive vs. Active Sensing

Besides passive sensors like cameras, laser scanners are very popular for the SLAM problem. These sensors directly output 3D points by measuring the time that it takes a projected laser beam to travel to an object and back. While the data is precise and conveniently acquired in real-time, in comparison to camera systems, laser scanners are more expensive and bulky. Furthermore, the active sensing principle with laser beams has two immanent drawbacks: active projection costs energy (increasing with the desired range) and can interfere with other sensors of the same principle. Also the low price of consumer cameras (especially when compared to laser scanners) and their readily availability has stimulated intense research in the field of camera-based SLAM in the last decade.

### 1.2.2 Edges vs. Points

For camera based SLAM systems, localization based on feature points has been most popular (Davison et al., 2007; Fitzgibbon and Zisserman, 1998; Klein and Murray, 2007; Konolige and Agrawal, 2008; Nistér et al., 2004; Sibley et al., 2010). Feature points can be found in numerous different ways, e.g. by computing Harris corners (Harris and Stephens, 1988), SIFT keypoints (Lowe, 2004), or AGAST/ FAST corners (Mair et al., 2010; Rosten and Drummond, 2006). All methods for

feature point detection have in common that they search for distinctive locations in the image that are constrained in x and y coordinates – i.e. the center of blob or corner-like pixel neighborhoods. In contrast, edge points of straight or slightly curved edges are not suitable as feature points since the location along the edge would be ambiguous. Accordingly, when edges are considered as features, an algorithm needs to take into account that the edge feature location is only well-constrained in gradient direction and not perpendicular to it. This spatial locatability trait is an important distinction between feature points and edges and hence complicates the successful utilization of edge information in algorithms. Additionally, the start and end point of an edge have to be assumed to be unreliable.

However, as can be seen in Figure 1.4, untextured scenes have a very limited number of good feature points whereas edges can still be abundantly available. This is the main motivation to introduce edges into the SLAM domain as a complementary information source that is completely self-sufficient and able to handle the extremely visually sparse corner cases that are especially common in indoor settings.

## 1.3 Outline of the Dissertation

This thesis is laid out to guide the reader from stereo edge and line segment matching over line-based visual odometry to a full line SLAM system and finally dense surface reconstruction from line segment maps.

First, we introduce our notation and preliminaries in Chapter 2. We cover the basics of stereo geometry and other concepts we will depend on in the following chapters. We also introduce our custom stereo camera that was used to capture all sequences that we evaluate our algorithms on.

Chapter 3 proposes and evaluates three sparse stereo matching methods for small baselines with known epipolar geometry (i.e. for stereo cameras). We propose algorithms for local (winner-takes-all – no global consistency) and semiglobal (globally consistent per edge chain) stereo edge matching and find that the error rate outperforms a competing sparse algorithm and is often even superior to computationally expensive state-of-the-art dense algorithms. We also describe an efficient stereo line segment matching algorithm that uses dynamic programming for global matching consistency and compare it to the edge-based stereo

matchers in the setting of indoor environments.

In Chapter 4, we propose a novel line-based stereo visual odometry algorithm. Using the stereo line-segment reconstruction from Chapter 3, the reprojection error is used as an optimization criterion to yield the motion between the previous and the current stereo frame. For the correspondence problem, a novel line segment matching algorithm is presented that allows to account for unreliable line segment end points using multiple matches. The incremental frame-to-frame motion estimates are concatenated to a trajectory which provides a linearly drifting localization. We show that our approach outperforms a state-of-the-art feature point-based visual odometry algorithm in several indoor sequences and point out the fundamental merits of our line-based approach in extremely sparse corner cases.

The complete Line SLAM system is introduced in Chapter 5. The inter-frame line segment matches and initial trajectory from the described visual odometry approach are used to formulate a line bundle adjustment problem to find the globally optimal solution for structure (i.e. the line segment map) and motion (camera poses). We discuss the line parameterization in the bundle adjustment problem and derive a novel minimal line representation with favorable properties. We also introduce techniques for estimating and updating line segment end points. Finally, we demonstrate the potential for loop closures under extreme viewpoint changes and showcase consistent line segment maps that already exhibit a high geometric expressiveness in their raw form.

Subsequently, in Chapter 6, we propose a novel surface reconstruction algorithm that exploits the richness of the line segment maps to efficiently infer planar surfaces. A pessimistic free space assumption makes this algorithm suitable for robotic exploration applications. Since we do not make restrictive assumptions about any geometry, we are able to reconstruct slanted and in moderately textured cases even curved surfaces.

Finally, Chapter 7 concludes the thesis by summarizing our results and findings. Furthermore, we discuss possibilities for future work.

For the reader interested in the robotic hardware that was developed to run the presented algorithms, Appendix A describes a novel quad-rotor research platform that was jointly developed at the TU Hamburg-Harburg and is designed to run computer vision algorithms, sensor fusion and control onboard in real-time.

## 1.4 Key Contributions

The key contributions of this thesis are centered around novel approaches for visual SLAM and dense reconstruction capable of handling even the theoretically minimal cases in scene sparsity. In particular, we propose:

1. two stereo edge matching algorithms based on local and semiglobal consistency approaches respectively, and a variation on a previously published stereo line segment matching algorithm (Chapter 3),

2. a method to reconstruct stereo camera motion by finding line segment correspondences (Chapter 4),

3. a complete line SLAM system using bundle adjustment to estimate the camera motion and 3D line segment map (Chapter 5), and

4. an approach for efficient dense surface reconstruction from line segment maps that is applicable to real-time robotic exploration (Chapter 6).

# Chapter 2

# Preliminaries and Notation

In this chapter, we introduce the notation and concepts that we will build upon in the following chapters. Specifically, we briefly outline the basics of stereo geometry and matching, as well as introducing the conventions that we adopt. We then continue with the technical details of our custom built stereo camera and finally cover several well-known algorithms for the sake of self-sufficiency of this thesis.

## 2.1 Notation

The mathematical notation that is used throughout this thesis is summarized in the following.

| Symbol | Meaning |
|---|---|
| $a, \lambda, x, \ldots$ | Scalar values |
| $\mathbf{p}, \mathbf{x}, \ldots$ | Vectors |
| $p_x, p_y, p_z$ | Scalar vector components |
| $\mathsf{R}, \mathsf{W}, \ldots$ | Matrices |
| $\mathsf{R}^\top, \mathbf{x}^\top$ | Transpose of a matrix or vector |
| $\|\mathbf{x}\|$ | $L_2$ norm of a vector |
| $\|\mathbf{l}\|$ | Length of a line segment |
| $\|z\|$ | Absolute value of a scalar |
| $S, L, F, \ldots$ | Objects and sets of objects with properties and relations to each other |
| $\|S\|$ | Number of elements in a set |

## 2.2 Camera Model

A camera projects the three-dimensional world to its image plane. While the actual physics of photons passing an array of lenses and finally reaching the digital imager are quite involved, it is common in computer vision to use a simplified model called the pinhole camera. Using this model, all viewing rays pass through the same point – the pinhole. This viewing ray intersection point is where the camera center is placed, also called the optical center. Figure 2.1 visualizes the corresponding geometry and coordinate system conventions. Note, that we assume the image plane to be in front of the optical center for simplicity, while physical (approximate) pinhole cameras would observe a mirrored projection on their image plane behind the optical center.

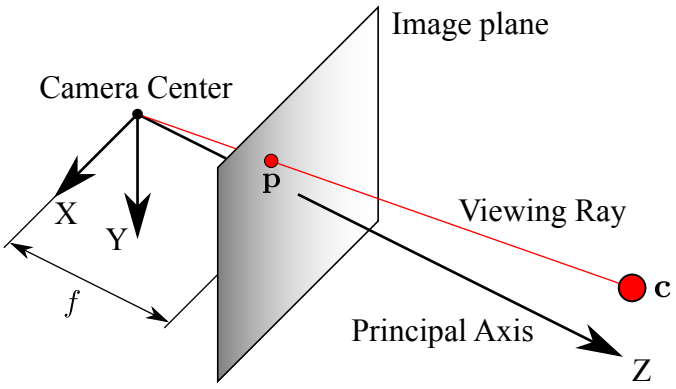



Figure 2.1: Pinhole camera model observing the 3D point **c** on its image plane as the projection **p**. Any point on the drawn viewing ray yields the same projection.

The principal axis is defined by being perpendicular to the image plane and passing through the optical center. The intersection of the principal axis with the image plane is called *principal point* and is usually located approximately in the center of the image. Since lens distortion is not modeled by the pinhole model, we have to estimate the distortion parameters of our physical camera by calibration. These distortion parameters can then be used to undistort the original image so that the pinhole assumptions hold. We used the OpenCV library for our camera calibration (Bradski, 2000). Besides the lens distortion, the *focal length* $f$ and the principal point $\mathbf{o} = (o_x, o_y)^\top$ are calibrated. These camera properties are called *intrinsic parameters*. Figure 2.2 shows the relation between the image extents and the optical center. Throughout this work we assume that image space coordinates are with respect to the optical center. The transformation to a coordinate system

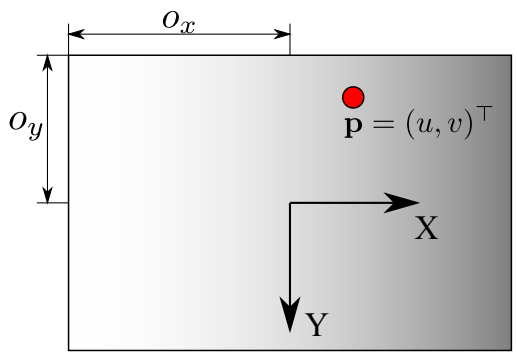



Figure 2.2: Image space with point projection **p**. The relation between the image space coordinates $u, v$ and the top left image corner is defined by the principal point $o_x, o_y$.

that has its origin at the top left image corner can be computed by adding the principal point coordinates. With this, the projection **p** of a point **c** is given by

$$\mathbf{p} = \begin{pmatrix} u \\ v \end{pmatrix} = \left( \frac{fc_x}{c_z}, \frac{fc_y}{c_z} \right)^\top . \tag{2.1}$$

Note that **c** is in the camera coordinate frame that is shown in Figure 2.1. If a point **w** is in world coordinates, we first have to transform it given the camera rotation matrix R and translation **t** that describe the camera pose in the world

$$\mathbf{c} = \mathsf{R}^\top (\mathbf{w} - \mathbf{t}). \tag{2.2}$$

Inspecting Equation (2.1), we can see that only the ratio between the x- and y- to the z-coordinate of **c** determine where a point is projected on the image plane. Accordingly, only knowing the projection **p**, we cannot infer at what depth on the viewing ray the 3D point actually lies.

## 2.3 Stereo Geometry

However, when we introduce a second camera with a known relative pose to our first camera, we can triangulate the depth from the two matching projections. Figure 2.3 visualizes the geometrical setup. For such a stereo setup it is most common to have a purely horizontal displacement between both cameras. Any small physical deviations from this can be corrected for by rectifying both images.

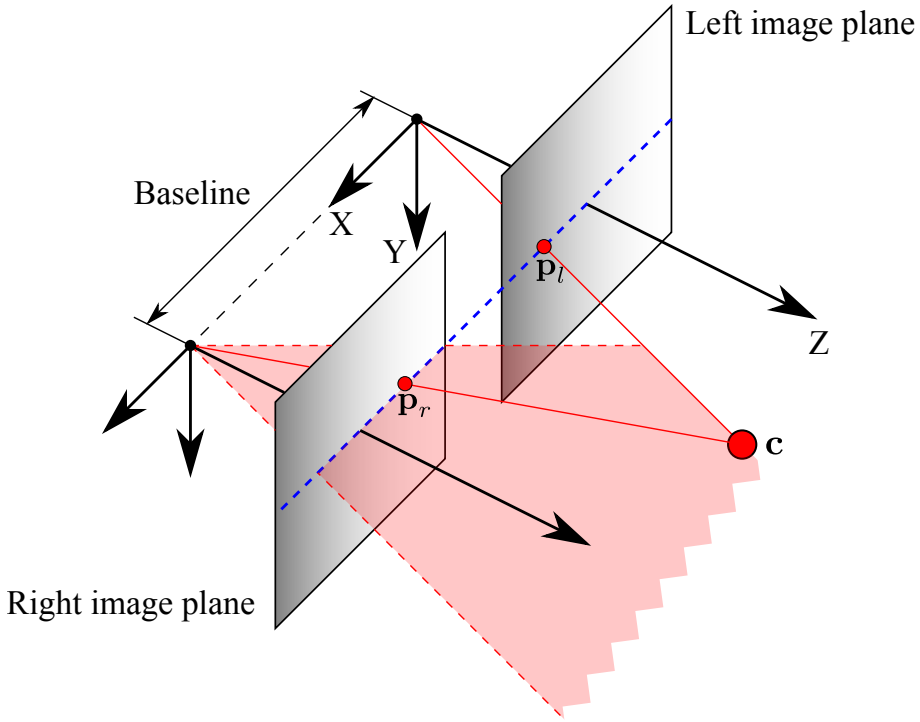



Figure 2.3: Stereo geometry with a fronto-parallel camera setup. Rectification allows to compensate for lens distortion and small deviations from a perfect single-axis camera displacement. For a rectified image pair, correspondences always lie on the same scanline (dashed blue). With a known baseline, $\mathbf{c}$ can be triangulated from the corresponding projections $\mathbf{p}_l$ and $\mathbf{p}_r$.

Rectification warps stereo images so that the resulting ones are perfectly fronto-parallel and share the same y-coordinate for each pixel row (or scanline). Hence, also correction of lens distortions is included. Accordingly, a projection $\mathbf{p}_l$ in the left image can only have its right image correspondence $\mathbf{p}_r$ on the same scanline. Throughout this thesis we will always assume rectified image pairs.

Using the matched image points $\mathbf{p}_l = (u_l, v_l)^\top$ and $\mathbf{p}_r = (u_r, v_r)^\top$, we can reconstruct $\mathbf{c}$. As we discussed, the vertical coordinate $v = v_l = v_r$ is the same for both image points. The reconstruction in camera coordinates is obtained by

$$\mathbf{c} = \left( \frac{bu_l}{u_l - u_r}, \frac{bv}{u_l - u_r}, \frac{bf}{u_l - u_r} \right)^\top \tag{2.3}$$

with known intrinsic stereo camera parameters. The baseline is denoted with $b$ and $f$ corresponds to the focal length. The difference in horizontal point coordi-

Figure 2.4: The custom stereo camera that was used for the acquisition of all sequences in this thesis. The baseline is 16cm and the field-of-view is $100^\circ \times 80^\circ$.

nates $u_l - u_r$ is the disparity $d$, corresponding to depth. In our setup a disparity of $d = 0$ indicates a point with infinite distance. World point coordinates $\mathbf{w}$ are computed with the camera rotation $\mathsf{R}$ and translation $\mathbf{t}$:

$$\mathbf{w} = \mathsf{R}\mathbf{c} + \mathbf{t}. \tag{2.4}$$

We choose the left camera frame as the reference for all coordinate transformations. If one is interested in a more comprehensive discussion of multiple view geometry, Hartley and Zisserman (2004) provide an excellent reference.

## 2.4 Stereo Camera Hardware

We built a custom stereo camera from two IDS UI-1241LE-C-HQ industrial color cameras. These have a native resolution of $1280 \times 1024$px and global shutter to avoid motion induced distortions. To achieve a lightweight and rigid frame con-

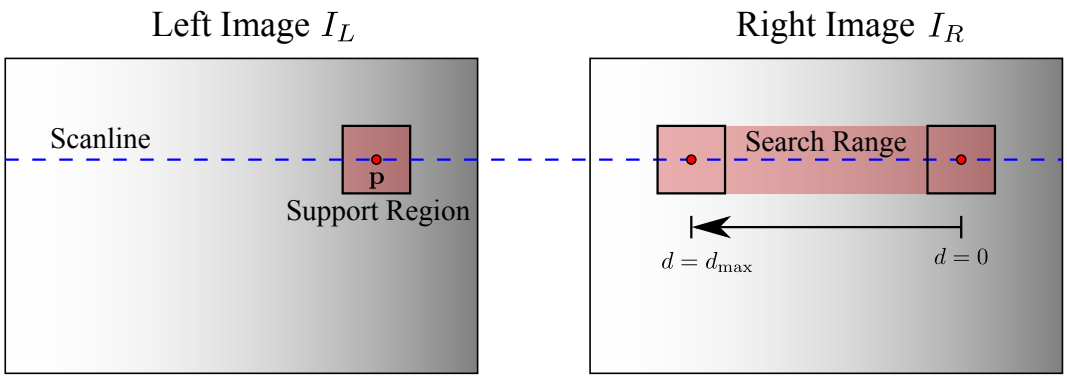



Figure 2.5: Stereo matching on rectified images only considers matches on the same scanline. A support region at **p** in the left image is compared to candidate support regions in the right image that are displaced by the sought-after disparity $d \in [0, d_{\max}]$.

struction, we used carbon fiber-reinforced polymer. Figure 2.4 shows the setup. The wide-angle lenses with a field-of-view of $100° \times 80°$ provide adequate visual overlap between consecutive frames of moderately fast rotations. For hand-held applications or other highly dynamic motions this has a significant impact on the achievable tracking robustness since more features are shared between observations. We used this stereo camera to record all sequences that are shown in this thesis. The native resolution was used as input to image rectification to preserve sharpness when correcting strong lens distortions. The rectified resolution that all sequences share is $640 \times 512$px. The practical synchronized framerate that we achieved was about 10Hz. As synchronization mechanism, the flash output of the left and trigger input of the right camera were connected.

## 2.5 Stereo Matching

To find correspondences between the left and right image for stereo reconstruction, we need to perform stereo matching. For this, we need to find the best match for a location **p** in the left image from a range of possible matches in the right image. Having rectified both images, the search space is restricted to the scanline of **p**. The horizontal pixel displacement between **p** and the matching location in the right image is called disparity and here denoted by $d$. When we choose a maximum disparity $d_{\max}$ to reduce the search space, we disregard all matches with a distance smaller than $\frac{bf}{d_{\max}}$. Figure 2.5 visualizes the stereo matching problem.

Now suppose we want to use the mean squared error (MSE) as matching cost to determine support region similarity, we do this element-wise for the support region of $\mathbf{p}$ and the matching candidate with disparity $d$

$$m(\mathbf{p}, d) = \frac{1}{|\Gamma|} \sum_{\mathbf{s} \in \Gamma} \left( I_L \left( \mathbf{p} + \mathbf{s} \right) - I_R \left( \mathbf{p} + \mathbf{s} - (d, 0)^\top \right) \right)^2. \tag{2.5}$$

Here, $\Gamma$ is the set of all 2D pixel displacements from $\mathbf{p}$ to cover the whole support region. Accordingly, the pixel-by-pixel squared difference around $\mathbf{p}$ and the candidate location $(p_x - d, p_y)^\top$ is computed.

To find the best match from the candidate range, the most simple approach is called winner-takes-all (WTA). It merely picks the candidate with the lowest matching cost. While being fast, this usually yields quite noisy results since neighborhood consistency is not considered. However, the pixels from the same 3D surface do have similar disparities which qualifies for using this information in a cost function to resolve ambiguities. WTA is also often referred to as a local technique, because it does not take into account global consistency – every match is computed completely independently. In contrast, global techniques utilize consistency information to find better solutions in the face of ambiguity.

## 2.6 Iterative Closest Point Algorithm

The iterative closest point (ICP) algorithm tries to minimize the distance between a reference set and a source set of points (Besl and McKay, 1992). The output is the iteratively optimized transformation to align both point sets. Basically, the ICP algorithm works as follows:

1. For each point in the source set, find the closest point in the reference set.
2. Based on the matches from the previous step, compute the gradient of the mean squared distance error with respect to the sought-after transformation. Compute a delta transform from the gradient.
3. Transform the source point set by the found delta transformation.
4. If the transformation has converged, exit. Otherwise go to 1.

While the most common ICP variant is 3D-3D point cloud alignment, also 3D-2D reprojection variants have been proposed. These minimize the distance between a 3D point set that is projected to an image plane and a 2D point set on the same plane. In this case, the Euclidean distances in image space are minimized. Since we do a reprojection to a camera, the transformation that is optimized is the pose of that camera with respect to the 3D point set. Since ICP is essentially founded on a gradient-based optimization it can suffer from local minima if the initialization is not close enough to the global optimum.

## 2.7 RANSAC

Random sample consensus (RANSAC) is a robust parameter estimation method using a hypothesize-and-test approach (Fischler and Bolles, 1981). Especially for data sets with significant outlier percentages this method is suitable. To find the sought-after parameters, it draws $N$ random samples from the data set. From each sample, a hypothesis about the solution parameters is computed and tested for consensus with the rest of the data set. Based on a criterion, the data set is then split into inliers that support the given hypothesis and outliers that do not. The hypothesis with the largest support group is chosen as the solution. Often, an iterative optimization is subsequently applied to refine the inlier fit as the outliers have now been identified.

# Chapter 3

# Stereo Edge Matching

This chapter investigates sparse stereo matching techniques for small baseline applications with known epipolar geometry. It is the goal to find correspondences of intensity edge locations between two rectified images, which in general are the left and right image of a calibrated stereo camera. With Edge Matching by Confidence-Based Refinement (EMCBR) and Edge-Based Search Using Dynamic Programming (EBDP), two edge pixel based techniques are proposed. For the special case of straight intensity edges, a line segment based method is investigated. This chapter is based on the corresponding publications for EMCBR (Witt and Weltin, 2012a), EBDP (Witt and Weltin, 2012b) and the section about stereo line matching in (Witt and Weltin, 2013).

## 3.1 Introduction

Dense stereo correspondence algorithms have been thoroughly studied in the last decades. Many different approaches with individual performance characteristics exist (Scharstein and Szeliski, 2002). Yet, for robotic applications like object detection and navigation, dense depth information is often not required. For typical robotic tasks like localization, mapping or object detection, point-based methods currently are most common. For this, corner and blob-like structures are detected and matched. However, in sparsely textured scenes, such point-features may not be available in sufficient numbers. Edge-based systems can fill the gap as shown e.g. by Tomono (2009b) and Chandraker et al. (2009) with their SLAM and visual odometry systems. In (Helmer and Lowe, 2010) an object detection system

was presented, which uses only 2D edges with depth information as a scale prior. A more sophisticated method that matches 3D object silhouettes to stereo edge segments was investigated by Sumi et al. (2002). An application of accurate long range stereo line matching for road markings was presented by Nedevschi et al. (2006). This work demonstrates the potential for high precision stereo since edges can easily be detected at subpixel accuracy. Further refinement is performed by incorporating adjacent edge pixels into higher level primitives (in this case line segments) which can yield a significant noise reduction.

Matching edge-segments across two views poses different challenges than dense matching. Many edges lie on object borders, which can be a problem for correlation based algorithms if the matching support region is not carefully chosen. For example, symmetric block-like support regions will significantly overlap the background at object contours which can lead to poor matching results. Additionally, horizontal edge segments are particularly difficult due to inherent ambiguities. Also, since the matching is only sparse, one cannot gain confidence in disparities over larger surfaces. Finally, corresponding edge pixels are not necessarily detected in both images and edge connectivity may not be preserved. On the other hand the search space is significantly simplified due to the restriction of disparities to edge loci, resulting in less computational effort.

## 3.2 Related Work

Previous papers present very different approaches to the problem of matching edges in two or three views. Algorithms for straight lines have been proposed by (Medioni and Nevatia, 1985), (Li, 1994) and (Ayache and Faverjon, 1987), the latter of which was also extended to parametric curves in (Robert and Faugeras, 1991). The seminal work by Baker (1982) investigates many aspects of edge-based stereo, using correlation and connectivity constraints. Kim and Bovik (1988) search for high-information points on edge contours to guide the matching of the remaining points. Similarly, Deriche and Faugeras (1990) propose to use distinctive high-curvature points along edges to find correspondences (by assuming a figural continuity constraint). Kawai et al. (1998) create a so-called boundary representation for the calibrated images and subsequently match these by incorporating intensity and angle information. A correlation-based method utilizing

color information was presented by Koschan and Rodehorst (1995), while a more recent publication proposes a phase-based algorithm utilizing multi-scale Gabor filters with a probabilistic model for matching (Ulusoy et al., 2004).

Ohta and Kanade (1985) use dynamic programming to find the minimum match cost for edge-delimited intervals along scanlines. The addition of another search dimension allows for enforcing inter scanline consistency at the expense of computational resources. However, especially when expensive sparse algorithms are considered, dense stereo methods should also be taken into account (Scharstein and Szeliski, 2002). To acquire the edge matching information in this case, one can sample a dense disparity at edge locations and optionally use the connectivity information to refine the initial matches. For simple dense techniques, the resulting processing speed can be competitive to the previously mentioned methods.

## 3.3 Outline

The first two techniques that are proposed in Section 3.4 are general in that arbitrarily shaped edge contours can be matched. They are both based on correlation-based matching and use a similar refinement step but differ in the way that matches are generated. Edge Matching by Confidence-based Refinement (EMCBR, Section 3.4.3) uses winner-takes-all matching and subsequent refinement which propagates confidence along edge contours. Special care is taken to ensure that matching support regions are suitable for the case of 3D object contour edges (i.e. when a standard symmetric support region would equally overlap a background and foreground object). The propagation of confidence along edges is based on the assumption, that the majority of edge pixels are correctly matched in the first place. This allows finding mismatches and interpolating erroneous and initially unmatched edge sections.

Edge-based Dynamic Programming (EBDP, Section 3.4.4) introduces a cost function that penalizes disparity discontinuities along edge contours and searches for the minimum cost for each edge contour (rather than searching along scanlines). This approach is beneficial for edges that have ambiguous support regions for large sections, which is especially relevant for horizontal edges. The performance of both approaches is compared to another sparse and several dense stereo

algorithms.

Finally, in Section 3.5, we take a look at line segment matching by adapting the method from Li (1994) and comparing it to EMCBR and EBDP for which line segments are extracted. While the underlying algorithms are not directly comparable, for applications that ultimately require line segments instead of pixelwise edge disparities, a performance comparison is interesting.

## 3.4 Stereo Edge Matching

The matching of sparse features like edges involves the same steps as for dense matching, namely preprocessing, matching cost aggregation, match selection/optimization and optionally refinement. For stereo edge matching, the general algorithm structure is visualized in Figure 3.1. We will first briefly introduce the utilized matching cost and then investigate cost aggregation for the case of edges. EMCBR and EBDP both use the same preprocessing and matching cost aggregation principles but do differ in the match selection/optimization step. Finally, confidence-based refinement that is first introduced for EMCBR in Section 3.4.3 is also employed in EBDP for invalidation of less confident edges and disparity interpolation where this is possible.

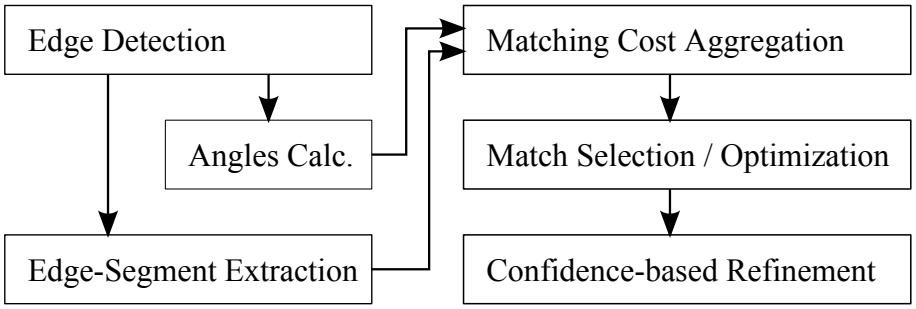



Figure 3.1: General algorithm structure for stereo edge matching.

For the edge detection block we use the Canny detector (Canny, 1986) with subpixel refinement (Devernay, 1995). This way we gain subpixel accurate disparities for vertical and diagonal edges with little effort. Matching can still be done at pixel level.

### 3.4.1 Matching Cost

Many different matching cost functions, similarity measures and transforms exist for the purpose of stereo matching (Scharstein and Szeliski, 2002). In Section 2.5, we provide a brief introduction to the topic. Common matching costs for real-time stereo matching are the squared intensity difference and the absolute intensity difference. Both measures can be truncated to improve robustness in the face of outliers, see (Gong et al., 2007) for a thorough evaluation. Several tests on the Middlebury stereo sets and with a stereo camera in an office environment resulted in the truncated sum of absolute differences (SAD) being selected as the measure of choice in our case. Additionally we subtract the mean intensity difference $\mu(\mathbf{p}, d)$ to improve matching of mildly shiny objects with specular reflections and also to cope with different camera sensor sensitivities. We also truncate this value at $t_\mu = 10$ to not match uniform surfaces with arbitrary intensity differences. The matching cost $m(\mathbf{p}, d)$ at location $\mathbf{p}$ and the candidate with disparity $d$ using the support region $\Gamma$ accordingly is

$$\mu(\mathbf{p}, d) = \max\left(\min\left(\frac{1}{|\Gamma|}\sum_{\mathbf{s}\in\Gamma} I_L(\mathbf{p}+\mathbf{s}) - I_R\left(\mathbf{p}+\mathbf{s}-(d,0)^\top\right), t_\mu\right), -t_\mu\right) \quad (3.1)$$

$$m(\mathbf{p}, d) = \frac{1}{|\Gamma|}\sum_{\mathbf{s}\in\Gamma} \min\left(I_L\left(\mathbf{p}+\mathbf{s}\right) - I_R\left(\mathbf{p}+\mathbf{s}-(d,0)^\top\right) - \mu(\mathbf{p}, d),\ t_{\text{trunc}}\right). \quad (3.2)$$

The term $\frac{1}{|\Gamma|}$ normalizes the matching score by dividing by the number of pixels in the support region $\Gamma$. $I_L(\mathbf{p})$ and $I_R(\mathbf{p})$ are the pixel intensity values at location $\mathbf{p}$ for the left and right image respectively. The truncation parameter was empirically adjusted and finally set to $t_{\text{trunc}} = 30$. Different matching windows are investigated in the following section.

### 3.4.2 Cost Aggregation

For the aggregation of the matching costs, using simple symmetric support regions as often chosen in real-time dense matching is not useful. This is due to the nature of edges, since they are gradient maxima which divide regions of different intensity. Edges can naturally occur on textured, locally planar surfaces but also at depth discontinuities as a result of overlapping surfaces of different intensity. Accordingly, many edge pixels lie on object borders which have an intensity that

is usually a mixture of both surface intensities. In effect, the edge pixel intensity depends on the subpixel location $\alpha \in [0, 1]$ of the edge and both surface intensities $I_1$ and $I_2$

$$I_{\text{edge}} = (1 - \alpha) I_1 + \alpha I_2 \tag{3.3}$$

which results in an arbitrary value $I_{\text{edge}} \in [I_1, I_2]$ depending on the orientation and position of the camera. Basically this is true for every pixel, but by definition edge pixels mark the locations where this effect has the biggest impact on pixel intensities. Accordingly, edge pixels themselves are not very suitable for including them in an intensity based matching score. Figure 3.2 shows block matching on an edge segment. Here, 20% of the pixels in the support region belong to the edge which can have a significant influence on the overall matching cost. Making the support region larger reduces this effect, but also decreases the ability to match small objects.

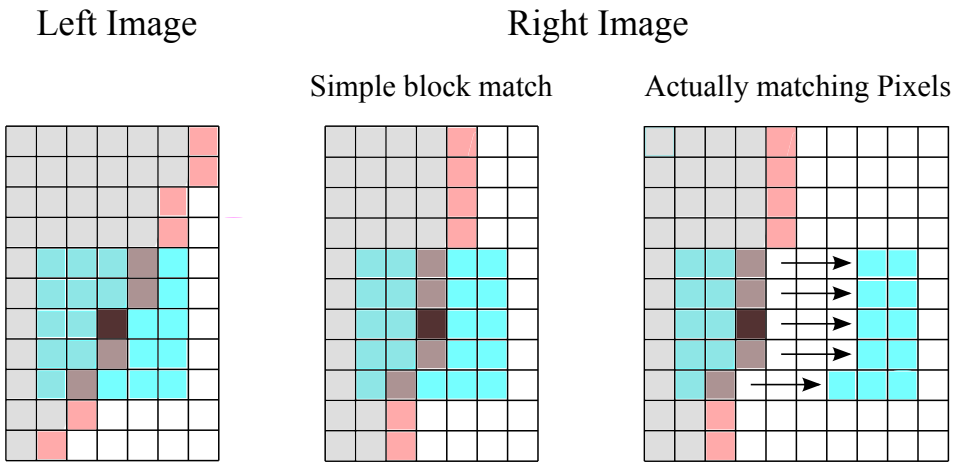



Figure 3.2: Block matching with a 5x5 support region (cyan). The center is marked by a dark pixel. The edge pixels (red) lie on an object contour and separate a foreground object (gray) and the background (white). Since the support region overlaps a depth discontinuity, the truly matching pixels are separated into two groups in the right image – one is shifted to the right, as indicated by the arrows.

The common occurrence of depth discontinuities at edge locations also has to be specifically incorporated into the design of the support region. Consider again the edge depicted in Figure 3.2. If the region to the left of the edge belongs to a foreground object and the region to the right to the background, the actually matching pixels of the background will be shifted by the difference in disparities (which is three in this case). This can be accounted for, as described in

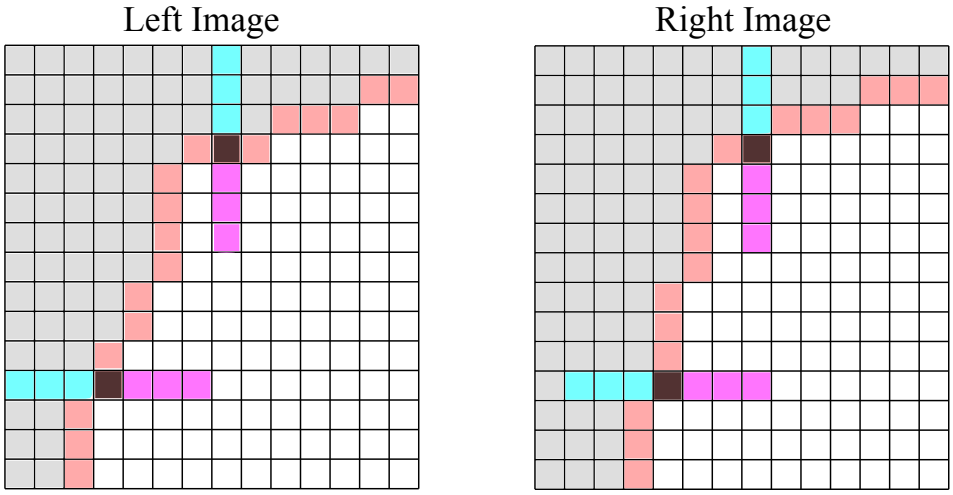



Figure 3.3: Shifted pixel-block matching windows on an edge (red) with vertical as well as horizontal sections. For each examined location (dark pixels), the minimum matching cost of the left/top (cyan) and right/bottom (magenta) 3x1 support windows is taken. On object contours, we yield consistent matches on the foreground object. The edge angle determines whether vertical or horizontal support regions are used.

(Hirschmüller et al., 2002), but at a computational cost. Shiftable filters as evaluated in (Scharstein and Szeliski, 2002) are a more efficient possibility. For the use in edges, they need to be adapted, though. With simple block matching we may end up with less than 50% of the pixels in the support region being suitable matching candidates on object borders.

For these reasons we propose simplified shifted pixel-blocks which do not suffer from any of these problems and introduce almost no computational overhead. These shifted pixel-blocks are matched on either side of a candidate edge pixel, as shown in Figure 3.3. Only edges that differ by no more than $\alpha_{\text{match}}$ in orientation in the left and right image are considered. The actual edge pixel is not included in the pixel-blocks, due to their intrinsic unsuitability for intensity-based matching. Depending on the edge orientation, left and right or top and bottom pixel-blocks are matched. This helps in disambiguating horizontal edge disparities. In either case only the minimum matching cost is taken. If an edge lies on an object border, the foreground disparity is retrieved and the consistency of the support region is preserved. Increasing the width of the support regions (e.g. from one to three or more pixels) makes individual matches more robust when a simple local matching technique like winner-takes-all (WTA) is employed. However, if

neighborhood consistency is incorporated by the matching approach the resulting support region overlap essentially yields no additional information.

### 3.4.3 Edge Matching by Confidence-Based Refinement

In this section, we will introduce a novel refinement algorithm that enforces consistency and smoothness among the disparities of edge segments. In EMCBR, the improvement over the initial WTA matches stems from the fact that many individual edge points are ambiguous, which leads to isolated and unsmooth disparities, if they are matched independently. The discriminative power of a whole edge segment in contrast is much higher. However, since common edge detectors do not yield perfect edges and frequently cross object borders or produce other "glitches" it is not trivial to take full advantage of the connectivity information.

In order to refine our initial disparities we first need to rank the reliability of the found matches. We do this with the ratio of the best match $m_{1\text{st}}(\mathbf{p})$ and the second best match $m_{2\text{nd}}(\mathbf{p})$ at a location $\mathbf{p}$:

$$C(\mathbf{p}) = \begin{cases} 3, & \text{if } m_{2\text{nd}}(\mathbf{p})/m_{1\text{st}}(\mathbf{p}) > 2 \\ 2, & \text{if } m_{2\text{nd}}(\mathbf{p})/m_{1\text{st}}(\mathbf{p}) > 1.5 \\ 1, & m_{1\text{st}}(\mathbf{p}) < t_{\text{match}} \\ 0, & \text{no match found} \end{cases} \tag{3.4}$$

Empirically, we found that if the second best match has more than a doubled matching score, our best match is probably the right one. We reward this with the highest confidence. A confidence value of one is usually assigned to ambiguous matches like horizontal edges or repetitive patterns. If the best matching cost is larger than the matching threshold $t_{\text{match}}$, the confidence is zero.

Edge connectivity can be enforced by a simple consistency check: if an edge is traversed in the left image, the corresponding pixels in the right image have to be connected. This can be checked for the stereo edge pixels $\mathbf{p}_1 = (u_1, v_1, d_1)^\top$ and $\mathbf{p}_2 = (u_2, v_2, d_2)^\top$ that are adjacent in the left image. The disparities $d_1, d_2$ are consistent if $|(u_1 - d_1) - (u_2 - d_2)| \leq 1$, meaning that the distance in x-direction of the corresponding edge pixels in the right image is less than or equal to one. In the following pseudo-code listing, this check is referred to by the `isConsistent(...)` function call. The function `neighbors(p)` searches the 8-connected neighborhood

```
function insertNeighbor(p, edge, curConf) // p is a pixel
  if visited(p)
    return false;
  add p to edge;
  if NOT isConsistent(p, parent(p)):    // pixels inconsistent!
    curConf := 0;
    refinedDisparity(p) := UNKNOWN;     // don't trust disparity
  else if curConf >= minFixConf:
    refinedDisp(p) = initialDisp(p);    // confident disparity!
  else    // pixels are consistent but not enough confidence yet
    curConf += confidence(p);           // build up confidence
    refinedDisp(p) := -initialDisp(p); // save with negative sign
    if curConf >= minFixConf:
      for all previous invalid pixels:
        change sign of negative disparities
        linearly interpolate UNKNOWN disparities
  return true;

function followEdge(p, parentEdge, curConf)
  edge := new Edge;
  link edge to parentEdge;
  curNeighbors := p;
  while sizeOf(curNeighbors) > 0:
    if sizeOf(curNeighbors) > 1:       // spawn child edges
      for each neighbor n of curNeighbors:
        childEdge := followEdge(n, edge, curConf);
        link edge to childEdge;
      break;
    if NOT insertNeighbor(curNeighbors[0], edge, curConf):
      break;
    curNeighbors = neighbors(curNeighbors[0]);
  return edge;

function refineDisparities()
  for each matched pixel p: // search for confident start points
    if confidence(p) == 3:
      for each neighbor n of p:
        if  confidence(n) >= 2 AND
            confidence(nextNeighbor(n)) >= 2 AND
            isConsistent(p, n) AND
            isConsistent(n, nextNeighbor(n)):
          e = followEdge(n1, 0, minFixConf); // good confidence!
          if length(e) > 1:
            add e to edges;
  return edges;
```

Listing 1: Pseudo-code of the confidence-based refinement algorithm.

of the pixel for adjacent edge pixels. It disregards the direction of its parent pixel, so we exclusively move forward along the edge.

The underlying idea of the refinement algorithm is to propagate a confidence level along the edge (named `curConf` in Listing 1). First, groups of three adjacent and consistent high-confidence edge pixels are searched for as seed point. Then, starting with maximum confidence, the edge is traversed, checking each pixel for consistency with its predecessor. If an unmatched pixel or an inconsistency is encountered, the confidence is dropped to zero. With each consistent pixel-pair the confidence value recovers until it is greater than the tuning parameter `minFixConf`. This is needed to enforce a larger connected group of consistent edge pixels than just e.g. two. Once the confidence is sufficient, the algorithm tries to recover the intermediate disparities. For inconsistent or unmatched pixels, linear disparity interpolation between the enclosing confident disparities is performed. This way it is possible to keep the total number of matches high and at the same time boost the percentage of correct matches.

Depending on the edge angle, either a horizontal or a vertical pixel-block is matched – whichever is the more perpendicular one. The cost is computed only if the edge angles in the left and right image differ by no more than $\alpha_{\text{match}}$.

### 3.4.4 Edge-Based Dynamic Programming

However, as can be seen in Figure 3.4, a simple winner-takes-all (WTA) matching still shows insufficient performance on horizontal edge-segments since the connectivity information is not incorporated. A mere refinement step like in EM-CBR can only invalidate such edge sections, since it is not able to gain confidence over several connected and consistent edge disparities in these cases. However, if viewed as a whole, the matching of an edge segment can be formulated as the minimization of a cost function that includes matching costs for all individual edge pixels and penalties for violating consistency constraints. Accordingly, the minimum cost solution for the whole edge is actively constructed. This section proposes Edge-Based Dynamic Programming (EBDP), which formulates the solution space as a graph and solves the mentioned minimization problem efficiently as a "shortest path problem" in two dimensions. Dynamic programming refers to a general method for solving problems of a certain structure in an optimal way. In our specific case we use the Dijkstra algorithm (Dijkstra, 1959).

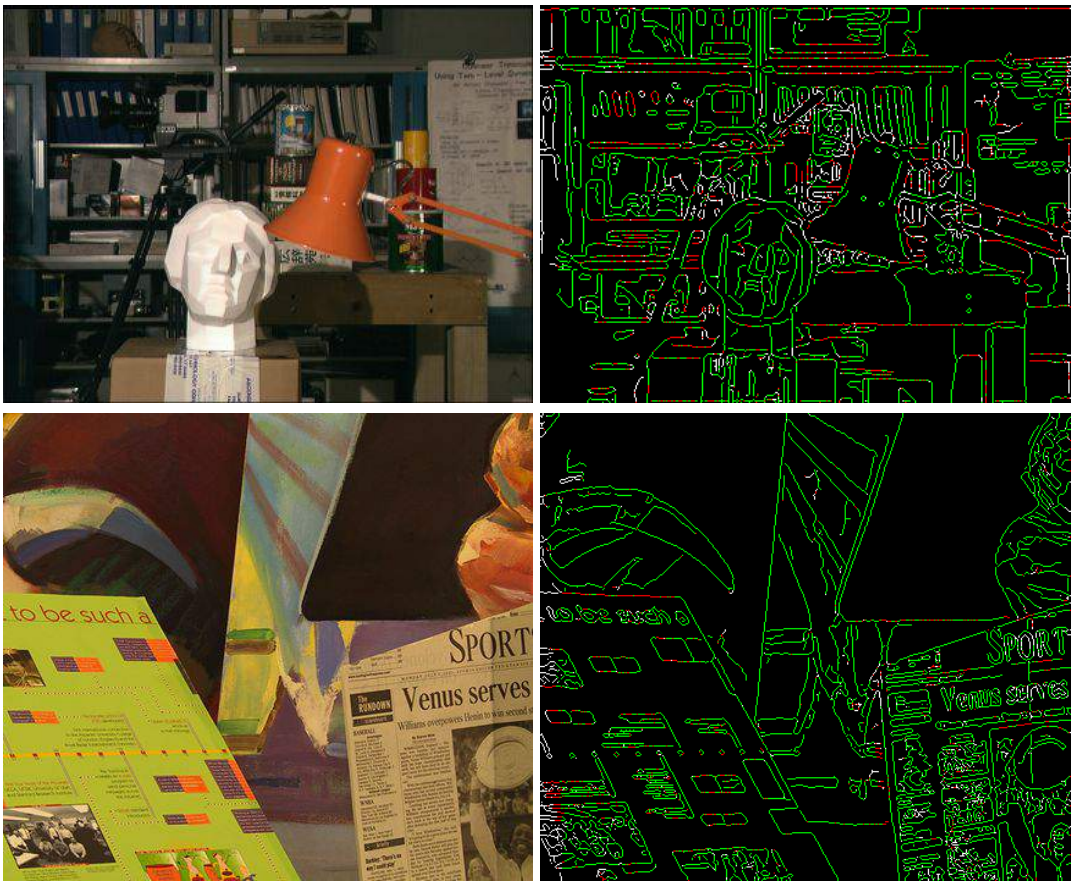



Figure 3.4: Disparity errors of WTA matching with a 15x1 pixel block for Tsukuba (top right) and Venus (lower right) image sets. For textured and non-horizontal edges, the majority of edge matches are correct. However, the inherent ambiguities along scanlines prevent a simple approach like WTA to generate sufficient results in most of the horizontal edges in the Tsukuba image set. Green: $e \leq 1$ px, red: $e > 1$ px, white: no match.

The method works as follows. After edges have been detected, edge chains are extracted by looking for edge pixels with only one neighbor (i.e. edge endpoints) and traversing the edge until a branch or no more (8-connected) neighbors are detected. At branches new edge chains are started. For each edge chain that was found, all disparity matches are computed according to Section 3.4.2. Finally, we can build a graph for each chain as depicted in Figure 3.5 and formulate an independent optimization problem. Due to this independence, the complexity is significantly reduced compared to a comprehensive global optimization formulation. The graph can be laid out in two dimensions as a shortest path problem, in which each node $\mathbf{n}_{i,j}$ corresponds to an edge point with index $i$ along the edge

chain and a disparity index $j$. Disparity indices merely enumerate all matching possibilities/disparities of an edge pixel $i$. Accordingly, the more matching candidates an edge pixel has, the more nodes there will be in the disparity index dimension. The connectivity information between edge chains at branches is currently not used as it would complicate the optimization problem formulation significantly.

The circles in Figure 3.5 represent the graph nodes along with their disparity $d(\mathbf{n}_{i,j})$ (black) and matching cost $m(\mathbf{n}_{i,j})$ (blue). The yellow nodes in the top row $\mathbf{n}_{i,0}$ of the graph are "no match"-nodes, for which the disparity is not defined (and thus set to $-1$ in this case). They have a matching cost of $m_{\text{nomatch}} = 12.5$ which is effectively a tuning parameter to control match willingness in case of ambiguity. The nodes with edge point indices 1 to 4 correspond to actual edge pixels, while node $\mathbf{n}_{0,0}$ is a virtual starting node with cost zero. It is needed to treat the first edge pixel equal to all subsequent edge chain pixel. Green nodes are special "gap-filler"-nodes with a defined disparity and matching cost $m_{\text{nomatch}} + \epsilon = 12.6$ which are introduced to fill gaps if no match with an appropriate disparity difference of one or zero is detected at the next edge pixel. The $\epsilon = 0.1$ is added to favor the "no-match"-nodes in the upper row in case of doubt. Effectively, this disallows graph edges from these nodes to all nodes with a disparity difference bigger than one, since the "no-match"-nodes in the upper row are always the cheaper path. One can see that for edge pixel 2 no match was found, as only a "no-match"-node and a "gap-filler"-node are available.

The graph edges are visualized by arrows, along with their penalty $p(\mathbf{n}_{i,j}, \mathbf{n}_{i+1,k})$ that is added to the edge's total cost. The second contributor to the edge cost $C(\mathbf{n}_{i,j}, \mathbf{n}_{i+1,k})$ is the matching cost $m(\mathbf{n}_{i+1,k})$ of the destination node. We chose this separate notation for visualization, because the matching cost does not depend on the previous node, while the penalty term $p(\mathbf{n}_{i,j}, \mathbf{n}_{i+1,k})$ does. The cost of a path $P = \{\mathbf{n}_0, \mathbf{n}_1, ..., \mathbf{n}_N\}$ in this graph accordingly is the sum of all matching costs and edge penalty terms that lie on its way:

$$C(P) = \sum_{i=1}^{N} m\left(\mathbf{n}_i\right) + p\left(\mathbf{n}_{i-1}, \mathbf{n}_i\right)$$

The penalty term $p(\mathbf{n}_{i-1}, \mathbf{n}_i)$ is used to favor paths that have consistent disparities along edge segments. Depending on the change in disparities between two

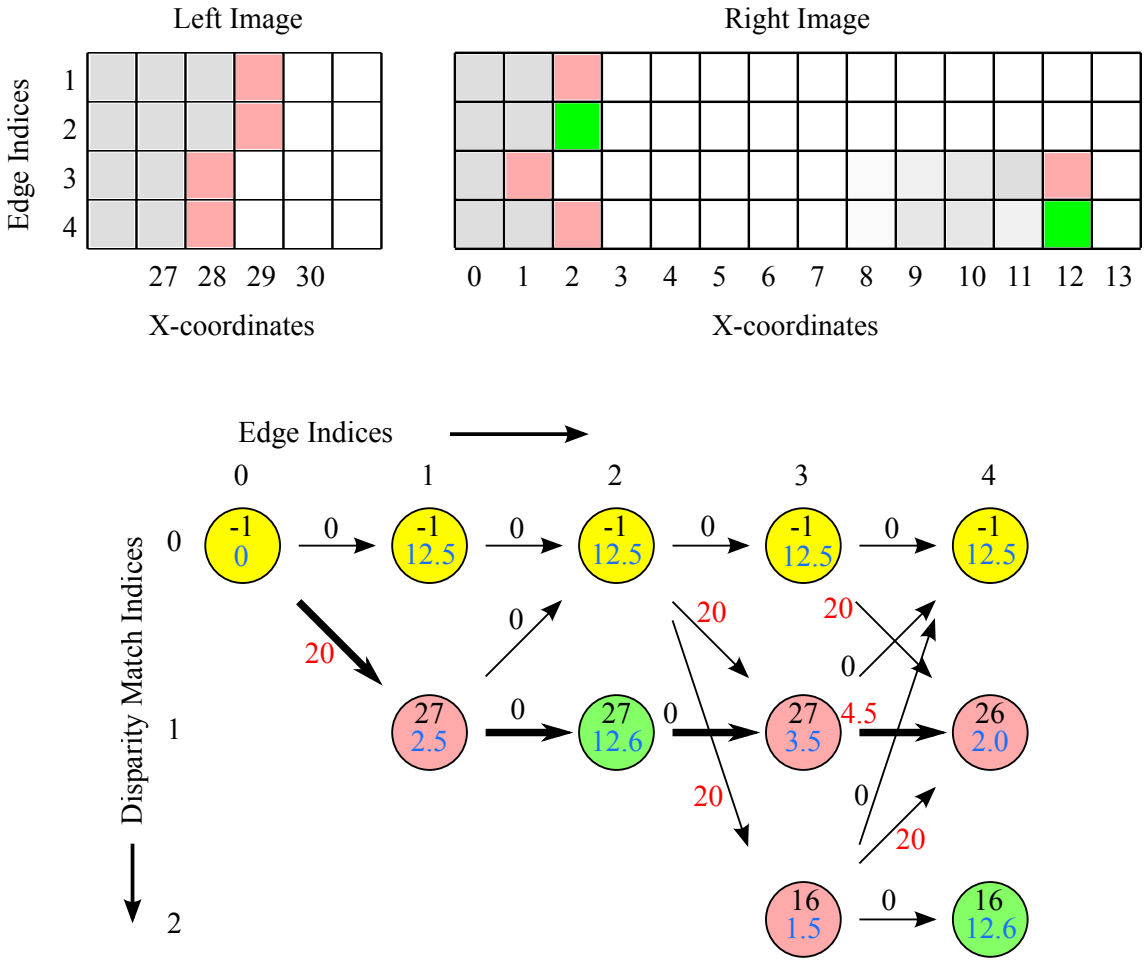



Figure 3.5: The 4 pixel edge chain in the left image (red) shall be matched. The regular edge pixel matching candidates in the right image and in the graph are drawn in red, while virtual "gap-fillers" are drawn in green. The yellow nodes in the graph are "no-match"-nodes. The disparity (black font in nodes) determines the correspondence from the left to the right image edges. The associated match cost is drawn with blue font. All edges (arrows) are annotated with their penalty. The minimum cost path is depicted by the bold arrows.

nodes, different penalties are assigned

$$p(\mathbf{n}_{i-1}, \mathbf{n}_i) = \begin{cases} 0, & \text{if } |d(\mathbf{n}_{i-1}) - d(\mathbf{n}_i)| = 0 \\ p_{\text{step}}, & \text{if } |d(\mathbf{n}_{i-1}) - d(\mathbf{n}_i)| = 1 \\ p_{\text{jump}}, & \text{if } |d(\mathbf{n}_{i-1}) - d(\mathbf{n}_i)| > 1 \end{cases} . \tag{3.5}$$

The tuning penalty term $p_{\text{step}}$ is added, when the disparity of adjacent edge pixels equals one, to reward constant disparities along horizontal edges. A larger

penalty term $p_{\text{jump}}$ is added for arbitrary disparity jumps or the transition from a "no-match"-node to a matched node. The values that are used in Figure 3.5 are $p_{\text{step}} = 4.5$ and $p_{\text{jump}} = 20$ and are highlighted in red.

Now that we have defined a graph with non-negative graph edge costs, we can apply standard techniques to find the minimum cost path along edge segments. In stereo algorithms the efficient optimal solution to this kind of problem is usually referred to as dynamic programming. In terms of graph algorithms, this is equivalent to Dijkstra's algorithm. We organize all nodes in two sets: the visited and the unvisited set. Initially, the visited set consists of only the starting node $\mathbf{n}_{0,0}$ while the rest is in the unvisited set. A brief outline of the algorithm can be given as follows:

1. Find the node in the unvisited set with minimal cumulative cost that is adjacent to a node in the visited set.

2. Remove this node from the unvisited set and add it to the visited set. The final minimum path to this node is now known.

3. Exit if the node belongs to the last pixel in the edge segment, otherwise go to step 1.

In Figure 3.5 we start by examining the adjacent nodes of $\mathbf{n}_{0,0}$ and find that the minimum cost node is $\mathbf{n}_{1,0}$. In the following, we denote the minimal cost from $\mathbf{n}_{0,0}$ to $\mathbf{n}$ with $\tilde{C}(\mathbf{n})$. As it is now added to the visited set, we take all its adjacent nodes into account when we search for the next node with minimal cost in the unvisited set. Thus, our next options are $\tilde{C}(\mathbf{n}_{2,0}) = 25$ and $\tilde{C}(\mathbf{n}_{1,1}) = 22.5$. We repeat this process until $\mathbf{n}_{4,1}$ is added to the visited set, which signals that the minimum cost path for the edge segment has been found, since this is the first node of the last pixel that is added to the visited set (the bold arrows in Figure 3.5 denote the final minimum cost path). Note that the cost formulation is symmetric, so if we started at the other end of the edge we end up with the same minimum cost path. The virtual start node would move to the other side, though. Some of the "gap-filler"-nodes possibly change positions, too. But due to the $\epsilon$ added to their match cost, one can see that any "gap-filler"-node has to be enclosed by at least two real match nodes. If this is not the case, the minimum path would rather lie on the cheaper "no match"-nodes.

If a lower bound for the remaining path cost can be calculated, the optimal path can be calculated in a more goal-oriented manner, effectively reducing the number of node evaluations. In the Dijkstra algorithm only the cumulative cost $\tilde{C}(\mathbf{n}_{i,j})$ is considered as a measure in order to determine which node should be examined next. Thus, the search is undirected. The A* search, as proposed in (Hart et al., 1968), makes use of a heuristic to direct the search towards the goal. We use a simple heuristic $H(\mathbf{n}_{i,j})$ which depends on the edge-segment length $M$, the edge point index $i$ and a tuning parameter $m_{\text{mincost}}$

$$H(\mathbf{n}_{i,j}) = (M - i)m_{\text{mincost}} \tag{3.6}$$

With this heuristic, step 1 of the algorithm now searches for the lowest goal-directed cost $\hat{C}(\mathbf{n}_{i,j}) = \tilde{C}(\mathbf{n}_{i,j}) + H(\mathbf{n}_{i,j})$. Theoretically it is possible to have zero matching costs, so in order to compute optimal paths in all instances, $m_{\text{mincost}}$ would have to be zero which would be equivalent to the Dijkstra search. For real image sets the optimal matches were still found for nonzero heuristics, while reducing the computational effort notably.

### 3.4.5 Experimental Results

In the following, we benchmark EMCBR and EBDP with the Middlebury database.[1] The selected image sets are shown in Figure 3.6. The parameterization of EMCBR and EBDP was empirically determined and set as follows: $\alpha_{\text{match}} = \pi/16$, $t_{\text{match}} = 12.0$, `minFixConf` $= 8$ and the maximum disparity was set to 64. Where not stated differently, EMCBR used a $15 \times 5$ pixel block for matching, while EBDP used a more narrow $15 \times 1$ support region. The EBDP specific parameters were chosen to $m_{\text{nomatch}} = 12.5$, $m_{\text{mincost}} = 1.0$, $p_{\text{step}} = 4.5$, $p_{\text{jump}} = 20$. All values were left unchanged throughout the experiments.

We first evaluate the influence of different support regions on matching performance. For this we collected results with WTA and EMCBR on the Middlebury dataset. Figure 3.7 shows the performance of 5 different support region configurations. We denote the proposed shifted pixel blocks that choose the minimum of the left/right-hand side support region cost by $X \times Y$. For horizontal edges, this

[1] The datasets are available on the vision homepage of Daniel Scharstein and Richard Szeliski `http://vision.middlebury.edu/stereo/`


Figure 3.6: Middlebury benchmark stereo image sets. The test images in the left column are Teddy (450 × 375), Sawtooth (434 × 380), Barn1 (432 × 381), Bull (433 × 381) and Venus (434 × 383). The images in right column are Cones (450 × 375), Tsukuba (384 × 288), Barn2 (430 × 381), Reindeer (447 × 370) and Poster (435 × 383).

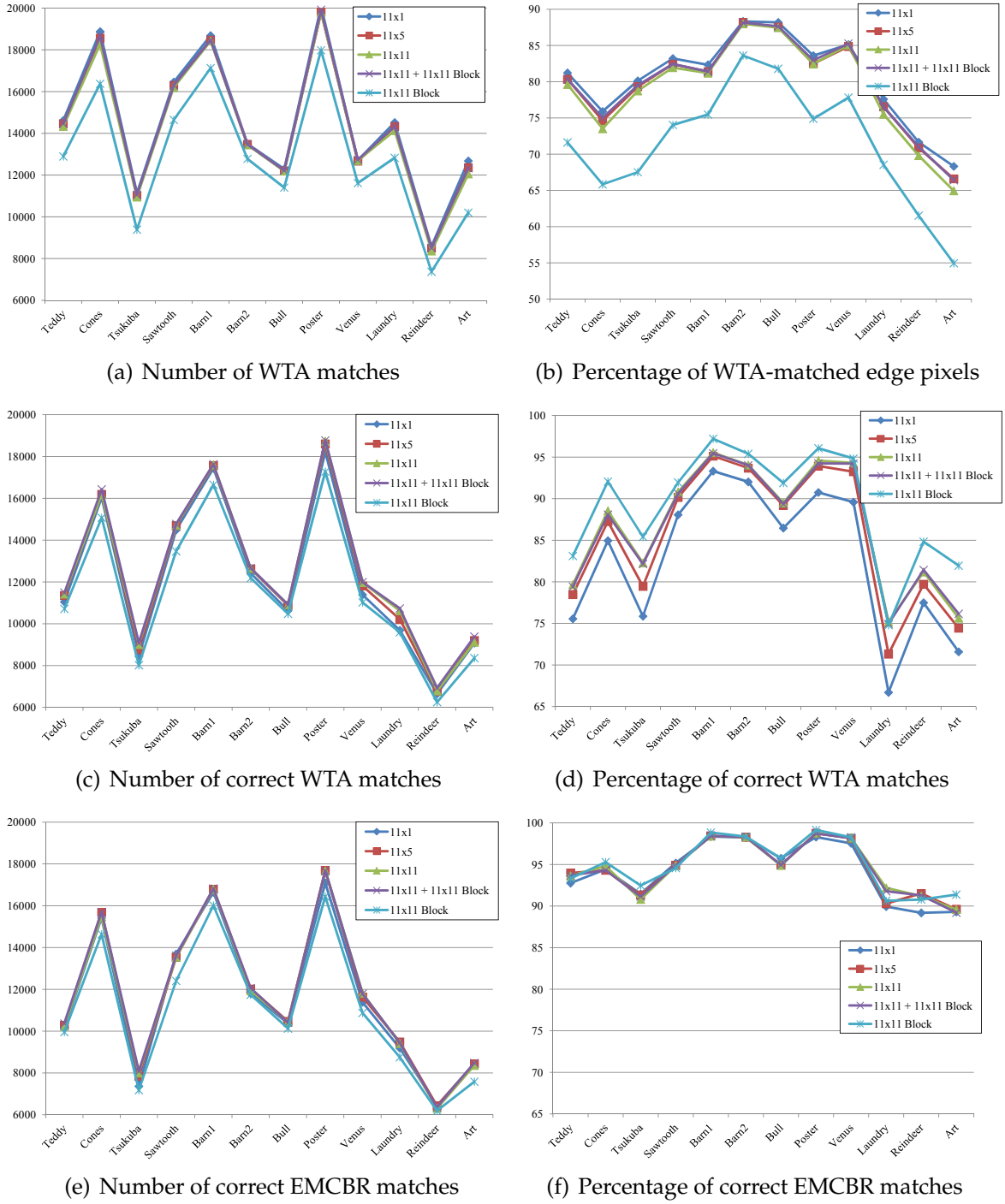
automatically becomes a $Y \times X$ region. Classic symmetrical block support regions that are centered on the edge pixels are denoted by $X \times X$ *Block*. In the special case of *11×11 + 11×11 Block* we mean that the minimum cost of the left, right and centered *11×11* support region is taken.

Inspecting the percentage of correct matches with pure winner-takes-all (WTA) in Figure 3.7(d), we can see that, despite being quite fast, the results are not yet overwhelming. It is visible, that the size of the support region has a considerable influence on the quality of the WTA matches.

In Figures 3.7(e) and 3.7(f) the results of EMCBR are shown. What is specifically interesting is that the dependency on the support region from Figure 3.7(d) has lessened significantly, which is due to the incorporation of edge connectivity information. Effectively, adjacent edge pixels build one big virtual support region along the edge, when disparity smoothness is enforced.

It is also evident that the total number of correct matches for classic symmetrical block matching (*11×11 Block*) is the lowest, see Figure 3.7(e). While this seems insignificant, the missing disparities often lie on object borders which are very interesting for robotic vision tasks. Another observation is, that the addition of a symmetrical pixel block does not seem to yield much further information, since the results of the *11×11* shifted support window and the combined *11×11 + 11×11 Block* window are basically indistinguishable. In the case of EMCBR, the best robustness/performance trade-off seems to be the *11×5* and in most cases even the *11×1* window.

A comparison of both EMCBR and EBDP with probabilistic phase-based sparse stereo (PPBSS, (Ulusoy et al., 2004)) and several popular dense methods is given in Table 3.1 and visually in Figure 3.8 and 3.9. The results of scanline optimization (SO), dynamic programming (DP) and graph cuts (GC) refer to (Scharstein and Szeliski, 2002), while semiglobal matching (SemiGlob) refers to (Hirschmüller et al., 2002), AD-Census (ADCensus, one of the best dense algorithms according to its Middlebury benchmark results) to (Mei et al., 2011) and graph cuts with occlusions (GC+occl) to (Kolmogorov and Zabih, 2001). Thus, a diverse mix of scanline-based algorithms to complex global optimization techniques is compared. To yield suitable sparse ground truth from the dense disparities, the Middlebury ground truth images were dilated with a $3 \times 3$ structuring element to always yield foreground disparities on object borders. Subsequently the images were sparsified by extracting only the disparities at edge locations. Results




(a) Number of WTA matches

(b) Percentage of WTA-matched edge pixels

(c) Number of correct WTA matches

(d) Percentage of correct WTA matches

(e) Number of correct EMCBR matches

(f) Percentage of correct EMCBR matches

Figure 3.7: Comparison of the influence of several different support regions on matching performance. Shifted pixel blocks (11x1, 11x5, 11x11), a simple block match (11x11 Block) and a combined variant (11x11 + 11x11 Block) are tried. Simple WTA matching and EMCBR are evaluated.

were available (with few exceptions) for the Tsukuba, Teddy, Cones, Venus and Sawtooth image sets. Additionally we compare EMCBR and EBDP for the Barn1, Barn2, Bull, Reindeer and Poster image sets for which the results are given in Table 3.2 and Figure 3.10 and 3.11.

Table 3.1: Edge matching performance comparison of sparse (EBDP, EMCBR and PPBSS) and dense algorithms that have been sparsified to edge loci.

| | Tsukuba | | Teddy | | Cones | | Venus | | Sawtooth | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Matches | Errors | Matches | Errors | Matches | Errors | Matches | Errors | Matches | Errors |
| **EBDP** | 9608 | 7.2% | 11715 | 9.0% | 16442 | 5.3% | 12060 | 1.4% | 14997 | 2.4% |
| **EMCBR** | 8550 | 8.8% | 10514 | 5.3% | 16147 | 5.3% | 11816 | 2.0% | 14202 | 2.4% |
| PPBSS | 2350 | 17.0% | - | - | - | - | 1310 | 6.0% | 3079 | 4.0% |
| DP | 13775 | 14.6% | 17192 | 11.1% | 23590 | 12.6% | 14601 | 6.7% | 19393 | 6.0% |
| SO | 13778 | 16.9% | 17709 | 20.8% | 24489 | 19.3% | 14770 | 8.2% | 19598 | 7.1% |
| SemiGlob | 13765 | 8.6% | 18001 | 10.6% | 24868 | 9.0% | 14930 | 1.5% | - | - |
| GC | 13801 | 9.0% | 17696 | 17.4% | 24399 | 15.0% | 14737 | 2.9% | 19560 | 4.9% |
| GC+occl | 13803 | 6.3% | 17995 | 15.8% | 24868 | 11.6% | 14929 | 1.7% | 19777 | 0.8% |
| ADCensus | 13900 | 20.8% | 18001 | 6.9% | 24868 | 6.8% | 14930 | 0.4% | - | - |

Table 3.2: Additional edge matching performance comparison of EBDP and EMCBR.

| | Barn1 | | Barn2 | | Bull | | Reindeer | | Poster | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Matches | Errors | Matches | Errors | Matches | Errors | Matches | Errors | Matches | Errors |
| **EBDP** | 16856 | 1.1% | 12554 | 1.0% | 11607 | 3.4% | 7278 | 8.9% | 18262 | 1.1% |
| **EMCBR** | 16140 | 1.2% | 11697 | 1.3% | 10422 | 3.3% | 6559 | 10.6% | 17051 | 1.7% |

The most obvious difference between the matching results of the sparse and the dense methods is the number of matches. This stems from inconsistently detected edges in the left and right images. For example the upper bound for correctly matched edge pixels (for our given edge detection results) in the Tsukuba image set without gap filling is 8529 matches. This number is calculated by taking the ground truth disparities at edge loci in the left image and checking if an edge with an edge angle difference smaller than $\alpha_{\text{match}}$ exists at the corresponding location in the right image. Since the dense algorithms do not restrict their disparity search to edge loci, this is the main reason for the difference in match counts. However, this is nonrelevant for the applications of sparse methods. It is much more important to extract consistent edge segments on sparsely textured objects. The Middlebury stereo sets can be regarded as a stress test for edge-based stereo matchers since they are highly textured, leading to many inconsistently detected edges. Nevertheless, EMCBR performs well in terms of error percentages, especially if one takes into account that (except for DP and SO) very sophisticated dense algorithms are compared which take at least seconds to execute on

a modern CPU without resorting to specialized hardware. Except for some long horizontal edge segments as in Tsukuba, the algorithm efficiently and reliably finds the correct disparities, while being much less of a computational burden. In comparison to PPBSS, many more features are detected while error rates are superior for all three comparable image sets. In further research the number of matches has been found to be critical for tasks like edge-based SLAM. Also, since PPBSS uses a set of Gabor filters with different rotations and scales for detection, one can appraise that it is computationally more costly than EMCBR.

As the overall matching performance of EMCBR is already decent, significant improvements by EBDP can only be seen in the challenging Tsukuba image set. Here, EBDP is able to recover most horizontal edges, while EMCBR rejects many of these edges due to inconsistencies. Unfortunately, like many dense methods, EBDP produces an increased number of false matches in the lower part of the Teddy image set due to the slanted ground plane. However, overall EBDP is able to recover more matches with fewer errors due to its optimization approach. This comes at a computational cost, though.

The experiments were run on an Intel i7-2640M CPU (2.8 GHz) using both cores. For a maximum disparity of 64, the computation times of EMCBR per image set were between 20ms and 45ms with the $11\times5$ support region including preprocessing and refinement. Cost aggregation and matching takes about 10ms to 15ms and about the same amount of time is spent on refinement. The overall computation times for EBDP were about 60-85ms and include 6-15ms Canny edge detection and angles computation. Approximately 5-13ms are used for confidence-based refinement and edge segment extraction. For the Cones image set, about 55ms are spent on cost aggregation and optimization. This phase with WTA matching (as done in EMCBR) takes less than 10ms. Accordingly, EMCBR is usually about twice as fast as EBDP.

Despite the popularity of the Middlebury stereo benchmark, its relevance for a number of practical applications is arguable. The scenes were carefully constructed and are mostly well textured. However, this work is particularly concerned with untextured scenes, where edges may be the only available visual features. Dense algorithms are known to struggle in such cases, as large regions are highly ambiguous, while edge-based algorithms only investigate areas that actually carry information (i.e. edges). An evaluation of EMCBR and EBDP in less untextured real environments is very interesting, accordingly.

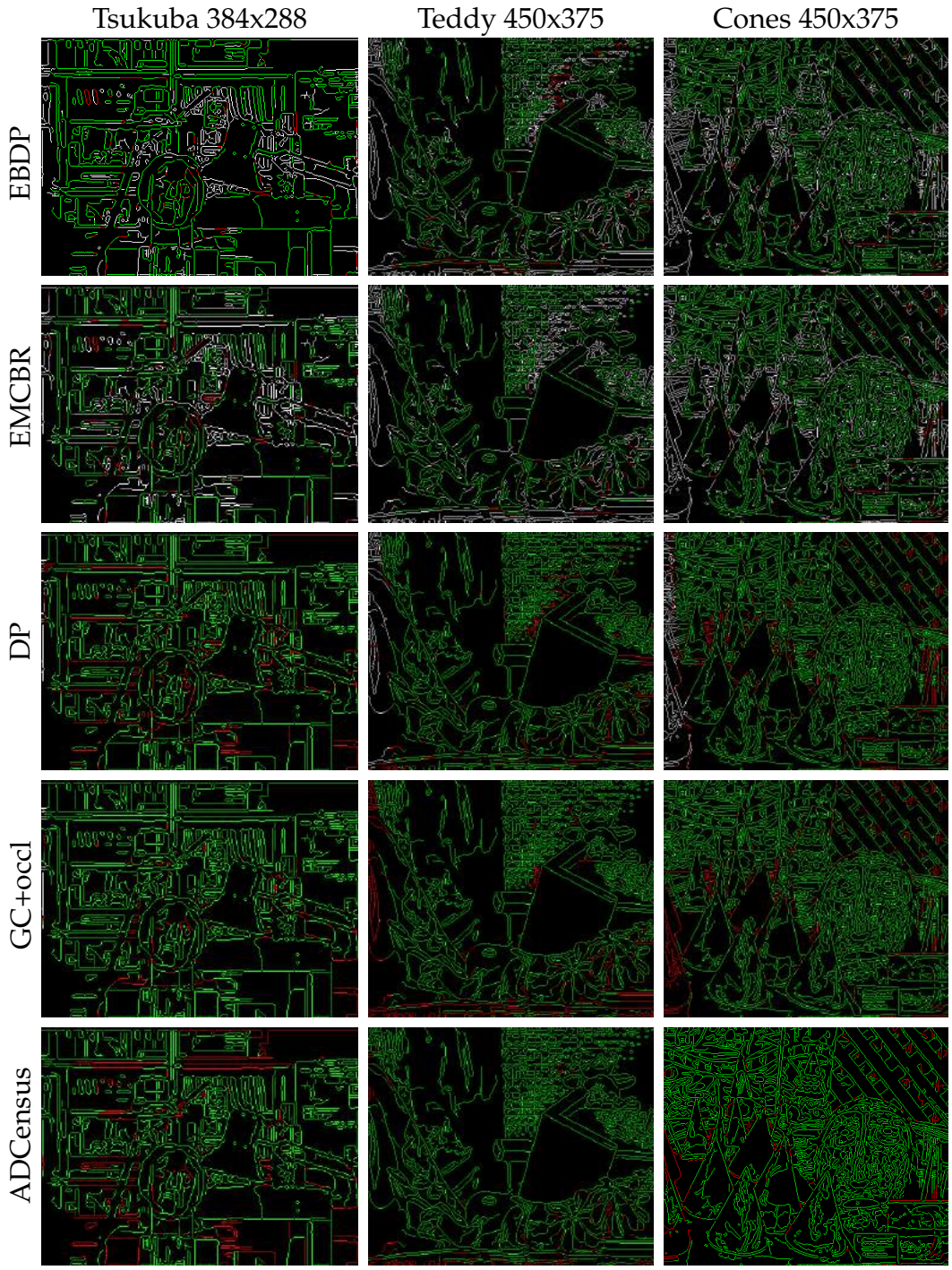



Figure 3.8: Comparison of EBDP and EMCBR to sparsified popular dense algorithms. Green pixels depict disparity errors $\leq 1$, red pixels are errors $> 1$ and white pixels correspond to unmatched pixels. The dense methods usually match almost all edge pixels.

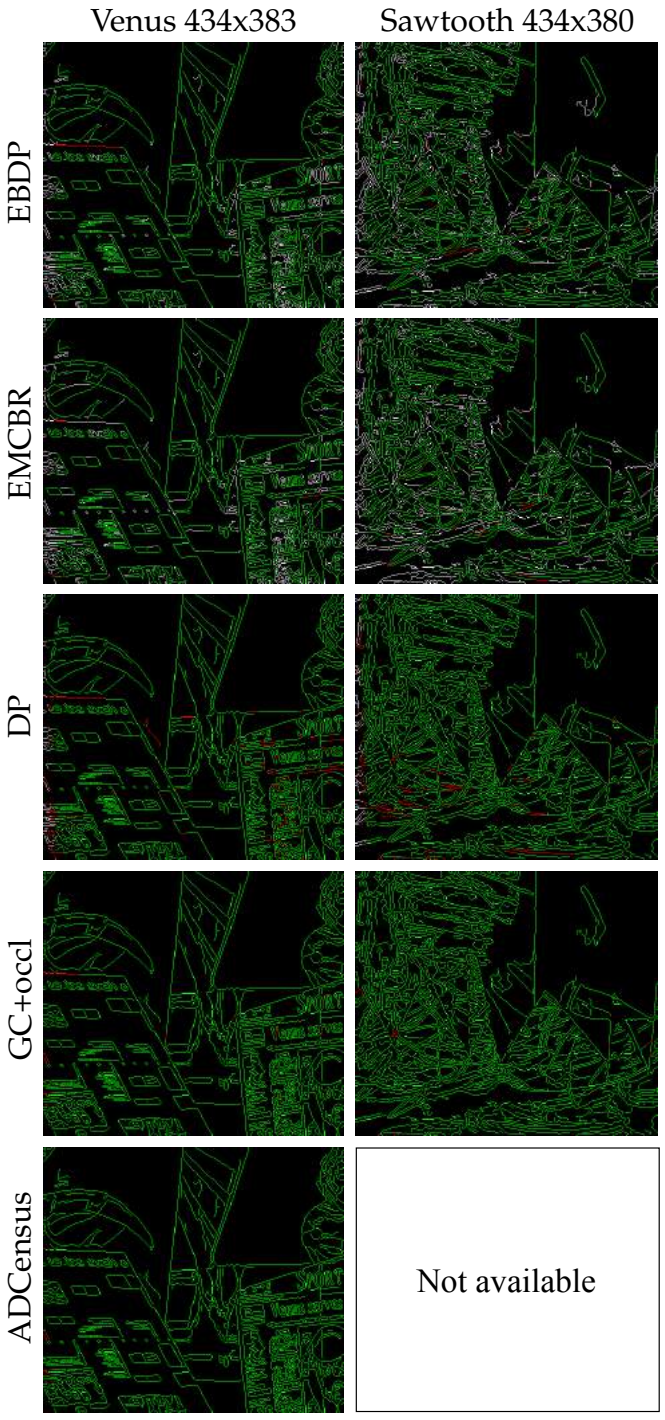



Figure 3.9: Comparison of EBDP and EMCBR to sparsified popular dense algorithms. Green pixels depict disparity errors $\leq 1$, red pixels are errors $> 1$ and white pixels correspond to unmatched pixels.

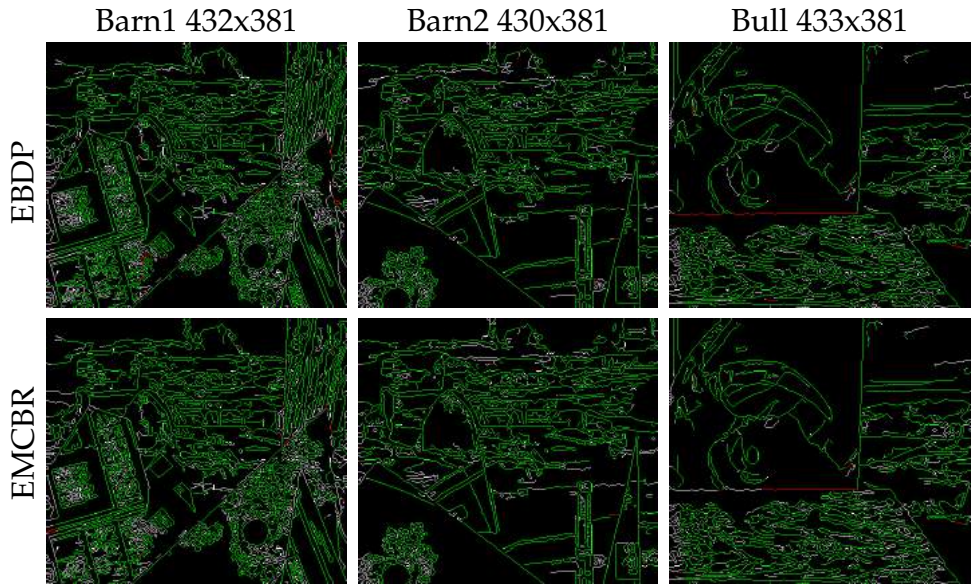



Figure 3.10: Comparison of EBDP and EMCBR. Green pixels depict disparity errors $\leq 1$, red pixels are errors $> 1$ and white pixels correspond to unmatched pixels.

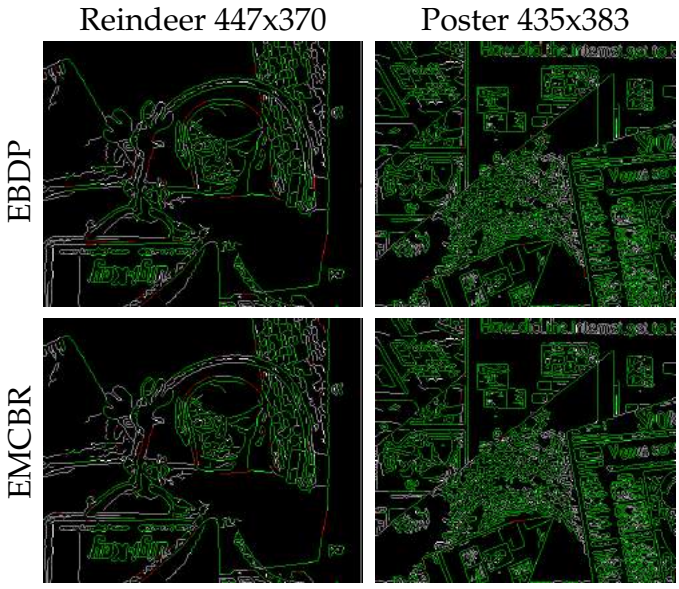



Figure 3.11: Comparison of EBDP and EMCBR. Green pixels depict disparity errors $\leq 1$, red pixels are errors $> 1$ and white pixels correspond to unmatched pixels.

## 3.5 Stereo Line Matching

Moving the test cases closer to the final application of line-based localization and mapping in untextured environments, it is also interesting to evaluate a different algorithm category: stereo line matching. In the case of EMCBR and EBDP, stereo lines would be generated by detecting line segments from the final sparse disparity maps. Another possibility is to directly perform stereo line matching. In this case, line segments are first detected in the left and right image of a stereo set and then are matched as atomic primitives. Here, we describe a stereo line matching technique that is largely inspired by Li (1994).

In contrast to wide-baseline stereo line matching techniques (Bay et al., 2005; Fan et al., 2010; Wang et al., 2009) we make the assumption that the epipolar geometry is known and therefore rectified stereo pairs are expected as input to the described algorithm.

### 3.5.1 Line Segment Detection

As a prerequisite, line segment detection is a vital component of both stereo line matching with known epipolar geometry and the general case of line matching. A classic and widely used approach for line detection is the Hough transform, see Illingworth and Kittler (1988) for a comprehensive survey. It performs voting for every pixel in the source image in a discretized solution space, the so-called Hough space. The Hough transform is rather general in scope, as it can be fitted for many shape detection applications. But for the case of lines, the Hough space is usually parameterized by the line angle $\theta$ and the distance to the origin $d$, forming a two-dimensional solution space, see Figure 3.12. As the source image, the output of a Canny edge detector is used. Going through all edge pixels in the source image, the voting mechanism iteratively increases all counts for the bins in Hough space that correspond to a line that passes through the current edge pixel in question. Afterwards, the winning lines are found by non-maxima suppression in Hough space.

However, since infinite lines are detected, connected segments along the line direction still have to be found in a separate step. Also, the voting scheme together with a fine discretization unfortunately is relatively expensive since it is basically a brute force approach (Matas et al., 1998). Probabilistic approaches try

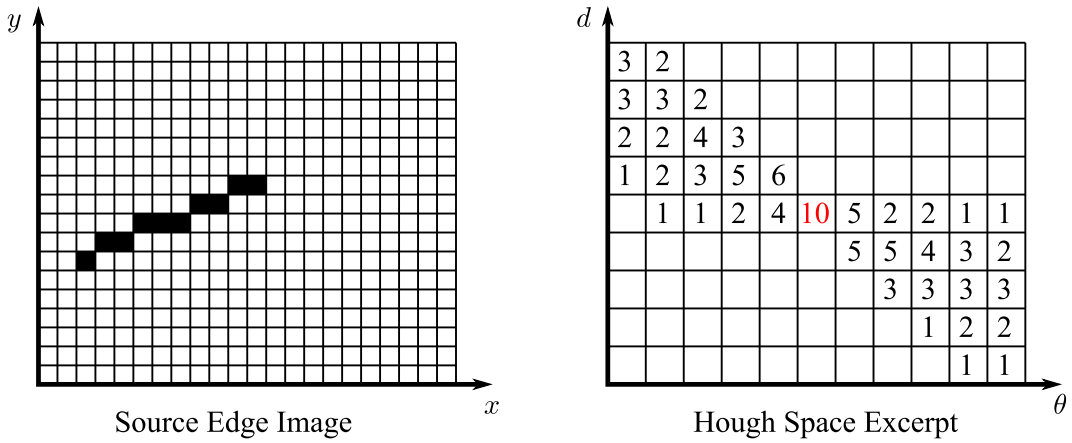



Figure 3.12: The source image on the left contains one line segment with 10 edge pixels. The corresponding Hough space (excerpt shown on the right) contains votes for all the possible lines that go through each individual edge pixel. The maximum (red) represents the best line, that contains the most (in this case all) edge pixels.

to reduce the burden of examining every pixel by only taking a randomized subset of all edge pixels. With the progressive probabilistic Hough transform, a popular approach was presented by Matas et al. (1998, 2000). It integrates the line segment detection by immediately extracting line segments (that are determined by the largest connected contributing edge), once a threshold is reached. The votes of all points that belong to the extracted line segment retract their votes to not clutter the Hough space unnecessarily. For evaluation, the OpenCV implementations of both methods were used (Bradski, 2000). In Figure 3.13, the detected lines for two indoor images are shown. In the case of the standard Hough transform, the resulting lines were directly plotted, instead of implementing line segment extraction as a post processing step. The computation times were around 20ms in the case of the standard and about 30ms in the case of the progressive probabilistic Hough transform. Both algorithms parameterize the Hough space with 1px resolution for $d$ and $1^\circ$ resolution for $\theta$. Especially when looking at the top left image in Figure 3.13, one can see that pure line detection in contrast to line segment detection (top right) significantly complicates the inference of structure even for a human observer. And also in computer vision, many applications (like SLAM and object detection) require the location of features (in this case line segments) to deliver meaningful results. Therefore, we will further only consider the progressive probabilistic Hough transform.

Figure 3.13: Two images from a corridor, for which the classic Hough line detection was performed in the left column. The right column shows the results with the progressive probabilistic Hough transform.

The quantization of the solution space and global voting are both the biggest strength and weakness of the Hough transform. In the face of noise, the Hough transform is able to detect fragmented lines, since (ideally) one single bin accumulates all corresponding edge pixel votes and generates a maximum in Hough space. However, every edge pixel votes for a large number of lines – in fact all lines that go through that pixel. Since only one of the votes actually corresponds to the correct line, this creates a computational burden, especially as complexity quickly increases with more fine grained Hough quantizations. Also, cluttered and coincidently sufficiently lined up edge pixels can generate false line detections as can be seen in the middle of the top left image of Figure 3.13.

Another problem is, that the Hough space quantization can force distinct maxima, when (long) image space lines lie between bins. This is especially visible for the vertical lines in the lower left image from Figure 3.15. While this effect can be reduced by increasing the Hough space resolution, the computational cost increases quadratically. In general, the Hough transform does not seem to be the

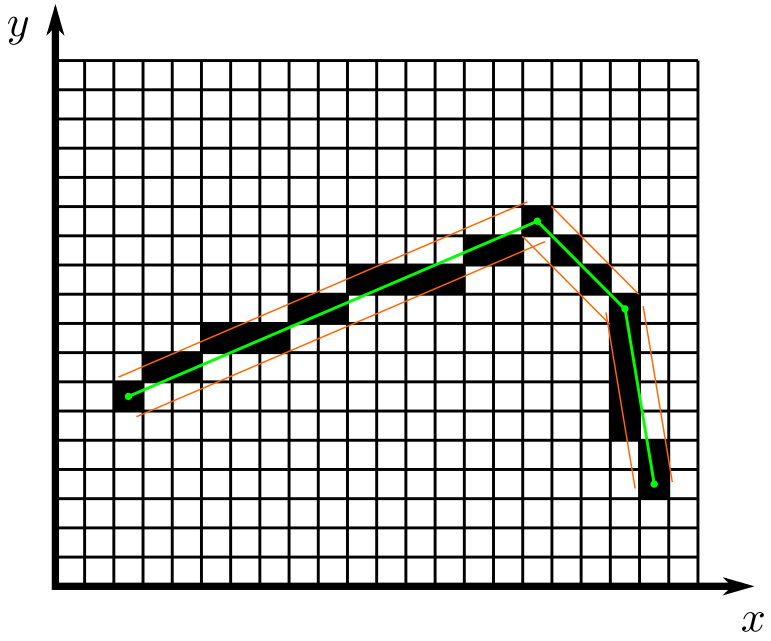



Figure 3.14: Douglas-Peucker algorithm approximating a sequence of connected pixels (black) with line segments (green). The $\epsilon$-margins with $\epsilon = 0.7$px are shown in orange. Since pixel connectivity is required during extraction, noisy edges will reduce the quality of detected line segments significantly.

ideal choice, when noise levels are not high.

We will now look into a different approach that is based on the Douglas-Peucker algorithm. For this, all connected edges are extracted from the input image and line segment sequences that approximate the original edges are computed according to Douglas and Peucker (1973). Their algorithm is parameterized with an $\epsilon > 0$ that determines how far pixels of the current line segment are allowed to deviate from its center. This is visualized in Figure 3.14, where the orange lines depict the $\epsilon$-range around the final segments (green). Starting from the left end point, the algorithm iteratively steps through all connected edge pixels and checks whether all pixels lie within $\epsilon$-range of the current line segment that is formed by the segment start point and the currently examined pixel. If a violation is detected, a new line segment is started. This results in single outliers being able to break line segments in two. However, this can easily be fixed during a post processing step. The main benefits over the Hough transform are that the solutions are not quantized in a predetermined solution space and allow for arbitrarily oriented and positioned line segments. And since pixel connectiv-

Figure 3.15: Comparison of the progressive probabilistic Hough transform (left) to Douglas-Peucker line segment detection with line fitting (right). For the tested scenes, computation times (30ms vs. 8ms on average) as well as detection quantity and quality are significantly superior in case of Douglas-Peucker.

ity information is used (in contrast to the Hough transform), no computational resources are wasted on unlikely lines. In practice, this decreases the computational complexity significantly.

However, in general, the Douglas-Peucker algorithm will not deliver line segments that fulfill any optimality criterion, and even different polygonizations are possible depending on the start point. It is also apparent that edge connectivity is vital for this approach to generate adequate results.

In this work we refine the initially returned segments by line fitting. This ensures that line segments are optimal with respect to pixel locations for a given edge segmentation (which is still determined by Douglas-Peucker). Experiments showed that this approach performs very well both in terms of computational performance and detection quality for our designated environments. Figure 3.15 compares the results to the progressive probabilistic Hough transform, for which processing times were 30ms in contrast to 8ms on average in the case of Douglas-

Peucker (including line fitting). In the rest of this work, Douglas-Peucker with line fitting will always be used when line segment detection is involved.

### 3.5.2 Stereo Line Matching Using Dynamic Programming

In this section, we consider the problem of matching line segments in two rectified images with a known small baseline to each other. We are especially interested in non-textured scenes, where traditional stereo methods that utilize neighborhood similarity scores fail. From the small baseline stereo setup we derive the following assumptions:

- Line segments have similar angles in the left and right image.
- The images are rectified so that a y-coordinate corresponds to the same scanline in both images.
- The order of corresponding pixels in each scanline from left to right is never reversed.

Since the absence of texture in indoor scenes is often problematic for local stereo methods, we derive our algorithm from the global method proposed in Li (1994). It exploits the strict scanline ordering assumption and formulates a 2D search problem that can be solved for each cluster of similar line orientations. Dynamic programming (again, effectively Dijkstra's algorithm) is employed to find the globally optimal solution (under the assumptions).

In order to find line orientation clusters, binning is performed. As shown in Figure 3.16, the bins cover a semicircle and adjacent bins overlap up to their center. As a consequence, each line segment is contained in exactly two bins. After one bin is processed with dynamic programming, its members are removed from all bins so that none are processed twice. An overview of the algorithm is given in the following:

1. Find largest orientation bin
2. Assemble unmatched right image line segments with similar orientations
3. Find matches with dynamic programming for this line orientation cluster
4. Empty bin and also remove its members from all other bins

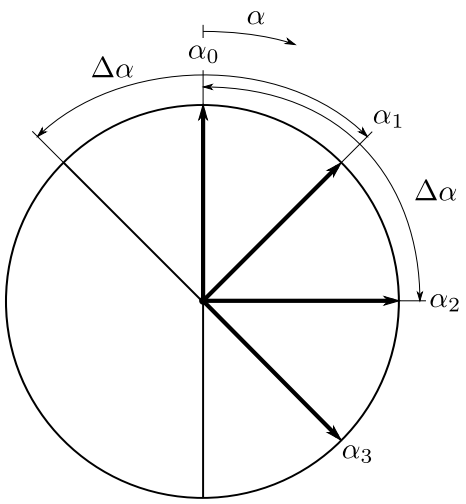


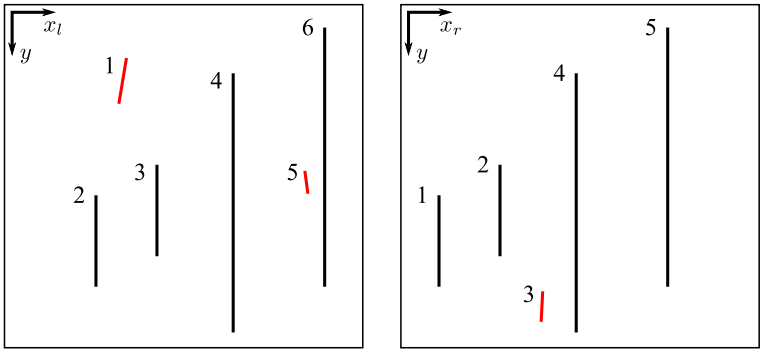

Figure 3.16: Line direction binning with $\Delta\alpha = 90^\circ$ resulting in 4 overlapping bins $\alpha_{0..3}$ that together cover $180^\circ$. Because of the overlap, every line segment is contained in exactly two bins. This is done to not split large orientation clusters due to arbitrary bin bounds. For example, all line directions with $0 \leq \alpha < 45^\circ$ are contained in both the $\alpha_0$ and $\alpha_1$ bins that have an angular range of $-45^\circ \leq \alpha < 45^\circ$ and $0 \leq \alpha < 90^\circ$ respectively (note that $-45^\circ = 135^\circ$ in the semicircle).

5. Update the set of unmatched right image line segments with new matches

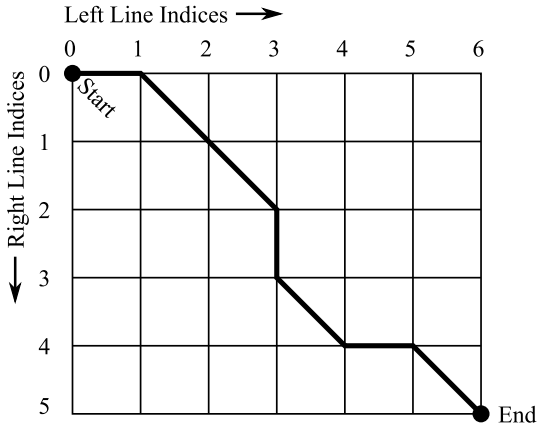
6. If any bin size is nonzero, start from 1. otherwise exit

By processing the bins from largest member count to smallest, it is ensured that the most significant line orientation clusters are not split due to arbitrary choice of bin bounds. Of course, this assumes that line orientations actually form clusters instead of distributing uniformly. However, man-made structures rarely have uniformly distributed edge orientations.

The matching of the line orientation clusters is visualized in Figure 3.17. An important step before the actual computation of the optimal matching by dynamic programming is sorting of the line segments. This is done for the left and right image individually. As formulated in our assumptions, the order of matching pixels along each scanline shall never be reversed. To exploit this assumption, a special line segment sorting step enables the efficient optimal solution of the line matching problem by allowing us to constrain the search space. The remaining problem is the one of uniquely assigning matches between two ordered sequences. This is visualized in Figure 3.17 (b) where only steps to the right, diagonal and down are allowed - each respectively corresponding to ”skip left line”,




(a) Sorted line segments of similar orientation in a stereo frame




(b) Search space of the corresponding matching problem.

Figure 3.17: Stereo line segment matching for segments of similar orientation. (a) Left and right image with indexed line segments (black: matched, red: unmatched). (b) Search plane for dynamic programming with optimal solution path. Diagonal path segments correspond to matches, while vertical and horizontal path segments skip an index match pairing.

"match" and "skip right line". As described in Li (1994), the line sorting works as follows:

1. Sort line segments by their minimum y-coordinate in increasing order and store them in $L_{initial}$.
2. Remove first line segment from $L_{initial}$ and add it to the otherwise empty $L_{result}$.
3. Repeat until $L_{initial}$ is empty:

- Remove first line segment $l$ from $L_{initial}$ and compare against every line $l_k$ in $L_{result}$:

  **if** *ahead_of*$(l, l_k)$, insert $l$ before $l_k$ – otherwise append $l$ to $L_{result}$

- Definition of *ahead_of*$(l_1, l_2)$:

  **if** $l_1$ has no vertical overlap to $l_2$, return *false*

  **else if** the endpoints of $l_1$ are on the left of $l_2$, return *true*

  **else** return *false*

Note, that this sorting algorithm requires that line segments are not intersecting. Such cases have to be resolved in a preprocessing step by splitting one line segment at the intersection point. Figure 3.17 (a) shows the resulting sorting order for an example stereo frame.

Finally, the sorted line segment sets for the left and right image are matched by dynamic programming. Figure 3.17 (b) shows the discrete search plane that is spanned by the line indices. Due to the nonreversal of line segments, we have to find the lowest cost path from the top left to the lower right corner in the search plane by only moving right, down and diagonally (down and right). This special case of dynamic programming is again (as for EBDP) equivalent to the Dijkstra optimal path algorithm (Dijkstra, 1959). Since this method was already outlined in Section 3.4.4, we will only describe how the costs for the different move options were chosen. In contrast to Li (1994), we do not use the similarity of edge gradient magnitudes as a matching criterion. Since gradients depend on the brightness difference between both sides of an edge, occlusion edges can have highly differing gradients in the left and right image. As for EMCBR and EBDP, the matching cost is computed from the minimum of the left and right side SAD matching costs, as defined in Eqn. (3.2). Only the vertically overlapping range is matched. This additional information supports the search by guiding the optimization in the right direction: prohibitively bad matching scores prevent false matches that might otherwise have been assigned when only inferring from the line segment order, gradient and overlap. We will denote this cost with $m(l_i, l_j)$ and use $t_{\mathrm{SAD}}$ as the matching threshold. To maximize the overall vertical overlap for matched lines, we make all costs proportional to the line segment length/overlap, denoted by $\|l_i\|$ and $o(l_i, l_j)$ respectively. The cost for skipping either a line segment in the left image (horizontal move) or right image (vertical move) is penalized by

$C_{\text{skip}}(l_i)$, while the cost for a match between $l_i$ and $l_j$ (diagonal move) is given by $C_{\text{match}}(l_i, l_j)$:

$$C_{\text{skip}}(l_i) = t_{\text{SAD}} \|l_i\| \tag{3.7}$$

$$C_{\text{match}}(l_i, l_j) = t_{\text{SAD}} \left(\|l_i\| + \|l_j\| - 2o(l_i, l_j)\right) + 2o(l_i, l_j)m(l_i, l_j) \tag{3.8}$$

From the final line segment matches, sub-pixel accurate disparities can be extracted. However, for horizontal lines, depth information cannot directly be computed as the epipolar geometry becomes singular. In these cases, a post-processing step can look up non-singular lines in the vicinity of the endpoints of a singular line segment and recover the depth information by interpolation. This approach is valid as long as both enclosing lines from which the depth is interpolated are located on the same surface as the horizontal line.

### 3.5.3 Experimental Results

To the best of the author's knowledge, neither the source code of other small baseline stereo line matching algorithms is available nor do suitable standard benchmark datasets exist. Hence, the evaluation in this section is done with three representative indoor datasets and the described stereo line matching approach is only compared to EMCBR and EBDP for which line segments are extracted with the technique from Section 3.5.1 as a post processing step. The datasets have differing levels of visual complexity, ranging from the very sparse Room dataset, the Corridor dataset with higher noise levels due to low lighting and medium complexity and finally the Big Room dataset with the highest visual complexity of the three. Unfortunately, ground truth depth data is not available. The quantitative analysis will accordingly only include match numbers and the cumulative length of matched line segments and no error statistics. However, since only very few matching errors are not rejected, these numbers can be interpreted as a measure of match success (see Table 3.3).

The stereo datasets were recorded using the stereo camera that we described in Section 2.4 and have a rectified resolution of $640 \times 512$px. The experiments were run on a single core of an Intel Xeon E5-1650 with 3.20GHz. No GPU acceleration was used. While EBDP and EMCBR are parameterized as described in the respective previous sections, the line matching parameterization is chosen

Figure 3.18: Exemplary line match result for the Room sequence. Matching line segments have corresponding colors in the left and right image. Top row: line matching, middle row: EBDP, bottom row: EMCBR.

with the same maximum disparity of 64, $t_{\mathrm{SAD}} = 12$ and 16 angular bins.

As can be seen in Figure 3.18, texture is almost completely absent from the Room sequence and additionally, specularities on the white wall make it challenging to match the affected regions based on neighborhood similarity. As a consequence, EMCBR (the only local technique) struggles and achieves significantly fewer matches (see the first column of Table 3.3 for quantitative results). Line Matching is most successful (largest cumulative matched line length), closely followed by EBDP. However, EBDP (and also EMCBR) takes more than one order of

Figure 3.19: Exemplary line match result for the Corridor sequence. Matching line segments have corresponding colors in the left and right image. Top row: line matching, middle row: EBDP, bottom row: EMCBR.

magnitude longer to compute, see Table 3.4 for the average computation times.

The Corridor sequence has significantly more visible geometry, leading to higher computation times (second column of Table 3.4). Figure 3.19 shows the resulting matched lines for all three algorithms. Again, the proposed line matching technique performs superior to EMCBR and EBDP while having significantly lower computation times (see the second column Table 3.3 for matching statistics).

Finally, the Big Room sequence challenges the tested algorithms with numer-

Figure 3.20: Exemplary line match result for the Big Room sequence. Matching line segments have corresponding colors in the left and right image. Top row: line matching, middle row: EBDP, bottom row: EMCBR.

ous thin free-standing objects and the highest number of matchable line segments, see Figure 3.20. Due to the partial availability of texture, EBDP is able to find more matches in some areas, however line matching achieves a similar cumulative matched line length and again separates itself from EBDP and EMCBR by more than an order of magnitude difference in computation time.

Additionally to all stereo matching times in Table 3.4 one has to add the given line detection times to yield the complete processing time. For line matching, this is the dominating contribution to the total computation time.

Table 3.3: Matching results as average line match count vs. average cumulative matched length

| | Room | | Corridor | | Big Room | |
|---|---|---|---|---|---|---|
| | Count | Length | Count | Length | Count | Length |
| Line Matching | 38 | 4538.2 | 94 | 5120.7 | 132 | 6052.3 |
| EBDP | 40 | 4313.5 | 84 | 4885.0 | 141 | 5971.1 |
| EMCBR | 33 | 3347.3 | 68 | 4149.6 | 112 | 4214.5 |

Table 3.4: Average computation Times of matching algorithms and line detection

| | Room | Corridor | Big Room |
|---|---|---|---|
| Line Matching | 0.9 ms | 4 ms | 5 ms |
| EBDP | 46 ms | 93 ms | 98 ms |
| EMCBR | 20 ms | 38 ms | 49 ms |
| Line Detection | 13 ms | 20 ms | 24 ms |

## 3.6 Discussion

This section presented three different edge matching techniques. EMCBR and EBDP both work directly on the connected chains of edge pixels, while the presented line matching technique uses line segments as matching primitives. All techniques are real-time capable, however the computation times still differ significantly - line matching being by far the fastest due to the lower count of matching primitives.

The local stereo edge matching technique EMCBR uses refinement on connected edge chains to improve the results of purely local stereo matching. We also investigated the consideration of depth discontinuities at the matching location and showed that good results are possible with sparse matching, producing better error rates than most dense algorithms while being computationally less demanding.

The refinement algorithm enforces consistent disparities along edges and is able to reliably interpolate missing ones. The main remaining source for errors and unmatched edges is the quality of the edge detector. Often, object edges are distorted or disrupted by adjacent surfaces of similar intensity. This is especially true for highly textured objects. This situation can be improved by more costly edge detectors that take color, edge biases and scale-space into account. Unfortu-

nately, sophisticated edge detectors would have a severe impact on the execution time.

Building on the results of EMCBR, EBDP is an efficient semiglobal optimization-based edge matching technique, that uses discontinuity penalties to find smooth disparities along edge-segments. The idea behind the support regions is simple, yet efficient and accounts for depth discontinuities and horizontal edge segments. This combination allows retrieving significantly more disparities than EMCBR and PPBSS in challenging setups like the Tsukuba image set. However, the advantage over EMCBR vanishes for image sets with only few horizontal edge-segments.

When ultimately only matched line segments are used as the output, both techniques are comparably wasteful with computational resources since the matching primitives are edge pixels in contrast to far fewer line segments. We have shown that the proposed stereo line matching technique produces results comparable to or better than EBDP while being substantially faster. As a result, this technique is chosen as a building block for all subsequent experiments. Finally, the presented line segment detection is a critical component for stereo line matching and the bigger scope of Line SLAM in general. We found that the approach using Douglas-Peucker polygonization and subsequent line fitting performs significantly better and faster than classic Hough methods.

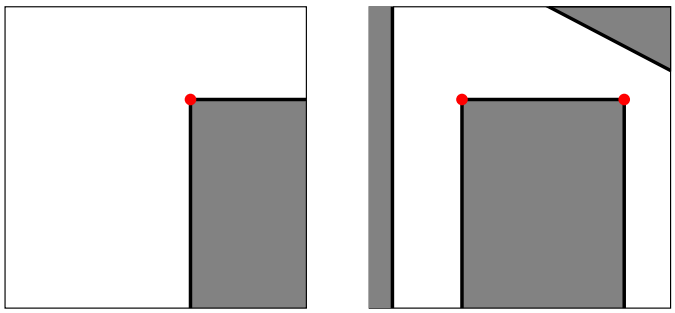
# Chapter 4

# Stereo Visual Odometry Using Lines

This chapter is concerned with stereo visual odometry using line segments as features. The intra-frame edge and line matching techniques from Chapter 3 solve the correspondence problem for known epipolar geometry. In visual odometry, one is interested in the camera motion between distinct frames. Accordingly, in contrast to stereo matching, one has to find correspondences in the face of unknown epipolar geometry. This chapter proposes algorithms to robustly find inter-frame line segment matches and recover the camera motion with high accuracy. This chapter is based on the corresponding publication (Witt and Weltin, 2013).

## 4.1 Introduction

Many visual odometry and navigation systems rely on point features (Davison, 2003; Klein and Murray, 2007; Konolige and Agrawal, 2008; Nistér et al., 2004). This is due to favorable properties like being easily detectable, matchable and their orthogonal gradient that locates them in x- and y-direction. In many environments, these systems function fast and robustly. However, reliable point-features are not always available in sufficient numbers. Untextured three-dimensional objects and environments are prone to be problematic to these kind of systems. Edges on the other hand are interesting image features for a host of robotic applications where textures are sparse. Many man-made structures lack texture while edges are usually still abundantly available. In (Tomono, 2009b), it was exemplarily shown that an edge-based system can prevail where proven point-based algo-



(a) (b)

Figure 4.1: These figures show two scenes, for which point-based algorithms cannot find the required three features to recover camera motion from two stereo frames. In contrast, only two lines are required. (a) The minimal two lines and only one point feature. (b) Five lines, but only two point features.

rithms such as the KLT feature tracker cease to succeed (Shi and Tomasi, 1994). We will go one step further and provide quantitative results of the state-of-the-art stereo odometry algorithm VISO2 (Geiger et al., 2011) for all our test sequences. In the KITTI benchmark (Geiger et al., 2012), this method has proven to be very accurate and one of the fastest feature point based odometry algorithms.

Investigating the most extreme case of scene sparsity, we find that two lines are sufficient to recover the camera motion, whereas at least three points are required in the stereo case. Figure 4.1 depicts scenarios for which point-based methods resign. These setups can be quite common in indoor environments and robust RANSAC-based algorithms usually require even more point correspondences to check for consensus. See Section 2.7 for a short introduction to the RANSAC approach.

Yet, inherent difficulties concerning the use of edges in vision-based navigation exist. By definition, edges divide regions of homogeneous intensity. Accordingly, their distinctiveness can be limited. Finding an individual edge correspondence between two images without known camera motion is not simple and sometimes not even possible in a reliable manner (Meltzer and Soatto, 2008). However, with a calibrated stereo camera, the known epipolar geometry simplifies intra-frame matching as we saw in the preceding chapter. With prior stereo matching, we are left with the problem of finding the rigid transformation that aligns the largest number of lines between the two stereo frames. This can make an appearance-based inter-frame edge matching unnecessary.

Finding the six motion parameters between two stereo edge frames is still challenging, since the search space contains local minima. This registration problem can be approached with the well-known Iterative Closest Point (ICP) algorithm in a 2D-3D variant (Lowe, 1991; Tomono, 2009b). The possibility of registration failures requires a robust fallback method, though. A short explanation of ICP is provided in Section 2.6. While ICP has been shown to work, point sequences are not a very compact representation for intensity edges, especially when it is expected that many edges are partially or completely straight. This leads to significantly higher computation times than for most feature point based methods. For example, a square is fully described with its four corners (which are good feature points) or the lines that make up the contour. On the other hand, the number of edge points in image space is much larger, while not necessarily carrying additional information.

The deduction of an analog "Iterative Closest Line" algorithm is not as straight forward as one might think at first, though. In contrast to points, for lines, no definitive scalar metric for closeness exists. Additionally, the detection of line endpoints is usually unreliable. An edge that was detected as one long line in a given image might be split into several shorter segments in another. Thus, a 1-to-1 matching would have to discard all of those segments but one.

In this chapter, a method called Iterative Closest Multiple Lines (ICML) is proposed to efficiently register lines with a one-to-many matching. This remedies the problems associated with the selection of a single best matching line. The registration rate even improves over the much more costly ICP. For robustness, a measure for automatic registration failure detection and a sample consensus based fallback solution are proposed. In contrast to RANSAC, the hypothesis selection is deterministic to make efficient use of computational resources.

## 4.2 Related Work

Several monocular SLAM systems that utilize edge segments as features have been proposed in recent years (Eade and Drummond, 2009; Klein and Murray, 2008; Smith et al., 2006). They try to establish an appearance-based matching between frames. However, only a subset of all possible edges is considered to allow for successful matching. In very sparse environments, this can lead to problems.

In (Lowe, 1991), known three-dimensional models are matched to edges that are detected in monocular images. The ICP algorithm is employed. It iteratively aligns the reprojected model edges with the detected image edges.

An extension of this technique to 3D SLAM using stereo edge points was presented in (Tomono, 2009b). Here, edge points are first matched within the stereo frame and afterwards the rigid transformation between two frames is iteratively computed by optimizing the reprojection error. In distinction to this work, the author uses individual edge points and employs a variant of the ICP algorithm to align the stereo frames. This direct method can suffer from local minima during the optimization, which is why the author introduces SIFT (Scale-Invariant Feature Transform, see (Lowe, 2004)) descriptors along edge points in (Tomono, 2010) for failure recovery.

A system using straight lines was proposed in (Chandraker et al., 2009). Dense stereo matching for intra-frame and multi-level Lucas-Kanade optical flow for inter-frame matching are employed. With the significantly reduced search space, RANSAC is used to find the best transformation by computing a rigid transformation hypothesis, built from two or three matched lines. Accordingly, the line matching and motion recovery are separate steps in this approach. A consequence of this appearance-based matching is, that the success depends on the performance of the optical flow algorithm.

## 4.3 Motion Reconstruction Using Lines

We represent line segments by their endpoints, which is convenient for reconstruction and reprojection. The respective equations were already reviewed in Section 2.2 and 2.3. Motion reconstruction is achieved through registration of the previously reconstructed three-dimensional lines with the currently detected (yet two-dimensional) image lines. Although it would be possible to do 3D-3D matching by reconstructing the current image lines prior to registration, a naive approach would introduce unnecessary errors as lines with large Euclidean errors would dominate the optimization (i.e. lines that are far away). Optimizing the reprojection error (3D-2D optimization) naturally accounts for the type of uncertainty that is inherent to image space measurements. In addition, the registration of monocular imagery with a 3D model is equally possible, as done in (Lowe,

1991). The minimum number of line matches is three in this case, while stereo matching requires only two nonparallel lines to recover the 6-DOF motion.

### 4.3.1 Line Reprojection Error

The reprojection for 3D points is defined by the intrinsic and extrinsic camera parameters. While the rotation matrix $\mathsf{R}_k$ and the translation vector $\mathbf{t}_k$ make up the unknown six degrees of freedom of the camera that we would like to recover at frame $k$, the intrinsic parameters, again, are known from prior calibration. While it is rather straightforward to define the reprojection error between two points (or a point and a line) as their minimum Euclidean distance, such a clear statement for the error between two lines is not possible. This stems from the fact that the error between 2D lines is two-dimensional (if endpoint locations are not considered). One possibility to parameterize this error is by representing one of the lines by its end points and computing the perpendicular distance to the other line for each.

We characterize the match between the $j^{th}$ reprojected line $\mathbf{l}^j(\mathbf{x})$ and the $i^{th}$ image line $\hat{\mathbf{l}}^i$ by their overlap $l(\mathbf{l}^j(\mathbf{x}), \hat{\mathbf{l}}^i)$ and mean distance $d_{ij}(\mathbf{x})$ in image space, where $\mathbf{x}$ represents the sought-after camera pose parameters that influence the reprojection.

$$d_{ij}(\mathbf{x}) = \frac{1}{2}\left(|d_{P_1}(\mathbf{l}^j(\mathbf{x}), \hat{\mathbf{l}}^i)| + |d_{P_2}(\mathbf{l}^j(\mathbf{x}), \hat{\mathbf{l}}^i)|\right) \tag{4.1}$$

The terms $d_{P_1}(\mathbf{l}^j(\mathbf{x}), \hat{\mathbf{l}}^i)$ and $d_{P_2}(\mathbf{l}^j(\mathbf{x}), \hat{\mathbf{l}}^i)$ denote the perpendicular distances of the end points of image line $\hat{\mathbf{l}}^i$ to the reprojected line $\mathbf{l}^j(\mathbf{x})$. We will omit the dependency of the overlap $l(\mathbf{l}^j(\mathbf{x}), \hat{\mathbf{l}}^i)$ on the transformation parameters $\mathbf{x}$ by writing $l_{ij}$ to convey that this value is kept as a weighting constant instead of participating in the optimization (although it is updated in between iterations). The product of $d_{ij}(\mathbf{x})$ and $l_{ij}$ equals the overlapping area between the two lines. Accordingly, the mean pixel error (ME) for all matching lines can be formulated in the following way

$$\mathrm{ME}(\mathbf{x}) = \frac{\sum_{i,j} d_{ij}(\mathbf{x}) l_{ij}}{\sum_{i,j} l_{ij}}. \tag{4.2}$$

Another natural choice would be the weighted mean squared error (MSE):

$$\mathrm{MSE}(\mathbf{x}) = \frac{1}{2}\frac{\sum_{i,j}\left(d_{P_1}(\mathbf{l}^j(\mathbf{x}), \hat{\mathbf{l}}^i)^2 + d_{P_2}(\mathbf{l}^j(\mathbf{x}), \hat{\mathbf{l}}^i)^2\right) l_{ij}}{\sum_{i,j} l_{ij}} \tag{4.3}$$

The weighting with the overlap $l_{ij}$ is introduced to reflect the significance of a match in the cost function. Accordingly, a number of small matching lines have the same influence on the result as one larger match with equal cumulative overlap.

While the MSE is a common, and usually sound, choice in optimization problems, the mean pixel error includes absolute value functions in $d_{ij}(\mathbf{x})$, which are problematic for optimization. However, if the absolute value terms are replaced by Huber functions $h(z)$, the equation becomes differentiable while keeping its linear behavior for larger errors.

$$h(z) = \begin{cases} \frac{1}{2}z^2 & |z| < k \\ k|z| - \frac{1}{2}k^2 & else \end{cases} \tag{4.4}$$

Using values like $k = 1\text{px}$ limits the region with quadratic behavior to small errors.

### 4.3.2 Iterative Closest Multiple Lines (ICML) Algorithm

The Iterative Closest Point (ICP) algorithm functions by finding the minimum distance to a model (which would be lines or edge point sequences in this case) for each point and improving this distance with gradient-based optimization. This is repeated until convergence. The publications (Lowe, 1991) and (Tomono, 2009b) use this approach for the monocular and stereo case, respectively.

Due to the two-dimensional reprojection error it is not clear what the closest line to another line is, as shown in Figure 4.2. Of course we can define a matching score and utilize e.g. the mean pixel distance and overlap between two lines as a criterion, but we will see why this is unfavorable.

Instead of finding a ”best” match among the model lines (1-to-1 matching), the ICML algorithm considers all lines in the neighborhood (1-to-N matching). For a line to be considered in the neighborhood it has to fulfill the following criteria: $d_{ij} < d_{\max}$, $l_{ij} > 0$ and $\alpha_{ij} < \alpha_{\max}$. The term $\alpha_{ij}$ refers to the angle between both

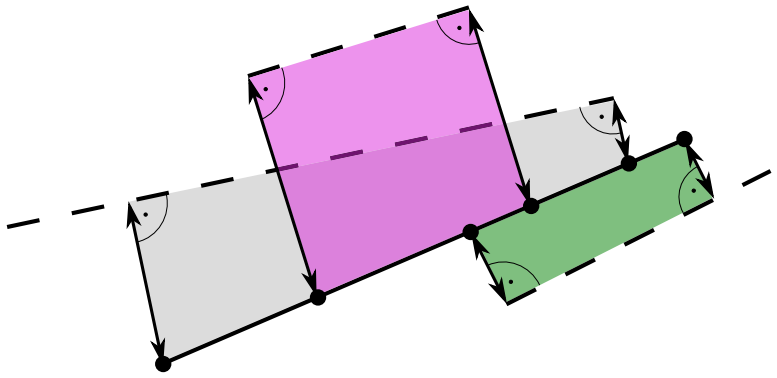


Figure 4.2: Multiple reprojected lines (dashed) near a line in the current frame (solid). Which one is the "best" match? Since the matching error is multidimensional, a scalar "best" metric as for the distance between points does not exist.

lines in the image plane, while $\alpha_{\max}$ and $d_{\max}$ are free parameters. During all stages we use $\alpha_{\max} = 12^\circ$.

### Optimization of the Camera Pose

The resulting optimization problem has two rows for each line match per view. For stereo optimization, the matches in the right image are added analogously. With the weighting matrix $\mathsf{W} = diag(l_{ij})$ we minimize the following quadratic cost function in the case of MSE

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{e}(\mathbf{x})^\top \mathsf{W}\mathbf{e}(\mathbf{x}) \tag{4.5}$$

$$\mathbf{e}(\mathbf{x}) = \left(..., \; d_{P_1}(\mathbf{l}^j(\mathbf{x}), \hat{\mathbf{l}}^i), \; d_{P_2}(\mathbf{l}^j(\mathbf{x}), \hat{\mathbf{l}}^i), \; ...\right)^\top. \tag{4.6}$$

Note, that the reprojection error $\mathbf{e}(\mathbf{x})$ measures the perpendicular distance (in image space) between the endpoints of the detected line segment $\hat{\mathbf{l}}^i$ and the reprojected line segment $\mathbf{l}^j(\mathbf{x})$. In each iteration of the optimization, we want to solve for the incremental transformation $\mathbf{x}$ which determines the location of all reprojected line segments.

To minimize the Huber ME we use $h(\mathbf{z})$ as an element-wise Huber function

and yield

$$f_h(\mathbf{x}) = \frac{1}{2}\mathsf{W}\mathbf{e}_h(\mathbf{x}) \tag{4.7}$$

$$\mathbf{e}_h(\mathbf{x}) = h(\mathbf{e}(\mathbf{x})). \tag{4.8}$$

These functions can finally be minimized with Levenberg-Marquardt optimization when the incremental motion update $\mathbf{x}$ and the error Jacobian J are defined

$$\mathbf{x} = \left(\Delta t_x, \Delta t_y, \Delta t_z, \phi_x, \phi_y, \phi_z\right)^\top \tag{4.9}$$

$$\mathsf{J} = \frac{\partial \mathbf{e}(\mathbf{x})}{\partial \mathbf{x}}. \tag{4.10}$$

Note, that $f(\mathbf{x})$ and $f_h(\mathbf{x})$ are scaled versions of Eq. (4.2) and (4.3) respectively. Since the denominator with the cumulative overlap is constant during one iteration, we omit it for the sake of readability.

The solution vector $\mathbf{x}$ constitutes the incremental transformation that we want to calculate. Besides a translation vector $\Delta\mathbf{t} = \left(\Delta t_x, \Delta t_y, \Delta t_z\right)^\top$ it contains an incremental rotation which relates to the following linearized rotation matrix

$$\Delta\mathsf{R} = \begin{bmatrix} 1 & -\phi_z & \phi_y \\ \phi_z & 1 & -\phi_x \\ -\phi_y & \phi_x & 1 \end{bmatrix}. \tag{4.11}$$

With the camera transformation for $\boldsymbol{c}_{k-1}$ given in (4.12), we can substitute into the delta update equation (4.13) and reorder the terms so that we can deduce the updated transformation

$$\boldsymbol{c}_{k-1} = \mathsf{R}_{k-1}^\top(\mathbf{w} - \mathbf{t}_{k-1}) \tag{4.12}$$

$$\mathbf{c}_k = \Delta\mathsf{R}\mathbf{c}_{k-1} + \Delta\mathbf{t} \tag{4.13}$$

$$= \Delta\mathsf{R}\mathsf{R}_{k-1}^\top(\mathbf{w} - \mathbf{t}_{k-1}) + \Delta\mathbf{t} \tag{4.14}$$

$$= \underbrace{\Delta\mathsf{R}\mathsf{R}_{k-1}^\top}_{\mathsf{R}_k^\top}(\mathbf{w} - \underbrace{(\mathbf{t}_{k-1} - (\Delta\mathsf{R}\mathsf{R}_{k-1}^\top)^\top\Delta\mathbf{t})}_{\mathbf{t}_k}). \tag{4.15}$$

Accordingly, the pose update equations from frame $k-1$ to $k$ are

$$\mathsf{R}_k = \mathsf{R}_{k-1}\Delta\mathsf{R}^\top, \tag{4.16}$$

$$\mathbf{t}_k = \mathbf{t}_{k-1} - \mathsf{R}_{k-1}\Delta\mathsf{R}^\top\Delta\mathbf{t}. \tag{4.17}$$

**Image Space Jacobian of an Incremental Transformation**

The mentioned Jacobian describes the relation between the reprojection error $\mathbf{e}(\mathbf{x})$ to the delta transformation $\Delta\mathsf{R}, \Delta\mathbf{t}$ (which is packed into the vector $\mathbf{x}$). Since the measured location of features is constant for a given frame, the Jacobian only depends on the line reprojection terms. This section derives the image space Jacobian with respect to a delta transformation $\mathbf{x}$. For visual clarity we will use the indices 0 and 1 in place of $k-1$ and $k$. Since we always linearize around the current transformation, we can express the new camera space coordinates $\mathbf{c}_1$ in terms of the current camera space coordinates $\mathbf{c}_0$ and the delta transformation $\mathbf{x}$, as we saw in Eq. (4.13). Substituting (4.11) into (4.13) yields

$$\mathbf{c}_1 = \begin{bmatrix} c_{0x} - \phi_z c_{0y} + \phi_y c_{0z} + \Delta t_x \\ \phi_z c_{0x} + c_{0y} - \phi_x c_{0z} + \Delta t_y \\ -\phi_y c_{0x} + \phi_x c_{0y} + c_{0z} + \Delta t_z \end{bmatrix}. \tag{4.18}$$

For the right camera, we get

$$\mathbf{c}_{1r} = \mathbf{c}_1 - \mathbf{b} \tag{4.19}$$

$$\mathbf{c}_1 = (c_{1x}, c_{1y}, c_{1z})^\top \tag{4.20}$$

$$\mathbf{c}_{1r} = (c_{1x} - b, c_{1y}, c_{1z})^\top. \tag{4.21}$$

The derivatives of $\mathbf{c}_1$ with respect to the delta transformation are

$$\frac{\partial \mathbf{c}_1}{\partial(\Delta t_x, \Delta t_y, \Delta t_z)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.22}$$

$$\frac{\partial \mathbf{c}_1}{\partial(\phi_x, \phi_y, \phi_z)} = \begin{bmatrix} 0 & c_{0z} & -c_{0y} \\ -c_{0z} & 0 & c_{0x} \\ c_{0y} & -c_{0x} & 0 \end{bmatrix}. \tag{4.23}$$

Using the product rule we can derive the projection equations for the left and right camera and get

$$\frac{\partial u_l}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}} \frac{f c_{1x}}{c_{1z}} = \frac{f}{c_{0z}} \frac{\partial c_{1x}}{\partial \mathbf{x}} - \frac{f c_{0x}}{c_{0z}^2} \frac{\partial c_{1z}}{\partial \mathbf{x}} \tag{4.24}$$

$$\frac{\partial u_r}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}} \frac{f c_{1rx}}{c_{1z}} = \frac{f}{c_{0z}} \frac{\partial c_{1x}}{\partial \mathbf{x}} - \frac{f(c_{0x} - b)}{c_{0z}^2} \frac{\partial c_{1z}}{\partial \mathbf{x}} \tag{4.25}$$

$$\frac{\partial v}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}} \frac{f c_{1y}}{c_{1z}} = \frac{f}{c_{0z}} \frac{\partial c_{1y}}{\partial \mathbf{x}} - \frac{f c_{0y}}{c_{0z}^2} \frac{\partial c_{1z}}{\partial \mathbf{x}} \,. \tag{4.26}$$

Finally substituting (4.22) and (4.23) for the remaining derivatives yields the Jacobian terms listed in Table 4.1.

Table 4.1: Partial derivatives of the left and right image coordinates with respect to the delta transformation.

| | $\Delta t_x$ | $\Delta t_y$ | $\Delta t_z$ | $\phi_x$ | $\phi_y$ | $\phi_z$ |
|---|---|---|---|---|---|---|
| $\frac{\partial u_l}{\partial \mathbf{x}}$ | $\frac{f}{c_{0z}}$ | 0 | $-\frac{f c_{0x}}{c_{0z}^2}$ | $-\frac{f c_{0x} c_{0y}}{c_{0z}^2}$ | $f\left(1 + \frac{c_{0x}^2}{c_{0z}^2}\right)$ | $-\frac{f c_{0y}}{c_{0z}}$ |
| $\frac{\partial u_r}{\partial \mathbf{x}}$ | $\frac{f}{c_{0z}}$ | 0 | $-\frac{f(c_{0x}-b)}{c_{0z}^2}$ | $-\frac{f(c_{0x}-b) c_{0y}}{c_{0z}^2}$ | $f\left(1 + \frac{c_{0x}(c_{0x}-b)}{c_{0z}^2}\right)$ | $-\frac{f c_{0y}}{c_{0z}}$ |
| $\frac{\partial v}{\partial \mathbf{x}}$ | 0 | $\frac{f}{c_{0z}}$ | $-\frac{f c_{0y}}{c_{0z}^2}$ | $-f\left(1 + \frac{c_{0y}^2}{c_{0z}^2}\right)$ | $\frac{f c_{0x} c_{0y}}{c_{0z}^2}$ | $\frac{f c_{0x}}{c_{0z}}$ |

### ICML Optimization Scheme

While for a given $d_{\max}$, the scheme of alternate matching and optimization is similar to regular ICP, we overlay another loop to control this matching parameter and implement a coarse-to-fine strategy. We iteratively reduce $d_{\max}$ from a large value $d_{\text{init}}$ down to a fine value $d_{\text{final}}$ to achieve convergence from a wider range of configurations while not sacrificing accuracy when the lines are correctly aligned. While $d_{\text{init}}$ and $d_{\text{final}}$ are both free parameters, we kept $d_{\text{final}} = 1\text{px}$ during all experiments. The parameter $d_{\text{init}}$ should be chosen large enough, so that the correctly matching lines are among the matches. The algorithm is listed in the following:

1. Set $d_{\max} = d_{\text{init}}$.
2. For each image line, find all model lines with $d_{ij} < d_{\max}$, $l_{ij} > 0$ and $\alpha_{ij} < \alpha_{\max}$.
3. With all found matches, do Levenberg-Marquardt until convergence.
4. If $d_{\max} \leq d_{\text{final}}$ exit, otherwise $d_{\max} = d_{\max}/2$ and go to step 2.

Optionally, one can add an offset in the order of $d_{\max}$ to each $l_{ij}$ to allow lines to influence the optimization that do not actually overlap, but are within the axial range of this offset. Accordingly, we need to measure the axial gap between lines as negative overlap. This can be of importance for fast simultaneous movements in horizontal and vertical image direction when no large lines are detected. However, in our trials this was not necessary.

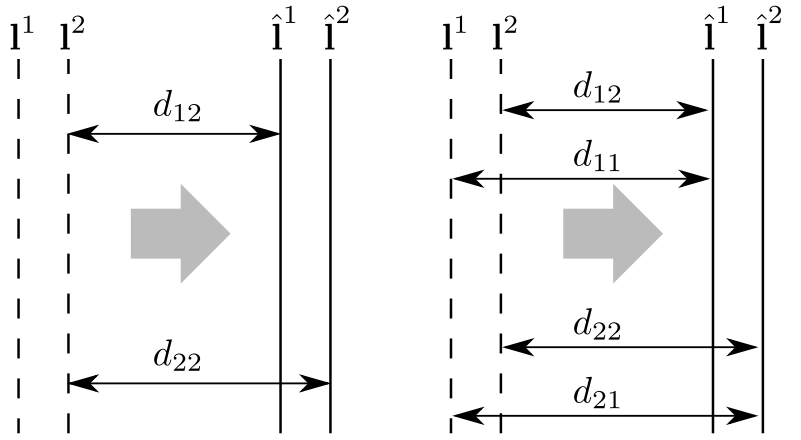
Due to the potentially large number of matched lines (especially for sizeable $d_{init}$), ICML can be biased towards one-sided groupings of parallel lines within match range. A typical configuration of this kind can be found when looking down corridors (see top left image of Figure 4.6). To remedy this affinity, we sort all matches of each line by their mean distance $d_{ij}$. Subsequently, the matches are removed starting from the largest $d_{ij}$ until the cumulative overlap of the remaining matched lines is smaller than $\mu_{\text{OM}}|\hat{\mathbf{l}}^i|$. While $|\hat{\mathbf{l}}^i|$ is the length of the image line that is matched, $\mu_{\text{OM}}$ is a tuning parameter that controls the amount of "overmatching". For example, when $\mu_{\text{OM}} = 3$ the cumulative overlap of all matched lines is restricted to three times the image line length. Accordingly, three lines would be matched if the overlap was 100% each. In the general case, a number $N \geq 3$ is matched for $\mu_{\text{OM}} = 3$ if enough lines are found in the $d_{\max}$ neighborhood. The introduction of sorting usually has no measurable influence on the optimization time.

### 4.3.3 Evaluation of Common Configurations

This section will investigate the behavior of ICP and ICML in common configurations. In the presented cases with infinite parallel lines, ICP and ICML with 1-to-1 matching (i.e. $\mu_{\text{OM}} = 0$) are equivalent. We will refer to this special case of ICML where only the line with the smallest $d_{ij}$ is matched as Iterative Closest

Line (ICL). For simplicity, we look at one dimension of the optimization problem, which is the horizontal displacement in this case.

Figure 4.3 shows a general configuration with parallel lines for which both 1-to-1 and 1-to-N matching are applied. One can see, that 1-to-1 matching is asymmetrical. So, if we iterated over the reprojected lines to find the best matches instead, we would get a different cost function. With 1-to-N matching, this becomes symmetrical (for $\mu_{\text{OM}} \to \infty$), since all mutual distances are included.



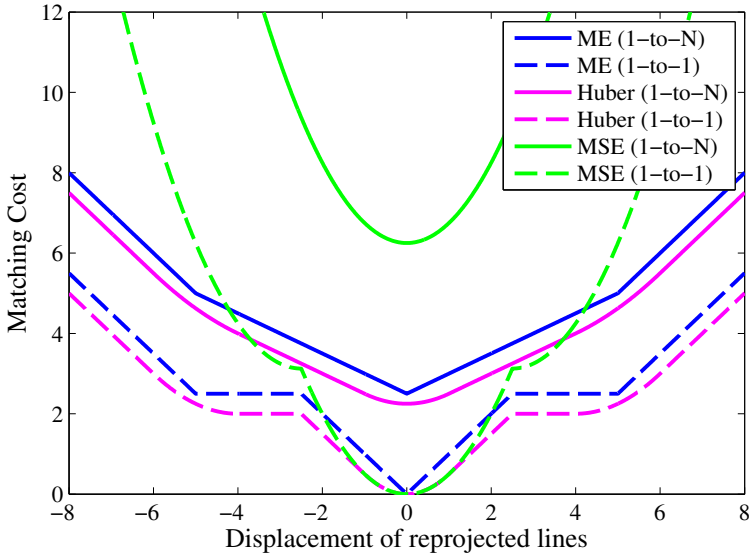

(a) The closest line is matched, which is equivalent to ICP/ICL for infinite parallel lines (1-to-1 matching).

(b) All close lines are matched, resulting in a symmetrical matching (1-to-N matching).

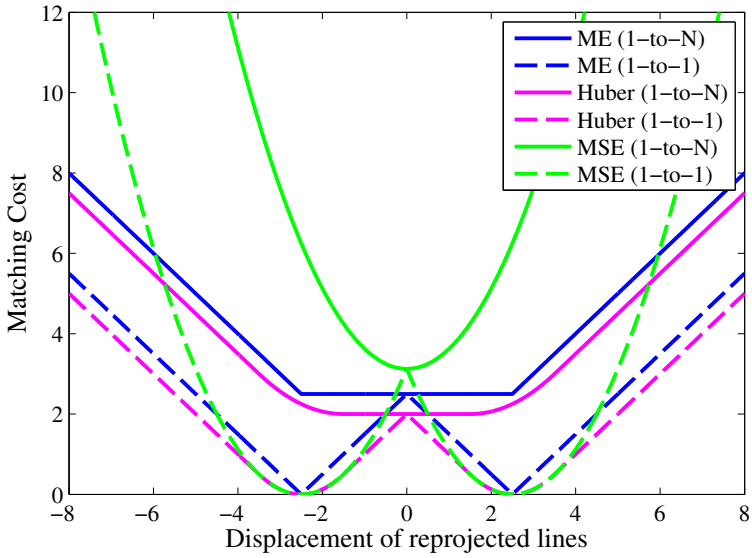
Figure 4.3: These figures depict the 1-to-1 and 1-to-N matching methods. The dashed lines in (a) and (b) illustrate reprojected lines, while the solid lines are image lines of the current frame. The lines are parallel and have a horizontal spacing of five units. In (c) and (d), the mean error (ME), the Huber mean error and the mean squared error (MSE) are given for two scenarios. One can see that 1-to-N matching (solid lines) has fewer local minima and plateaus for these common cases.

### Matching with Successful Line Detection

We will first review the ideal case, when all lines have successfully been detected. The corresponding cost function is shown in Figure 4.4(a). First, we find that all cost functions are strictly decreasing within a displacement of less than half the line spacing. While with 1-to-N matching, the function stays strictly decreasing, with 1-to-1 matching we can locate plateaus on either side of the global minimum




(a) Cost function comparison when all lines were successfully detected.




(b) Cost function comparison for the case when $\hat{\mathbf{l}}^2$ is not detected.

Figure 4.4: These figures depict the 1-to-1 and 1-to-N matching methods. The dashed lines in (a) and (b) illustrate reprojected lines, while the solid lines are image lines of the current frame. The lines are parallel and have a horizontal spacing of five units. In (c) and (d), the mean error (ME), the Huber mean error and the mean squared error (MSE) are given for two scenarios. One can see that 1-to-N matching (solid lines) has fewer local minima and plateaus for these common cases.

for ME and Huber. This section corresponds to the displacement for which one reprojected line is matched to both image lines and the distances have opposite signs. If no other edge matches can drag the optimization over such a plateau, the optimization is stuck. So for the optimization to converge, the maximal tolerable displacement is half the line spacing (both in image space). In many practical situations, this does not hinder optimization, though.

**Matching with Partial Line Detection**

When lines are only detected partially, the cost function changes. Consider the same configuration, except that $\hat{\mathbf{l}}^2$ is not detected. The resulting cost functions are plotted in Figure 4.4(b). With 1-to-1 matching we see two minima. This actively hinders successful optimization even if other lines have been matched correctly. In the case of 1-to-N matching on the other hand, a plateau permits effortless transition between both possibilities in this ambiguous setup for ME and Huber. Of course, other matches are needed to drive the optimization in the right direction, but even a correct line match between small line segments can provide the correct direction. When $d_{\max}$ is iteratively reduced, the line is finally only associated with one of the reprojected lines. When this is the case, the cost function has a unique minimum which supports an accurate alignment. In some scenarios, the MSE can be problematic for this case, because it tries to push the solution to the middle of both matches. However, in the sequences that we tested, line detection was reliable so that those situations were rare. When the line detector or the stereo matching algorithm perform unreliably, this has been found to make a difference.

### 4.3.4 Registration Failure Detection

Although optimizations most often converge to the global minimum with 1-to-N matching, as with any gradient based technique, local minima and other failures cannot be ruled out. In order for other more robust and costly techniques to take over, a measure for failure detection is required. In (Tomono, 2010), the ratio of matched and detected edge points is used for failure detection. We use a similar measure by taking the ratio of the cumulative matched line length that was calculated with $d_{\max} = d_{\text{final}}$ and the minimum of the cumulative lengths

of all detected image lines and reprojected lines. This ratio must be larger than $M_{\min}$.

$$M_k = \frac{\sum_{i,j} l_{ij}}{\min\left(\sum_i |\hat{\mathbf{l}}^i|, \sum_j |\mathbf{l}^j(\mathbf{x})|\right)} > M_{\min} \tag{4.27}$$

We further test whether the mean error $\mathrm{ME}(\mathbf{x})$ exceeds the threshold $G_{\max}$. However, sometimes these criteria alone are not a sufficient indicator whether we can trust the solution. We found it necessary to include a measure which indicates whether the solution is stable. Imagine a scene with mostly parallel lines. In such a case, we can find a transformation which easily fulfills both of the other criteria. However, this solution is not unique. No reliable line match constrains the camera motion to not slide in the direction of those parallel lines. The measure that we propose here is an indicator for the stability of the solution. For this, we determine the directional cumulative matched line lengths $l_h$, $l_{diag1}$, $l_{diag2}$ and $l_v$ for horizontal ($|\alpha_i| < 22.5°$), diagonal ($22.5° \leq \alpha_i < 67.5°$ and $-22.5° \geq \alpha_i > -67.5°$) and vertical ($|\alpha_i| \geq 67.5°$) lines to gain knowledge of how the line directions are distributed. The angle $\alpha_i$ is the line orientation in image space. If three of these lengths are small we have an unstable solution. We desire a minimum diversity in line orientations to trust the solution. For this, $l_h$, $l_{diag1}$, $l_{diag2}$ and $l_v$ are sorted and assigned to $l_1 \leq l_2 \leq l_3 \leq l_4$. The stability criterion that we impose is the following

$$L_k = l_1 + l_2 + l_3 > L_{\min}. \tag{4.28}$$

Accordingly, the sum of the lesser three directional cumulative line lengths must be at least $L_{\min}$ long. This does not constrain the ratio of the lengths, but ensures an absolute minimum orientation diversity among the matched lines.

### 4.3.5 Robust Sample Consensus Matching

When a registration failure was detected, a more robust algorithm is needed to circumvent a possible local minimum. In contrast to edge point matching, with lines, the geometry information is significantly more compact which makes a hypothesize-and-test scheme tractable. Since we are not building a map at this point, we are interested in a fallback solution for incremental motion that effi-

ciently provides good starting values for a subsequent refinement with ICML.

Instead of assigning line match hypotheses purely randomly (like RANSAC), for incremental motion we can restrict the matching possibilities to the coarse initial line matching with $d_{\max} = d_{\text{init}}$. Further, we cluster the line matches in horizontal and vertical ones (again with the image space orientation $\alpha$) and sort them by their overlap. In the last section, we already discussed the necessity of diverse line orientations to yield a robust solution.

Of both directional clusters, only $N_{\text{SAC}}/2$ line matches with the longest overlap are considered to maximize the line orientation diversity. Then, we iterate over all $N_{\text{SAC}}$ matches and form hypotheses of two line matches for all possible combinations. Accordingly, up to

$$N_H = \sum_{i=1}^{N_{\text{SAC}}} i = \frac{N_{\text{SAC}}(N_{\text{SAC}}+1)}{2} \tag{4.29}$$

hypotheses are formed. Recall, that two nonparallel line matches are sufficient to calculate the 6 degrees of freedom (DOF) in the stereo case. Consequently, we disregard hypotheses whose line orientations are too similar. We employ a threshold of 45° by which orientations have to differ at least. For the remaining hypotheses we do the following:

1. Calculate 6 DOF transformation $\mathbf{x}_H$ for the two lines of the hypothesis with gradient descent (fixed matching).
2. If $\text{ME}(\mathbf{x}_H) > G_{\text{SAC}}$ drop hypothesis and go to 1).
3. Compute matching with all lines and $d_{\max} = d_{\text{SAC}}$ to measure consensus. Save cumulative line overlap.
4. Optional: Do an early-out test with the measures from Section 4.3.4 to speed up the algorithm in some cases.

The hypothesis with the largest cumulative line overlap is chosen as the winner. Afterwards, the solution is refined with ICML and a comparably small $d_{\text{init}} = 2d_{\text{SAC}}$. We parameterized the sample consensus (SAC) algorithm with $N_{\text{SAC}} = 60$, the maximum initial hypothesis error $G_{\text{SAC}} = 0.2$px, and the inlier threshold $d_{\text{SAC}} = 4$px.

## 4.4 Experimental Results

To benchmark the presented algorithm in realistic environments, we recorded image sequences containing different visual complexity with the stereo camera that we introduced in Section 2.4. All sequences were recorded hand-held and have a resolution of 640×512 pixels. The Corridor sequence consists of 532 stereo frames over a total travel distance of approximately 48m. The sequence is noisy, due to low lighting and reflections in the ceiling and has a medium amount of visual features. A visually complex environment is tested with the Big Room sequence. It contains many three-dimensional objects and texture. Finally, in the Room sequence, the camera is moved through a mostly non-textured scene with only very few visual corners, which makes it specifically hard for feature point based systems. To evaluate the accuracy, the ground truth end positions were computed with bundle adjustment by matching several frames in the beginning and end of the sequences by hand (manual loop closure).

For all sequences, the parameterization was left unchanged after being determined empirically. The maximum disparity for stereo matching was set to 140px. The optimization used $\mu_{\text{OM}} = 2.5$, $d_{\text{init}} = 64\text{px}$ and $d_{\text{final}} = 1\text{px}$. The registration failure detection was parameterized with $L_{\text{min}} = 100\text{px}$, $G_{\text{max}} = 0.7$ and $M_{\text{min}} = 0.4$. The experiments were run on a single core of an Intel Core i7 with 2.8 GHz. No GPU acceleration was used.

The ICP implementation uses the same coarse-to-fine scheme and parameterization as ICML. The edge points for ICP are extracted from the same refined lines that ICML uses in these experiments. Recall, that ICL refers to a special case of ICML where $\mu_{\text{OM}} = 0$ and only the lines with the smallest $d_{ij}$ are used to build a 1-to-1 matching. In Table 4.2, $\text{ICML}_h$ denotes the use of the Huber cost function $f_h(\mathbf{e})$ instead of the MSE. Since the Huber function becomes quadratic near its minimum, the final accuracy is the same as for MSE, if the same lines are matched. Accordingly, we only compare the registration failure rates between the two. To put our results in perspective with state-of-the-art feature point based visual odometry methods, we include VISO2 in the comparison (Geiger et al., 2011).

Figure 4.5 shows two characteristic frames from the Room sequence along with a trajectory plot. Very few point features are found, which leads to numerous failures and an unusable trajectory with VISO2. While it generally performs very well for its algorithm category, it cannot succeed in these very sparse cases

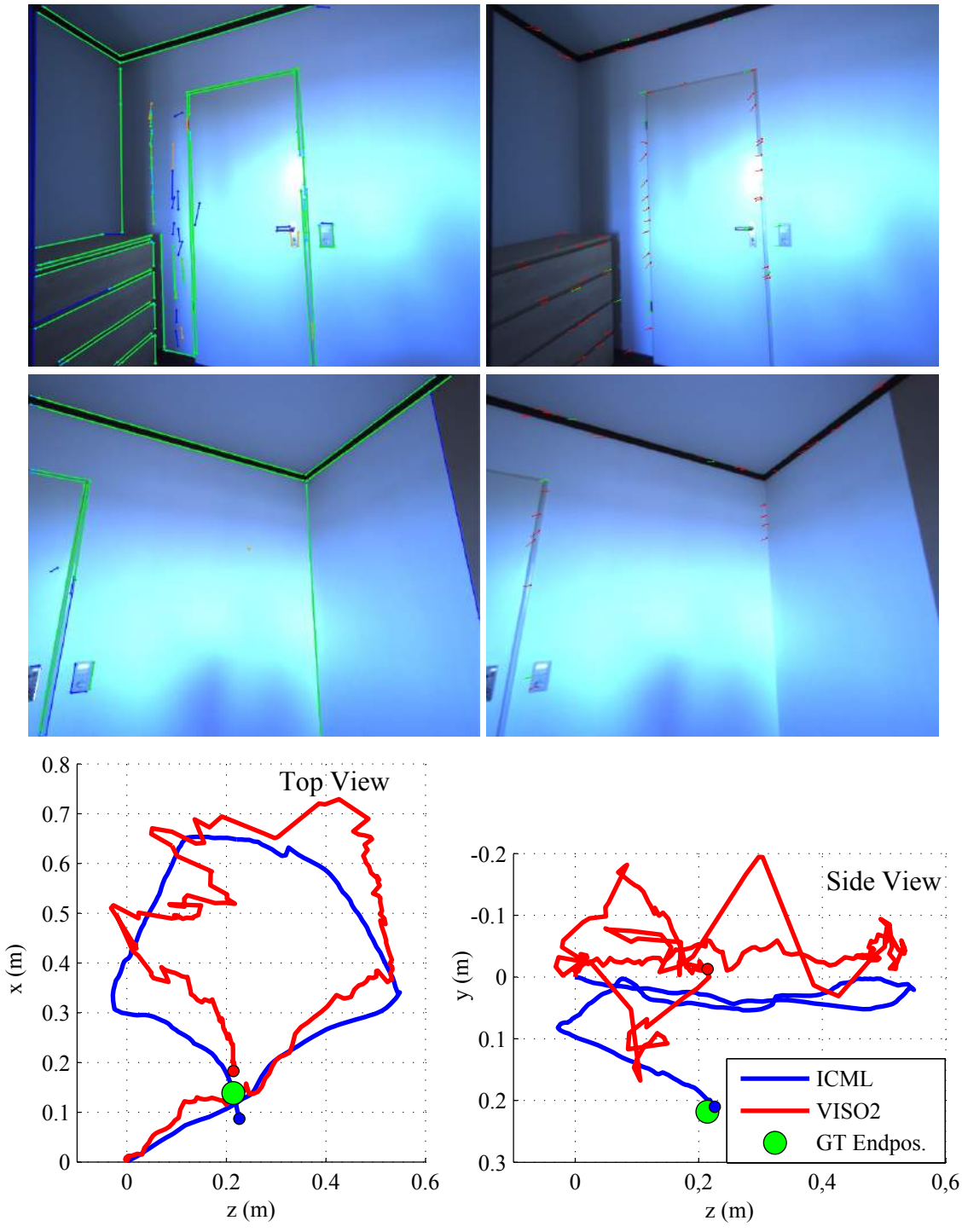



Figure 4.5: The images show frames of an indoor Room sequence for which ICML (left column) successfully computes odometry, while VISO2 (right column) fails to recover the camera motion in many frames. Even the matched feature points (green) are often inaccurate, when they are not located on corners. The result is a zig-zag trajectory with large jumps. ICML achieves an accuracy of 2.5% on this short trajectory of 2.1m length.

(we used the full resolution mode to maximize the chance to find features). ICML on the other hand successfully recovers the motion. Since this scene is generally easy to register for edge-based systems, ICP and ICL also complete without failures and achieve a similar accuracy as ICML (they are omitted in the plot for clarity). However, the stereo matching is very challenging for a number of frames, since horizontal lines have to be recovered in 3D to succeed. As this can be virtually infeasible for lines that do not have locatable features at both ends, we opted for the following strategy: for small numbers of detected lines, we do not only match stereo lines with ICML, but a mixture with all remaining image lines - thus simultaneous stereo and monocular optimization. Additionally, we remember 3D lines from previous frames, as long as they get matched to at least one image line in consecutive frames. For easy cases the horizontal line recovery step can directly compute the desired disparities.

Table 4.2: Registration failure rates for the Corridor and Room sequences. Sample consensus fallback rates are bracketed.

| | Corridor | | | Big Room | | Room |
|---|---|---|---|---|---|---|
| Frames / Inc. | 532 / 1 | 266 / 2 | 177 / 3 | 272 / 1 | 90 / 3 | 240 / 1 |
| ICML | **0%** (0.4%) | **0%** (4.9%) | **0%** (11.3%) | **0%** (0%) | **0%** (4.4%) | **0%** (0%) |
| ICML$_h$ | **0%** (0.4%) | **0%** (4.5%) | **0%** (15.3%) | **0%** (0%) | **0%** (3.3%) | **0%** (0%) |
| ICL | **0%** (1.7%) | **0%** (10.9%) | **0%** (19.8%) | **0%** (0%) | **0%** (10%) | **0%** (0%) |
| ICP | **0.9%** | **7.1%** | **16.4%** | **0%** | **5.5%** | **0%** |
| VISO2 | **0%** | **0%** | **0%** | **0%** | **0%** | **12.9%** |

Table 4.2 lists the registration failure rates for different frame increments to test for robustness with increasing distances between frames. Without sample consensus matching (the results of which are bracketed) the pure ICML convergence rates can be directly compared to ICP. Figure 4.6 depicts two configurations where ICML was able to find the correct motion while ICP failed, because many of the closest matches are not the correct ones. It is also interesting to compare ICML with ICL in this way, as the registration failures have been approximately halved without sacrificing speed. In comparison to ICP, the registration failures are often reduced by about 30%. The comparison of the different cost functions for ICML reveals no overall winner. While the Huber function theoretically seems more robust than the MSE, the results of ICML$_h$ in Table 4.2 show no clear indication for the tested sequences. Often, ICML$_h$ beats ICML by just one successful registration. On the other hand, ICML can be superior in some cases like for the Corridor sequence with large frame increment.

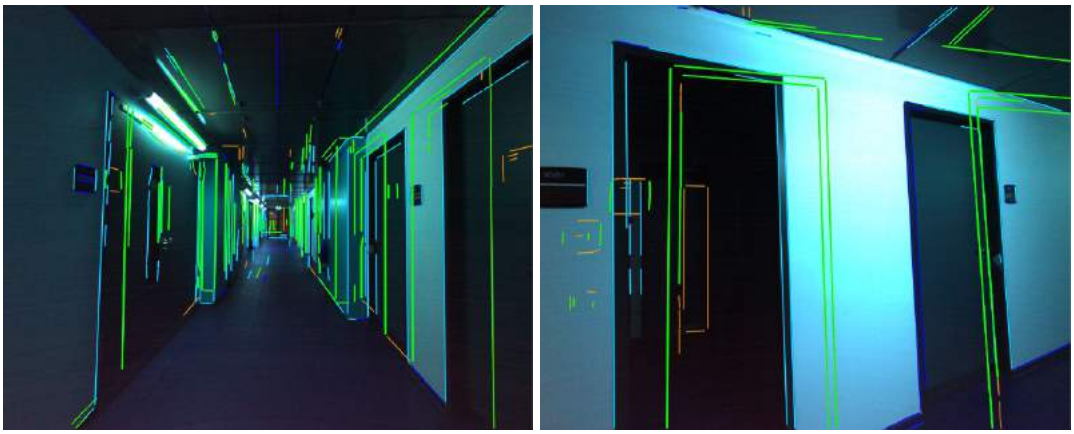


Figure 4.6: These images show initial matching configurations of the Corridor sequence. ICML was able to register the lines without sample consensus matching. ICP falls into local minima on the shown setups, because some of the closest lines are false matches. The images show the initial reprojection of the previous frame prior to registration with $d_{\max} = d_{\text{init}} = 64$px. Green depicts matched reprojected model lines, while unmatched ones are drawn in orange. Detected image lines are cyan if they were matched and dark blue for non-matched ones.

Table 4.3: Average total computation times per frame. For the iterative optimization variants: stereo line segment time + iterative optimization time = total time.

| | Corridor | Big Room | Room |
|---|---|---|---|
| ICML | 35+7=42ms | 44+8=52ms | 16+4=20ms |
| ICML$_h$ | 35+8=43ms | 44+9=53ms | 16+4=20ms |
| ICL | 35+7=42ms | 44+7=51ms | 16+3=19ms |
| ICP | 35+169=204ms | 44+180=224ms | 16+51=67ms |
| VISO2 | 91ms | 103ms | 63ms |

As the Big Room sequence is the visually most complex of the three, one would expect no big difference in accuracy between feature point and edge based methods. However, the large number of precisely detectable straight lines seems to benefit the edge based methods, see Figure 4.7. The trajectory of ICP is almost identical to the one of ICML, as anticipated if no registration errors occur (except for a minor misregistration of ICP near the sequence end). As ICL can only match one line, it does not always choose the best one in terms of accuracy (e.g. it could choose a small line that is closest by coincidence). While the accumulated error is not big here, it could be in other scenarios where the line detection is more unreliable.

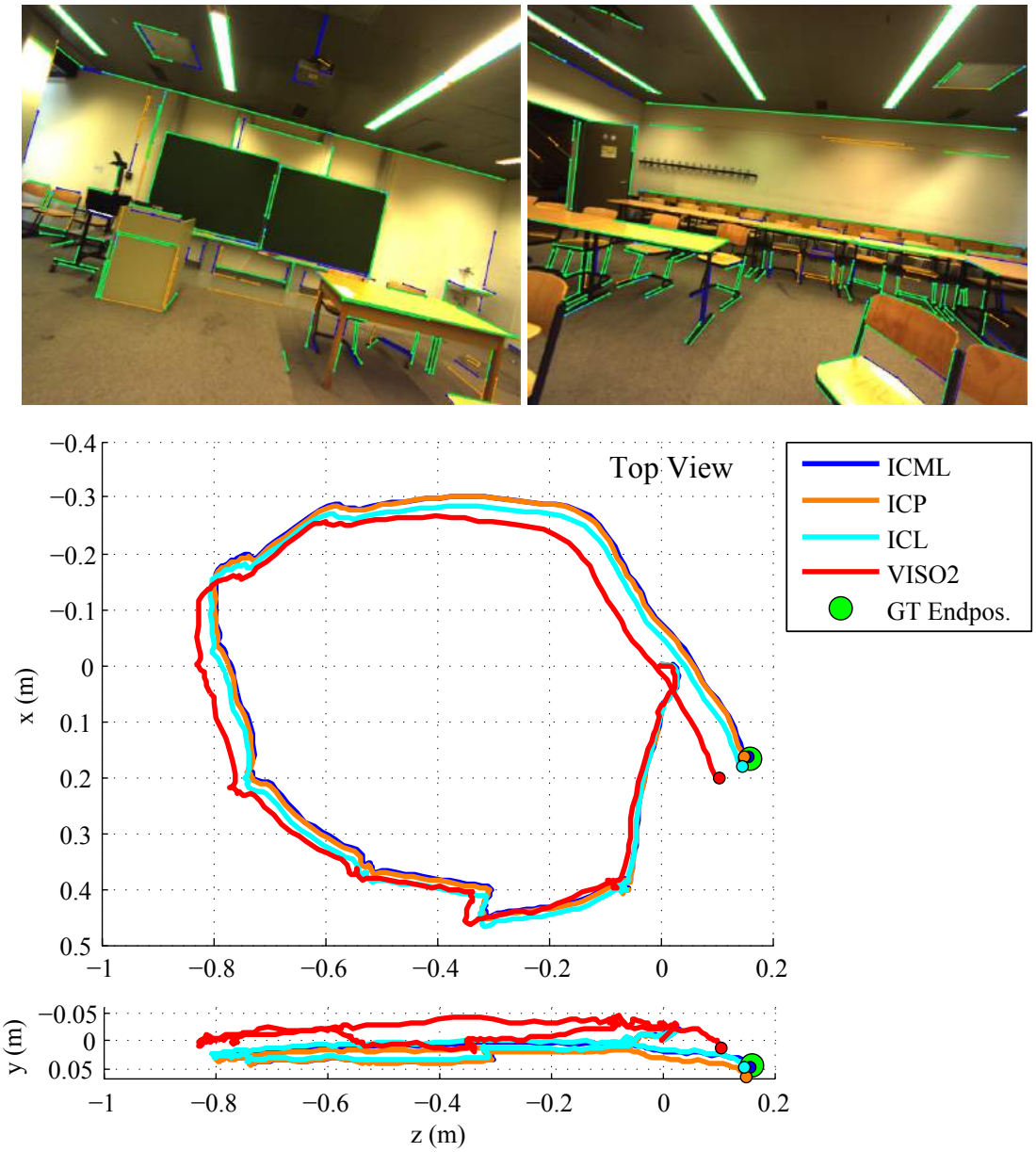



Figure 4.7: The Big Room sequence contains many three dimensional objects and texture, as visible in the two frames above the trajectory plot (matched lines drawn green). With 0.2%, ICML achieves the highest accuracy on this short loop of 3m length, followed by ICL with 0.7% and ICP with 0.8%. VISO2 only achieves 2.4%, despite numerous feature points.

The given time values in Table 4.3 are the mean of all measurements in a sequence. For ICML, ICL and ICP the stereo matching and line detection time (first number) and reprojection optimization time (second number) are given sep-

arately. Note, that the optimization times of ICP are more than an order of magnitude larger than for ICML. The mean times for sample consensus (SAC) were about 50ms in the case of the Corridor sequence, with its extrema between 2ms (with early out detection) and 130ms for $N_{\text{SAC}} = 60$. Figure 4.8 shows the trajectories for the Corridor sequence with a frame increment of one. Due to the medium feature count of the scene, VISO2 performs well. Yet, ICML is twice as fast and more accurate. ICP suffers from the lack of a registration failure recovery technique to compete in such challenging scenes. While a method was proposed in (Tomono, 2009a), a significant increase of the performance gap to ICML is expected due to the use of expensive SIFT features. Furthermore, in very sparse cases, falling back to a feature point based technique may not be optimal.

## 4.5 Discussion

We have presented a new projective line registration algorithm, called ICML, that is substantially faster than ICP for the analogous problem. We showed that a naive "Iterative Closest Line" algorithm is inferior in terms of registration success, since it is impossible to find the ultimately best matching line in many cases. The allowance for multiple matches in ICML eliminates this hindrance and even leads to an improvement in registration performance, compared to ICP. Additionally, we proposed a robust sample consensus based bootstrapping algorithm which is automatically used once a failure criterion is fulfilled. The performance was demonstrated in an untextured and more complex, moderately textured environments. The comparison with a state-of-the-art feature point based algorithm demonstrated the potential of this technique both in accuracy and speed. However, environments without sufficient numbers of straight line segments will certainly favor traditional feature point based odometry. This suggests a hybrid system for optimal performance in diverse environments. Nevertheless, a fast and self-contained edge based method can be vital for successful navigation. The source code of the developed C++ Line Vision Library and videos of the experimental results are available online.[1]

[1] Visit `http://www.jonaswitt.de/ICML.shtml` for the ICML source code and videos.

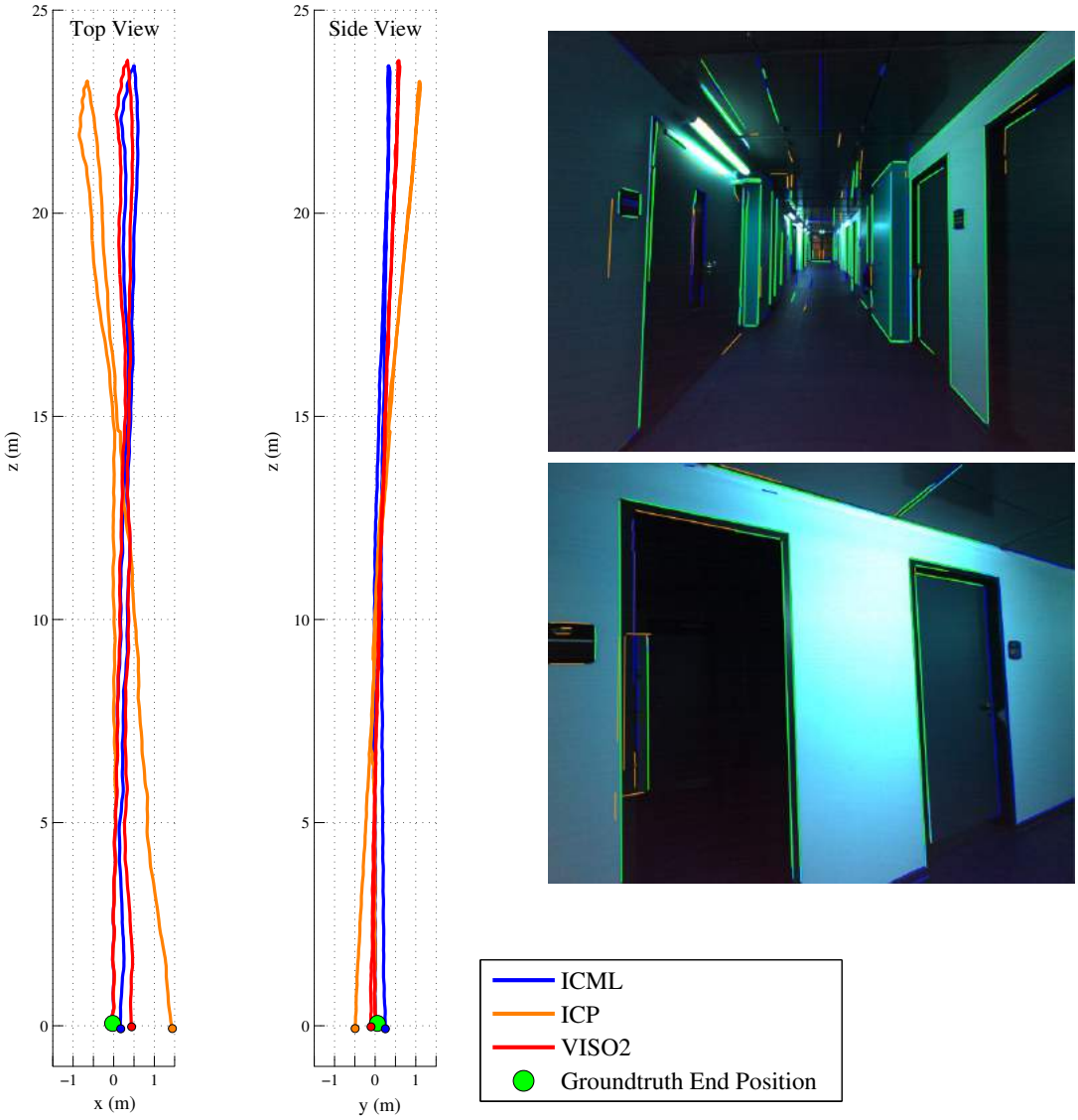



Figure 4.8: In the Corridor sequence, a total distance of 48m is walked moderately fast with a hand-held stereo camera. Due to the medium feature point count, VISO2 is able to achieve an accuracy of 1%. However, with 0.65%, ICML is even more accurate since every edge pixel refines the solution, in contrast to fewer reliable corner-like features. The accuracy of ICP suffers from the lack of a registration failure recovery algorithm (3.3% accuracy) but would otherwise be comparable to ICML. ICL (not plotted) approaches the accuracy of ICML, but falls back to sample consensus matching more often.

# Chapter 5

# Line SLAM using Bundle Adjustment

With the line segment tracking and frame-to-frame odometry from Chapter 4 we are able to establish line correspondences over multiple frames. This is the foundation of the Simultaneous Localization and Mapping (SLAM) solution that we propose in this chapter. Since we are interested in solving the visually sparsest permissible cases, we do not require the availability of any texture to contribute information to the correspondence problem. Another distinction to feature point-based SLAM systems is the explicit estimation of the one-dimensional extent of line segments. The unreliable nature of line segment end point locations is specifically challenging in this respect. We address all these aspects in the proposed Line SLAM system and additionally showcase the loop-closure potential by matching views differing by as much as 180°.

## 5.1 Introduction

The SLAM problem is fundamental to robotics and many different approaches have been proposed (Bailey and Durrant-Whyte, 2006). As explained in Section 1.2, we wish to map the environment and, at the same time, localize the robot based on this map. Although research on camera-based SLAM problems has been conducted for a while (Harris and Pike, 1987), the last decade has seen significant advances in the field through improved algorithms and increased computing power. One of the first real-time monocular SLAM systems was presented in

(Davison, 2003). Landmark and camera poses were modeled as (often Gaussian) probability distributions and updated using filtering algorithms (e.g. Kalman filter variants). While more sophisticated batch optimization techniques (i.e. bundle adjustment) were already well-known, the belief that they were incompatible with real-time requirements prevented their application for online environment mapping. The first ”live” uses of bundle adjustment were limited to small sliding windows of recent frames to improve visual odometry (Nistér et al., 2004). Klein and Murray (2007) were the first to propose a real-time system that performed tracking and batch optimization based mapping in parallel tasks which still allowed fast tracking while reconstructing the maximum likelihood feature map.

Regardless of batch optimization undoubtedly being the superior technique in terms of accuracy, filtering was conceived to usually be faster, and thus was preferred for small computational budgets. In general, however, it was unclear how the two compared in terms of computing cost per accuracy. The first work to do a thorough investigation on this matter was done by Strasdat et al. (2010). They finally conclude that in most modern applications, keyframe batch optimization gives the most accuracy per unit of computing time.

A problem with the initial formulations was that the computation time drastically increased with the map size (i.e. quadratically) until an online solution became infeasible. Addressing this fundamental issue, relative SLAM formulations eventually reduced the problem complexity and enabled scalable real-time mapping, while allowing for large loop-closures (Bosse et al., 2004; Konolige and Agrawal, 2008; Sibley et al., 2010).

## 5.2 Related Work

Despite there being comparably little research on line-based SLAM solutions, line reconstruction has received attention for a longer time. Taylor and Kriegman (1995) proposed one of the first reprojection optimizations for line reconstruction. They addressed the pure geometry recovery, omitting the line correspondence problem by manual matching. Later, Bartoli and Sturm (2005) provided a comprehensive article on line representation, triangulation and bundle adjustment. However, automatic matching of lines and handling of segment end points

was also not addressed in this work.

Schmid and Zisserman (1997) explore automatic line matching in two and three views by utilizing a cross-correlation based matching score and geometric constraints. In the three view case, 3D line segment reconstructions are computed. They later extend their work to curve matching for small baselines (Schmid and Zisserman, 2000). Since cross-correlation is used as a criterion, a somewhat textured line neighborhood is required for this to be expressive. Another shortcoming of correlation as a score is its photometric strictness. Due to specularities and other photometric effects, perspective changes will affect the matching adversely even despite the expensive surface homography estimation that Schmid and Zisserman (2000) propose.

The approach presented by Chen and Wang (2011) builds upon a dense point reconstruction such as proposed by Furukawa and Ponce (2010). Hence, the line reconstruction can only be successful in sufficiently textured scenes and inherits the significant computational burden from the point-based method. Hofer et al. (2013) also rely on conventional point-based methods as a preprocessing step – but only to reconstruct the camera poses. Subsequently, the recovered epipolar geometry is used to restrict the search space and find line matches. They demonstrate the successful reconstruction of large wiry objects, however with textured surroundings which permits recovering the camera trajectory by feature point matching. The methods of Werner and Zisserman (2002) and Schindler et al. (2006) both restrict the solution space by enforcing geometrical constraints such as the Manhattan world assumption and thus have limited scope. Jain et al. (2010) utilize topological constraints and a computationally expensive sweeping approach to find the most probable 3D reconstruction.

None of the above approaches are targeted at real-time mapping on a robot. In fact, the computational burden of most methods is prohibitive in this respect. On the other side of the spectrum, Jeong and Lee (2006) propose an EKF-based SLAM algorithm for a robot with a ceiling-facing camera. Only vertical and horizontal lines are considered. Zhou et al. (2015) propose a line SLAM method that assumes a Manhattan world to eliminate angular and reduce positional drift in indoor and other predominantly orthogonal environments.

Smith et al. (2006) describe a fast-running monocular line SLAM system that is very restrictive when selecting line segments: only segments that are enclosed by two corners are used. For tracking, correlation of the segment neighborhood

is used. Eade and Drummond (2009) use edgels as matching and mapping primitive. Edgels are edge fragments of a fixed size that are matched by correlating its neighborhood between frames. The selection of edgels is restrictive and the enforced fragmentation of edges to fixed size primitives leads to incomplete reconstructions. Finally, Tomono (2009b) proposed a stereo edge SLAM method that was inspirational to our Line SLAM approach. Both use a reprojection optimization to find frame-to-frame correspondences and require a stereo camera. In contrast to Tomono's EKF-based edge point estimation, we estimate line segments using bundle adjustment. Another important distinction is the allowance for loop-closures in our system.

To solve the sparse bundle adjustment problem efficiently, a number of well-written libraries have been published. Lourakis and Argyros (2009) developed a sparse bundle adjustment package named SBA (implemented in ANSI C). More general in scope, Dellaert (2012) and his group continuously extend their C++ based factor graph framework GTSAM which is suitable for many different problems that can be expressed in this form. Featurewise similar, Kümmerle et al. (2011) open-sourced their efficient C++ based framework $g^2o$. In their paper, they showed that their implementation outperformed other implementations in terms of computational efficiency, which is why we chose to base our line bundle adjustment on this library.

## 5.3 The Bundle Adjustment Problem

This section briefly introduces the bundle adjustment problem. For a more in-depth review, Hartley and Zisserman (2004) and Triggs et al. (2000) provide an excellent reference. In general, bundle adjustment is a technique for estimating projective reconstructions from a set of images. This includes the reconstruction of camera related parameters such as its pose and projection matrix as well as the scene geometry that is viewed. The previous chapter focused on solving the line association problem from an image sequence and recovering the camera motion through a reprojection optimization of the most recent stereo reconstruction with respect to the current frame. Bundle adjustment formulates the reprojection optimization for an arbitrary number of camera observations and features simultaneously. We will first review the case of corresponding feature points in

multiple views and then transfer the method to line segments. The relation between a world point $\mathbf{p}_j$ and its projection $\mathbf{z}_{ij}$ as seen through a camera with the projection matrix $\mathsf{P}_i$ is

$$\mathbf{z}_{ij} = \mathsf{P}_i \mathbf{p}_j. \tag{5.1}$$

The homogeneous $3 \times 4$ projection matrix $\mathsf{P}$ includes the intrinsic and extrinsic parameters

$$\mathsf{P} = \begin{bmatrix} f_\mathrm{x} & s & o_\mathrm{x} \\ 0 & f_\mathrm{y} & o_\mathrm{y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathsf{R}^\top & -\mathsf{R}^\top \mathbf{t} \end{bmatrix}. \tag{5.2}$$

Often the intrinsic parameters are simpler as the *skew* parameter $s = 0$ and for square pixels, the focal length is $f = f_x = f_y$. If the principal point $o_x, o_y$ and the focal length $f$ are calibrated before the bundle adjustment, they can even be left out of the optimization completely – leaving only the six extrinsic parameters to be estimated per camera observation. As in the previous chapter, $[\mathsf{R}|\mathbf{t}]$ describes the camera-to-world transformation.

In general, the measurements $\bar{\mathbf{z}}_{ij}$ will be noisy and thus the corresponding reprojections $\mathbf{z}_{ij}(\mathsf{P}_i, \mathbf{p}_j) = \mathsf{P}_i \mathbf{p}_j$ cannot exactly coincide for all points. Accordingly, the reprojection error

$$\mathbf{e}(\mathsf{P}_i, \mathbf{p}_j, \bar{\mathbf{z}}_{ij}) = \bar{\mathbf{z}}_{ij} - \mathbf{z}_{ij}(\mathsf{P}_i, \mathbf{p}_j) \tag{5.3}$$

will usually be nonzero. However, we intend to find the Maximum Likelihood solution to this reconstruction problem. As is well-known, under the assumption of Gaussian noise, this can be achieved by minimizing the sum of squared errors

$$\min_{\mathsf{P}_i \mathbf{p}_j} \sum_{ij} \mathbf{e}(\mathsf{P}_i, \mathbf{p}_j, \bar{\mathbf{z}}_{ij})^\top \; \mathsf{W}_{ij} \; \mathbf{e}(\mathsf{P}_i, \mathbf{p}_j, \bar{\mathbf{z}}_{ij}). \tag{5.4}$$

In this case, the information matrix $\mathsf{W}_{ij}$ contains the inverse covariance $\Sigma_{ij}^{-1}$ of the measurement noise corresponding to $\bar{\mathbf{z}}_{ij}$. Condensing the sought-after parameters $\mathsf{P}_i, \mathbf{p}_j$ into the joint parameter vector $\mathbf{x}$ and building the block diagonal matrix $\mathsf{W}$ from the individual $\mathsf{W}_{ij}$, we can write the linear system with the delta parameter vector $\Delta\mathbf{x}$ whose solution minimizes the quadratic error for the current

linearization around $\mathbf{x}$

$$\underbrace{\mathsf{J}^\top \mathsf{W} \mathsf{J}}_{\mathsf{H}} \, \Delta\mathbf{x} = -\underbrace{\mathsf{J}^\top \mathsf{W} \mathbf{e}}_{\mathbf{g}}. \tag{5.5}$$

The Jacobian $\mathsf{J} = \frac{\partial \mathbf{e}(\mathbf{x}+\Delta\mathbf{x})}{\partial \Delta\mathbf{x}}$ relates the joint error vector $\mathbf{e}$ to the increment $\Delta\mathbf{x}$ around the current linearization point $\mathbf{x}$. After each step, $\mathbf{x}$ is updated with the found delta parameter vector

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}. \tag{5.6}$$

The iterative non-linear optimization can be carried out using the popular Levenberg-Marquardt algorithm (Hartley and Zisserman, 2004) which uses the scalar $\lambda$ to control the descent direction between the gradient solution and Gauss-Newton

$$(\mathsf{H} + \lambda \mathsf{I}) \, \Delta\mathbf{x} = -\mathbf{g}. \tag{5.7}$$

As for all gradient-based optimization techniques, a good initialization has to be provided in order for the algorithm to converge to the desired solution. However, the solution from the sequential motion reconstruction that was discussed in the previous chapter can readily serve this purpose.

### 5.3.1 Robust Optimization

Triggs et al. (2000) point out that according to the Gauss-Markov theorem, least squares gives the Best Linear Unbiased Estimator (BLUE) even for non-Gaussian noise (with the given mean $\bar{\mathbf{z}}_{ij}$ and covariance $\mathsf{W}_{ij}^{-1}$) as long as the model $\mathbf{z}_{ij}(\mathsf{P}_i, \mathbf{p}_j)$ is linear. 'Best' in this case being minimum variance. However, albeit a given measurement noise may be approximately Gaussian in general, occasional outliers can quickly dominate the error functional (Hartley and Zisserman, 2004). This is due to its small tails: the outlier frequency that a Gaussian model does not resemble most real measurements. We could try to find the true noise distribution and attempt to model it as a mixture of Gaussians, but in practice we just want to model the inliers (assuming a Gaussian) and reject outliers. A popular choice for this are heuristically justified robust cost functions. Figure 5.1 shows several robustified versions of the squared error. The corresponding mathematical defi-

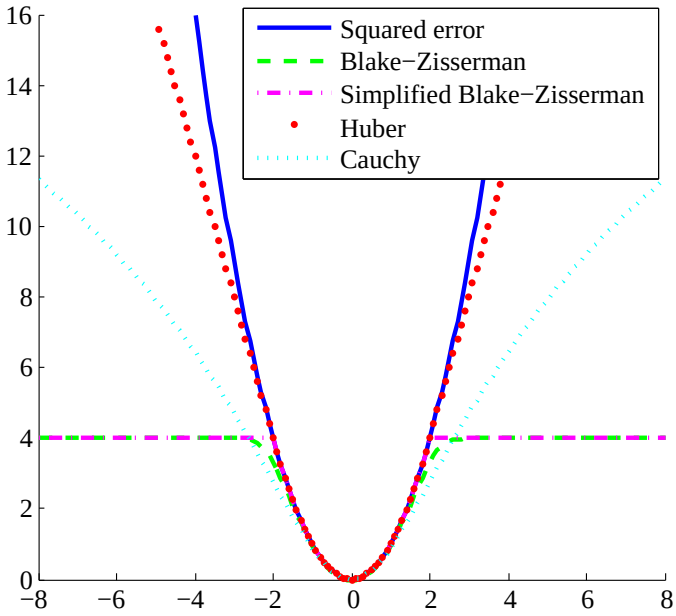



Figure 5.1: Robust cost functions in comparison to squared error. For errors below the outlier threshold $\alpha = 2$ all functions approximately resemble the squared error.

nitions are given in the following:

$$\text{Squared Error:} \quad \rho(\delta) = \delta^2 \tag{5.8}$$

$$\text{Blake-Zisserman:} \quad \rho(\delta) = -\log\left(\exp\left(-\delta^2\right) + \exp\left(-\alpha^2\right)\right) \tag{5.9}$$

$$\text{Simplified Blake-Zisserman:} \quad \rho(\delta) = \min(\delta^2, \alpha^2) \tag{5.10}$$

$$\text{Huber:} \quad \rho(\delta) = \begin{cases} \delta^2 & \text{if} \quad |\delta| < \alpha \\ 2\alpha\,|\delta| - \alpha^2 & \text{else} \end{cases} \tag{5.11}$$

$$\text{Cauchy:} \quad \rho(\delta) = \alpha^2 \log\left(1 + \frac{\delta^2}{\alpha^2}\right) \tag{5.12}$$

The parameter $\alpha$ is usually set to the outlier threshold. All functions have in common that they reduce the influence of outliers while keeping the favorable properties of least squares for inliers. The Blake-Zisserman variants limit the influence of outliers to a constant value, while Cauchy keeps a logarithmic contribution towards infinity. However, all three are non-convex. In contrast, the Huber function has the desirable combination of being robust to outliers while staying convex (Hartley and Zisserman, 2004). The latter has the positive effect

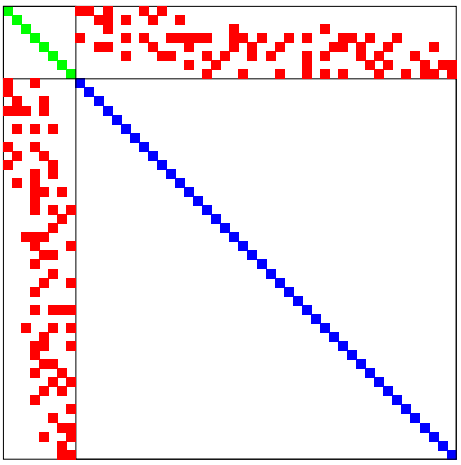


Figure 5.2: Sparsity pattern of the H matrix for a toy bundle adjustment problem with 8 camera observations (green) and 42 features (blue). The red entries indicate which features where visible from which camera observations.

that no new local minima are introduced. The robustified error functional takes the following form

$$f(\mathbf{x}) = \sum_{ij} \rho_{ij} \left( \sqrt{\mathbf{e}_{ij}(\mathbf{x})^\top \, \mathsf{W}_{ij} \, \mathbf{e}_{ij}(\mathbf{x})} \right) . \tag{5.13}$$

Here, $\mathbf{x}$ is the parameter vector with respect to which $f(\mathbf{x})$ shall be minimized.

### 5.3.2 Exploiting the System Structure

When we order the entries in the parameter vector $\mathbf{x}$ by camera and feature parameters $\mathbf{x}^\top = [\mathbf{x}_\mathsf{c}^\top, \mathbf{x}_\mathsf{f}^\top]$, the H matrix has a very characteristic structure, as exemplarily depicted in Figure 5.2. Rewriting Eqn. (5.5) in this block form yields

$$\begin{bmatrix} \mathsf{H}_{\mathsf{cc}} & \mathsf{H}_{\mathsf{cf}} \\ \mathsf{H}_{\mathsf{cf}}^\top & \mathsf{H}_{\mathsf{ff}} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}_\mathsf{c} \\ \Delta\mathbf{x}_\mathsf{f} \end{bmatrix} = \begin{bmatrix} -\mathbf{g}_\mathsf{c} \\ -\mathbf{g}_\mathsf{f} \end{bmatrix} . \tag{5.14}$$

Now, $\mathsf{H}_{\mathsf{cf}}$ is the only dense matrix in the equation and this enables us to do a simplifying reduction. Using the Schur complement, we can solve the equivalent system much more efficiently (Triggs et al., 2000)

$$\left( \mathsf{H}_{\mathsf{cc}} - \mathsf{H}_{\mathsf{cf}} \mathsf{H}_{\mathsf{ff}}^{-1} \mathsf{H}_{\mathsf{cf}}^\top \right) \Delta\mathbf{x}_\mathsf{c} = -\mathbf{g}_\mathsf{c} + \mathsf{H}_{\mathsf{cf}} \mathsf{H}_{\mathsf{ff}}^{-1} \mathbf{g}_\mathsf{f} . \tag{5.15}$$

Through this, we only have to invert the block-diagonal matrix $\mathsf{H}_{\mathrm{ff}}$ and then solve the reduced system, which is usually significantly more efficient when the features outnumber the camera observations (Kümmerle et al., 2011). Using $\Delta\mathbf{x}_{\mathrm{c}}$, we can back-substitute and yield the remaining system for the feature update $\Delta\mathbf{x}_{\mathrm{f}}$

$$\mathsf{H}_{\mathrm{ff}}\Delta\mathbf{x}_{\mathrm{f}} = -\mathbf{g}_{\mathrm{f}} - \mathsf{H}_{\mathrm{cf}}^{\top}\Delta\mathbf{x}_{\mathrm{c}} \,. \tag{5.16}$$

However, if we would be interested in just the camera poses we would not even have to solve this equation.

## 5.4 Line SLAM

Transferring the bundle adjustment problem to line segments involves tackling a number of challenges that feature point based SLAM methods naturally do not have. We will first explore different possibilities for parameterizing lines for geometric reconstruction and then describe the approach that was used to manage line segments during SLAM.

### 5.4.1 Line Parameterization

One main difficulty is the algebraic representation during line reconstruction (Bartoli and Sturm, 2005). As elaborated in the previous section, the unreliable nature of line segment end points forces us to treat them as infinite lines during optimization and transform their representation for this purpose. Feature points are a lot more straight forward in this respect, since they are well located and have a natural minimal representation as either a 3-dimensional vector (excluding points at infinity) or a 4-dimensional homogeneous vector when infinite distances shall be considered in a reconstruction. In contrast, the analog representation of a line by two points is overparameterized since they can freely be slided along without changing the line's geometry. These gauge freedoms are an unnecessary burden during optimization since the parameter vector is bloated by an additional 50% in comparison to the minimal representation of 4 parameters per line (Hartley and Zisserman, 2004). In the case of two homogeneous points, the resulting 8 parameters per line double the parameter vector size. Apart from being a hindrance to efficient reconstruction, such internal gauge freedoms may

even induce numerical instabilities.

One important distinction between different representations is completeness. This property describes whether it is possible to cover all of $\mathbb{P}^3$ with a representation (including infinity). For example, a 3-dimensional point is an incomplete representation while a 4-dimensional homogeneous vector is able to represent points at infinity without becoming singular. Yet, for our Line SLAM system, we do not consider lines at infinity relevant and are rather interested in practical considerations and efficiency when transforming from line segments to infinite lines and back. However if one is interested in complete reconstructions, Bartoli and Sturm (2005) provide a comprehensive discussion including several incomplete representations. In the following, we will discuss the properties of four different incomplete parameterizations.

#### Two Points

While it is an overparameterization with two gauge freedoms, the line segment representation with two three-dimensional points can be used during line optimization. Using $\lambda_1, \lambda_2$ to parameterize the gauge freedoms, the line $\mathbf{l}$ is equivalent to all lines $\mathbf{l} + \Delta\mathbf{l}(\lambda_1, \lambda_2)$

$$\mathbf{l} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} \tag{5.17}$$

$$\Delta\mathbf{l}(\lambda_1, \lambda_2) = \begin{bmatrix} \lambda_1(\mathbf{p}_2 - \mathbf{p}_1) \\ \lambda_2(\mathbf{p}_2 - \mathbf{p}_1) \end{bmatrix}. \tag{5.18}$$

Apart from the computational bloat during optimization, this representation was found to be numerically safe, since the gradients of the error functional become zero in the line direction. Accordingly, the non-linear optimization has no incentive in excessively moving the endpoints other than perpendicular to the line axis.

#### Closest Point and Direction

A similar variant is the parameterization with the closest point to the origin $\mathbf{p}$ and a normalized direction $\mathbf{d}$. The main difference to the previous representation is

that the gauge freedoms are transformed into constraints

$$\mathbf{l} = \begin{bmatrix} \mathbf{p}_{\text{min}} \\ \mathbf{d} \end{bmatrix} \tag{5.19}$$

$$\|\mathbf{p}_{\text{min}}\| \leq \|\mathbf{p}_{\text{min}} + \lambda \mathbf{d}\| \quad \forall \quad \lambda \in \mathbb{R} \tag{5.20}$$

$$\|\mathbf{d}\| = 1. \tag{5.21}$$

However, if the constraints are not enforced during optimization, this representation is essentially equivalent to the two point variant. While the gauge freedoms can be eliminated this way, enforcing the constraints is tedious and the dimensionality is the same. The reprojection of both this and the two point representation can be done by simply projecting any two points on the line and is thus bilinear.

#### Denavit-Hartenberg Parameters

A line parameterization using only the minimal 4 parameters can be derived from the Denavit-Hartenberg representation. In its original form it describes the relation between two robotic coordinate frames by two distances and two angles (Denavit and Hartenberg, 1955). Without going into detail, one can imagine that if the $z$-axis of the target frame is interpreted as a line, all possible lines can be represented by the transform. Roberts (1988) bases a similar representation on this, however both have the drawback that non-linear operations are required for reprojection since the rotations introduce trigonometric functions. Hence, it is not ideal for all cases.

#### Orthogonal Relative Subspace

However, during optimization we do not need to be able to represent the whole space of possible lines. In fact, we are only interested in a singularity free delta space around the current estimate. This is the motivation behind the formulation of the orthogonal relative subspace, see Figure 5.3. Since in the end we are interested in line segments and the geometry of their endpoints, it is beneficial to use a line representation that naturally transforms forth and back. This is achieved by parameterizing the two-dimensional subspace that is orthogonal to the given current estimate of a line segment $\mathbf{l}^\top = (\mathbf{p}_1^\top, \mathbf{p}_2^\top)$. With the orthonormal basis

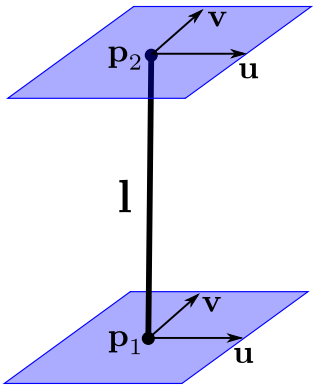



Figure 5.3: Orthogonal relative subspace representation around the current estimate of a line segment $\mathbf{l}^\top = (\mathbf{p}_1^\top, \mathbf{p}_2^\top)$. The orthonormal basis $\{\mathbf{u}, \mathbf{v}\}$ spans the two-dimensional subspace orthogonal to $\mathbf{l}$.

$\{\mathbf{u}, \mathbf{v}\}$ we can transform any 4-dimensional update vector $\mathbf{x}$ that we yield during optimization back to our 6-dimensional line segment $\mathbf{l}$:

$$\mathbf{l}_{k+1} = \mathbf{l}_k + \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{u} & \mathbf{v} \end{bmatrix} \mathbf{x} \tag{5.22}$$

where $\mathbf{x}^\top = (u_1, v_1, u_2, v_2)$. While this representation becomes singular for all lines that are perpendicular to $\mathbf{l}$, this poses no problem since the line segment estimate $\mathbf{l}$ is updated after each bundle adjustment iteration. If $\mathbf{l}$ is rotated, so is its orthogonal subspace.

In general, it does not matter what basis vectors $\{\mathbf{u}, \mathbf{v}\}$ are picked. It is not even a hard requirement that they are normal or orthogonal to each other as long as they are not collinear. However, making $\{\mathbf{u}, \mathbf{v}\}$ orthonormal ensures that the magnitudes of the parameters $\mathbf{x}$ are similar and in general well-behaved for optimization. Another advantage of this representation is that reprojection is as computationally efficient as for the two point parameterization. Table 5.1 summarizes the properties of the discussed line representations.

### 5.4.2 Estimating Line Segment End Points

While the bundle adjustment step treats line segments as if they had infinite extent, for matching and the final reconstructed geometry, this information is essential. This estimation step is specific to SLAM for line segments. In the case of

Table 5.1: Properties of different line representations. Gauge freedoms and constraints add to the complexity of the optimization. The last column states the type of equation required to reproject the given line to a perspective camera.

| | # Gauge | # Constraints | Reprojection |
|---|---|---|---|
| Two Points | 2 | 0 | Bilinear |
| Closest Point and Direction | 0 | 2 | Bilinear |
| Denavit-Hartenberg Parameters | 0 | 0 | Non-Linear |
| Orthogonal Relative Subspace | 0 | 0 | Bilinear |

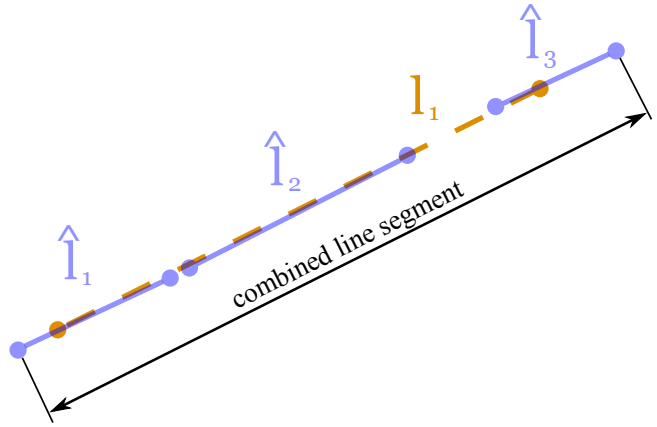



Figure 5.4: When several detected line segments (light blue) are matched to a world line (dashed orange), the combined line segment has to be determined to yield more consistent endpoints. The location of the inner splits is often rather arbitrary and not relevant to for the desired reconstruction.

feature point SLAM, point locations are explicitly estimated during bundle adjustment.

As already mentioned earlier, the endpoint locations of line segments are in general unreliable. A line segment may be detected as monolithic in one frame and split into several distinct ones in another. However, we are interested in gaining an estimate of the underlying geometry over time. Here, we propose a robust yet efficient method for estimating line segment end points.

Figure 5.4 shows a case where three distinct line segments $\hat{\mathbf{l}}_{1..3}$ were detected and matched to a single world line $\mathbf{l}_1$. Since we cannot assume that we already saw the full extent of the world line, we have to update its length according to our recent observation. For this, we find the combined line segment of all matched detections by determining its outer endpoints. Empirically, it has been evident

that the outer endpoints of such a combined line segment are significantly more stable than the inner splitting points. This can be explained when looking at the reasons for split detections:

- slight curvature leading to different polygonizations,
- appearance changes due to lighting and
- occlusions by foreground objects.

All of these factors can lead to arbitrary splits that change with the observation angle. However, the true geometry (like the true endpoints) is invariant to viewpoint changes and as long as the segment extents are visible, most of the time, one of the partial detections will have one of its endpoints close to the true one. This is leveraged by combining the matched detections into one.

Once this combined observation segment is found, sight lines can be cast through its endpoints and intersected with the world line $\mathbf{l}_1$. In general, these lines will be skew to each other and the closest points on $\mathbf{l}_1$ to the cast sight lines are declared its updated endpoints.

One corner case has to be accounted for, though. When the full extent of a world line has been observed, we would not like to crop it only because it leaves the view port and is thus only partially observable. For this reason, we use a safety margin around the extents of the image within which cropping is not allowed. Accordingly, if one endpoint of a combined observation segment is close to the image border, it is disregarded if it would crop the line length. Nevertheless, the other end is still considered as usual.

### 5.4.3 Merging Line Segments

Another important case during Line SLAM is shown in Figure 5.5. Whenever a line was initially detected as several distinct segments, we need to be able to merge them if future observations suggest so. Using the geometry and respective covariances of the merging candidates $\mathbf{l}_1, \mathbf{l}_2$ we can formulate a criterion that determines geometric compatibility. Whenever an observation $\hat{\mathbf{l}}_1$ is matched to more than one world line segment, the following criterion decides whether the

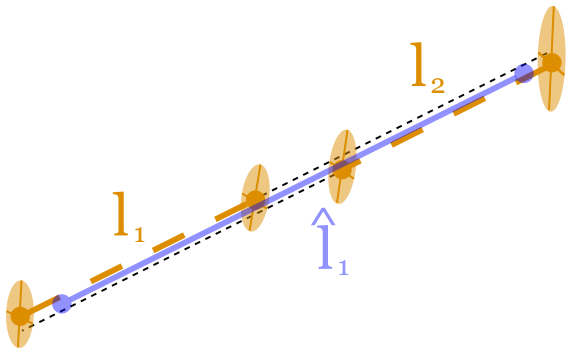



Figure 5.5: Whenever a detected line segment (light blue) is matched to two world lines (dashed orange), a merge has to be considered. The merge is only performed, if the extensions of both world lines (drawn as black dashed lines) are within one standard deviation (depicted as yellow ellipses) of each other.

merge is performed:

$$merge(\mathbf{l}_1, \mathbf{l}_2) = ext(\mathbf{l}_1) \subset \Sigma(\mathbf{l}_2) \ \wedge \ ext(\mathbf{l}_2) \subset \Sigma(\mathbf{l}_1) \tag{5.23}$$

where $\Sigma(\mathbf{l})$ is the single standard deviation volume around $\mathbf{l}$. This means that geometric compatibility is ensured by a mutual check that tests whether each segment encloses the extension $ext(\mathbf{l})$ of the other segment within its one sigma volume.

### 5.4.4 Handling Change and Outliers

Another important consideration for SLAM systems are outliers and physical change of a mapped location. In both cases the system needs to be able to remove parts of the map. Determining whether an unmatched feature should be removed from the map can be determined by checking if its reprojection should be visible in the current frame. However, immediately removing all such mapped features would not be robust to temporary appearance changes and occlusions. For this reason we introduce a life count $life(\mathbf{l})$ to each line. Whenever the life count reaches zero for a given line, it is removed from the map. Using the follow-

ing rules, the life count is updated

$$life_{k+1}(\mathbf{l}) = \begin{cases} life_k(\mathbf{l}) + 1 & if\ m(\mathbf{l}) > \alpha_{\text{good}} \\ life_k(\mathbf{l}) - 1 & if\ m(\mathbf{l}) < \alpha_{\text{bad}} \\ life_k(\mathbf{l}) & else \end{cases} \tag{5.24}$$

where $m(\mathbf{l})$ is the ratio of the cumulative overlap of matched line segments to the length of the world line segment reprojection, accordingly $m(\mathbf{l}) \in [0, 1]$. Appropriate values for the thresholds $\alpha_{\text{good}}$ and $\alpha_{\text{bad}}$ were empirically found to be 0.6 and 0.25 respectively. For the reprojected world line, it is important to crop its length to the actually visible image space. Otherwise the update will be biased towards decreasing the life count.

## 5.5 Experimental Results

The reconstruction experiments were run on the same sequences as used in the previous section. Selected frames of these three datasets are shown in Figures 5.6, 5.11 and 5.9. For the keyframe selection, we merely chose to skip two out of three frames to simplify matters. Accordingly, the Room sequence has 80 keyframes, 177 keyframes in the case of the Corridor sequence and the Big Room sequence has 90 keyframes. However, using a more sophisticated keyframe selection technique as e.g. proposed by Dong et al. (2009) could significantly reduce the number of processed frames.

Since the matching is done with ICML, matches are exclusively determined by the geometry of the lines (as described in the previous section). While this solves an important and difficult problem, the remaining challenge for the reconstruction of this sequence lies in its closeness of the camera to the featureless wall with many long horizontal line segments. The Line SLAM system must cope with partially visible long segments and continuously estimate the segment lengths in order to be able keep matching without starting a new feature (and thus breaking the matching chain).

Figure 5.7 shows the evolution of one of the long horizontal ceiling lines. While the camera trajectory does not build up a lot of baseline, the standard deviation is still reduced significantly through the numerous observations. Through the continuous estimation of the line segment end points, it slowly grows to its

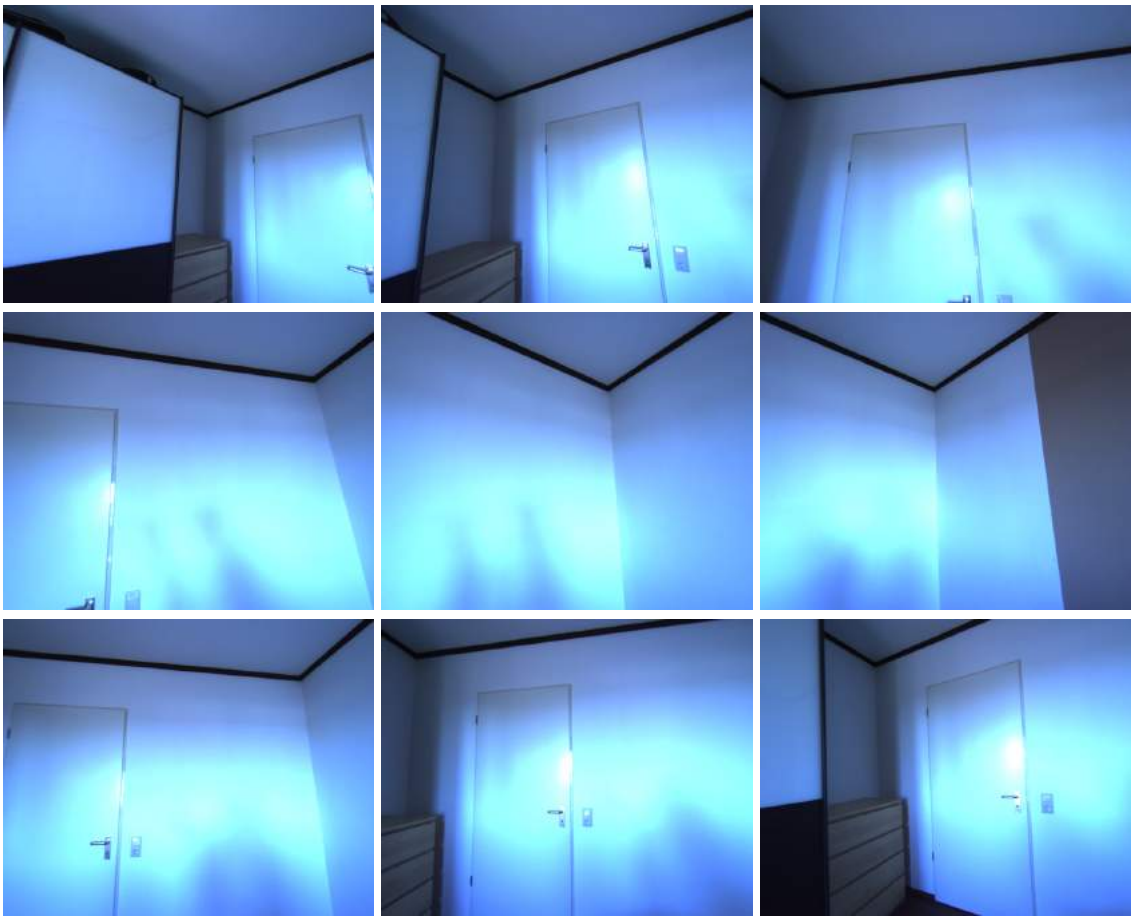


Figure 5.6: Chronologically ordered example frames of the Room sequence. The camera is moved close to the featureless wall. The scene is never observed as a whole. Strong specularities on the wall and door complicate feature-based matching.

true size. While always only partially visible, it is ultimately matched in every frame of the sequence. Such long tracks are very important for the overall accuracy of any reconstruction method.

The final reconstruction of the Room sequence is shown in Figure 5.8. Although, the detections of e.g. the door frame are always broken up into many separate line segments, the proposed methods manage to fuse the different observations into long and geometrically well defined line segments.

The horizontal line segments are specifically challenging since the camera trajectory is moderately ill-posed for their recovery. The detail pictures of the cupboard show that the depth of the horizontal segments is the same as for the adjacent vertical segments. Note that the image neighborhood was not used to for-

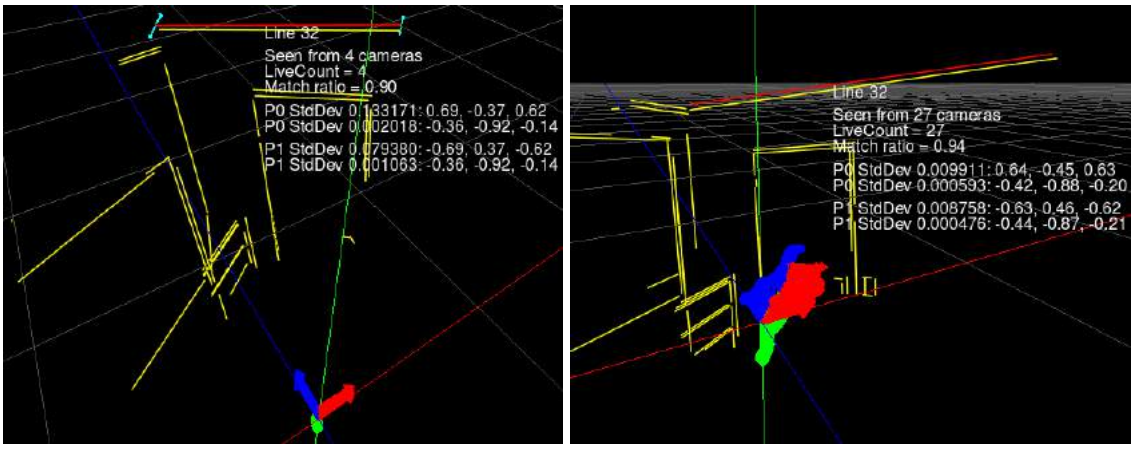



Figure 5.7: These two images show the evolution of a line segment (red) that initially was only partially in the field of view (left image). As the camera rotates to the right, the left segment end point leaves the view frustum (right image). Notice that the same line segment is consecutively matched and its end points are estimated. The camera never observes the full line segment in a single frame. The cyan arrows visualize the standard deviations (indiscernible in the right image).

mulate constraints. The recovered line geometry is only coupled by the shared camera frames.

In lieu of sophisticated explicit loop closure techniques, we chose to simply reproject the full line map into each frame. This yields implicit loop closures and in general only works if the accumulated drift is within the convergence range of ICML. However, while this can be dangerous for long sequences where the positional drift is large, this approach still yields illuminating insights with regards to the potential of line geometry for robust loop closures. However, if a more elaborate method is required, one can adapt algorithms from the rich literature on feature point based methods. Williams et al. (2009) compare a number of popular loop closure approaches and provide a good overview of the topic.

The bundle adjustment was run with a fixed number of 4 iterations every time a keyframe was added. With a sliding window size of 5 keyframes, the optimization took on average 15ms.

The Big Room sequence (see Figure 5.9) has significantly more geometry than the previous sequence. Another difficulty is the average high object distance to the camera and again the camera trajectory. The approximately circular path does not build up any significant baseline towards the many horizontal lines. This results in significantly higher standard deviations for these segments.

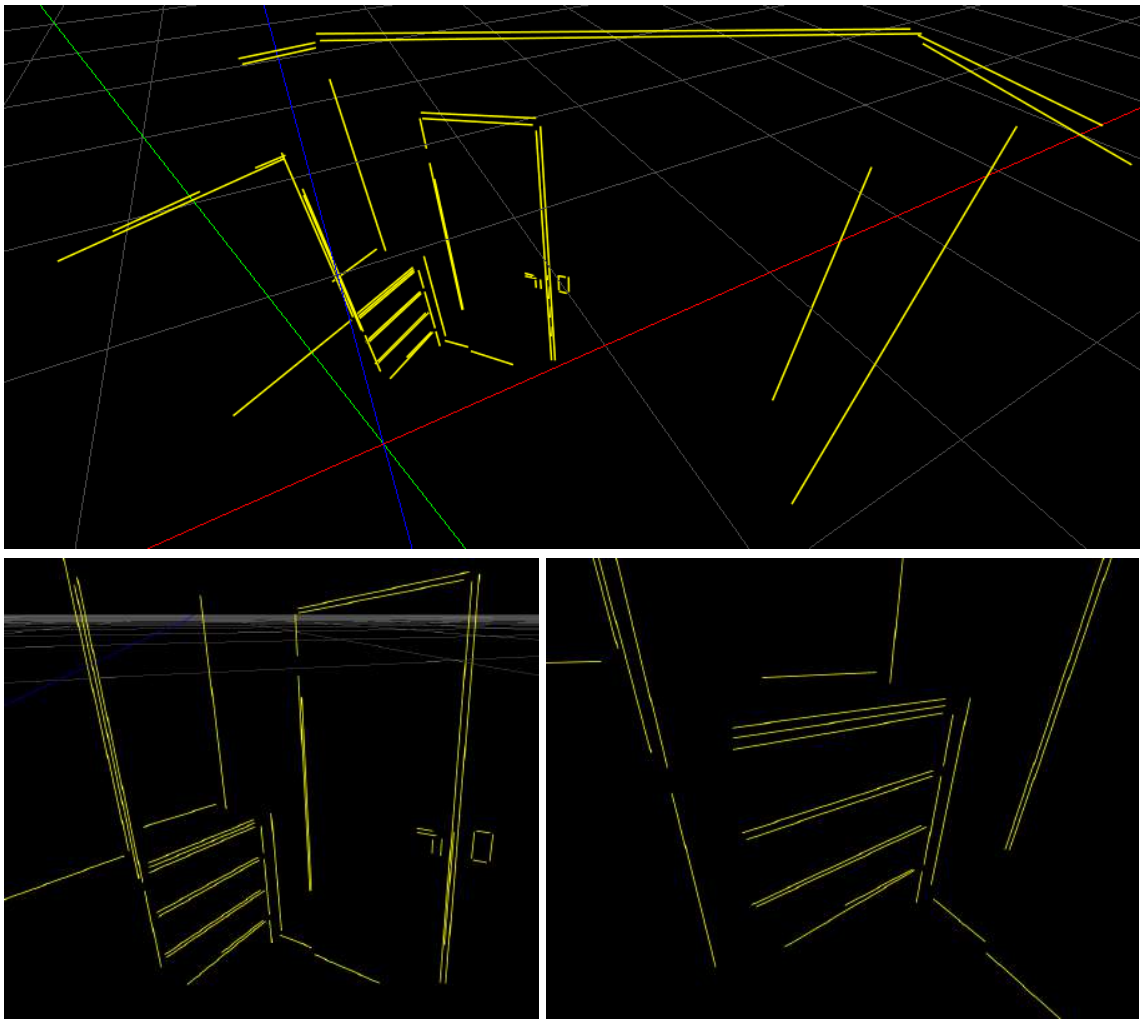


Figure 5.8: The Room reconstruction shows how well all the different observations have been fused into consistent and long line segments that were matched over many observations. The long horizontal ceiling lines have been matched in every single frame of the sequence. The detail of the cupboard (bottom right image) shows that the difficult horizontal segments lie as accurately in the cupboards front plane as the adjacent vertical segments.

Figure 5.10 shows the reconstruction results from the sequence with the camera trajectory in the center. The top image is unfiltered and also includes the horizontal segments with low accuracy (e.g. several segments of the blackboard). The lower images include only higher accuracy segments with a standard deviation smaller than 10cm. One can see that the fine geometry of chairs and tablelegs was recovered for much of the visible geometry. Due to the higher scene complexity, the average computation time per new keyframe increases to 76ms (4

Figure 5.9: Chronologically ordered example frames of the Big Room sequence. The camera trajectory is approximately a circle on the same height.

bundle adjustment iterations with sliding window size of 5).

The Corridor sequence demonstrates the great loop closure potential of the proposed Line SLAM system. The sequence was captured with a hand-held stereo camera and walks up and down the same corridor amounting to a total travel distance of approximately 48m. Figure 5.11 shows selected frames in chronological order.

This sequence also nicely displays the refinement of initially coarse stereo geometry to a finally precisely localized line segment. Figure 5.12 shows the life of a selected line segment from its initial detection (with coarse standard deviation) to its stable final location having been matched in a total of 26 frames.

The most challenging part is the drastic change in viewpoint of 180°. Most loop closure techniques that are based on visual similarity of the neighborhood

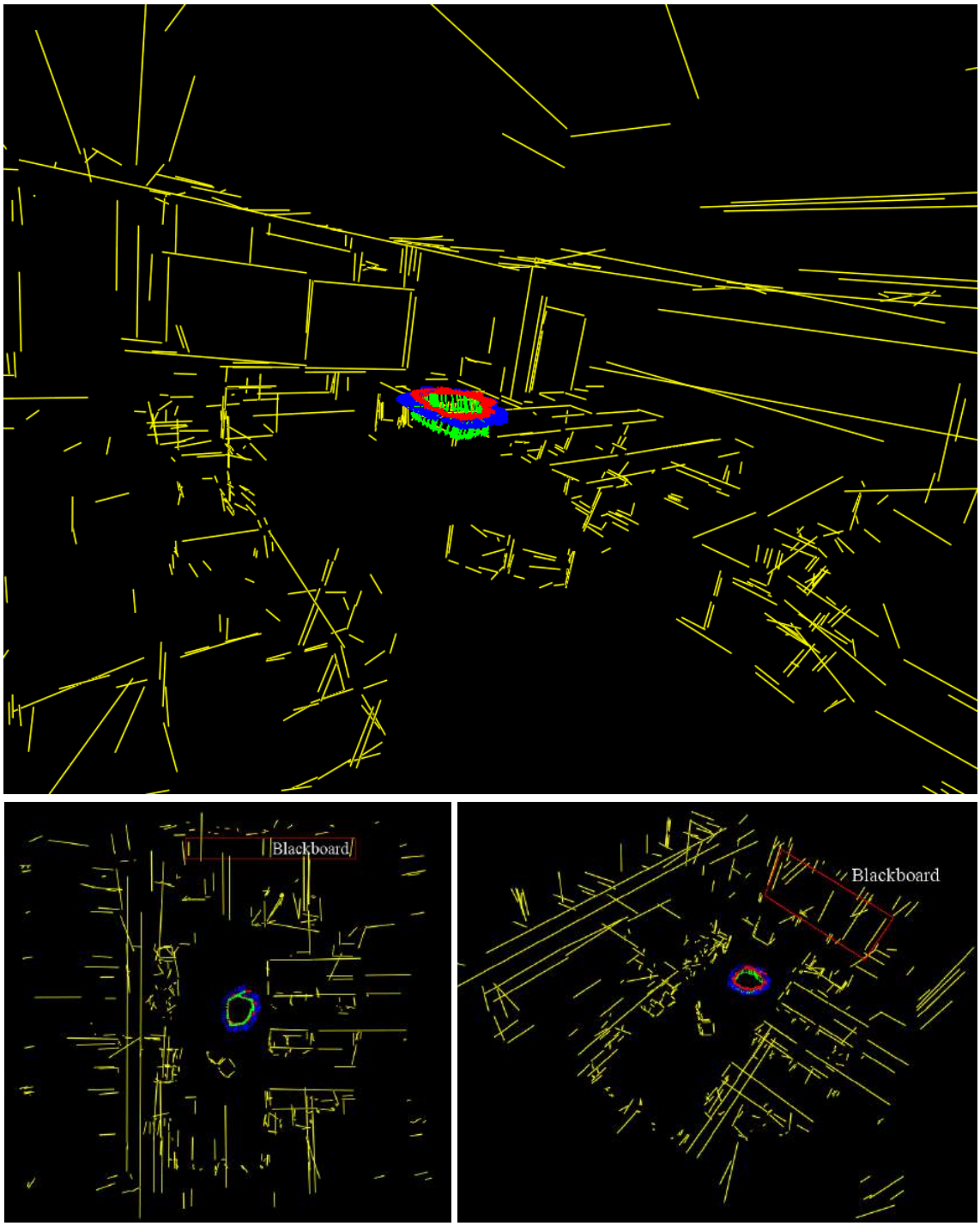



Figure 5.10: The top image shows an unfiltered view of the reconstructed line segments from the Big Room sequence. The bottom images are filtered by lines with a standard deviation smaller than 10cm. This shows that most horizontal lines were not estimated to this fidelity due to the ill-conditioned camera trajectory and large average object distance. However, much of the geometry was reconstructed. The blackboard has been marked in the bottom images for orientation.

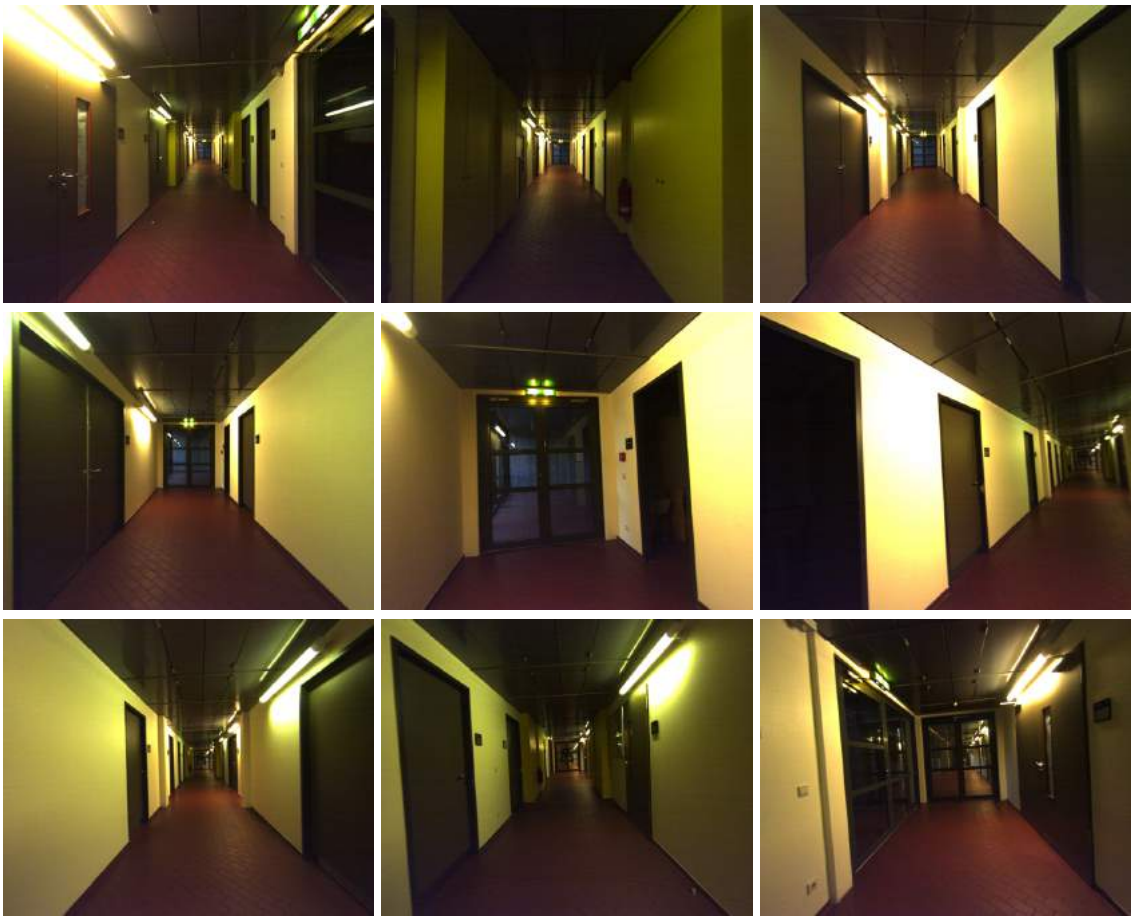


Figure 5.11: Chronologically ordered example frames of the Corridor sequence. The camera was carried up and down the corridor, with a total travel distance of approximately 48$m$.

around features have difficulties in this scenario. However pure geometry is invariant with respect to such drastic photometric changes, which enables straight forward loop closing using our approach. Figure 5.13 shows the complete reconstruction of the Corridor sequence. Note how the vertical lines of the door frames are all single segments although their detection is often fragmented into several distinct line segments due to specularities and other effects. This is only possible through segment merging and continuous updates of the segment end point estimates. The computation time in this case is 47ms per added keyframe (again, 4 iterations with a sliding window size of 5).

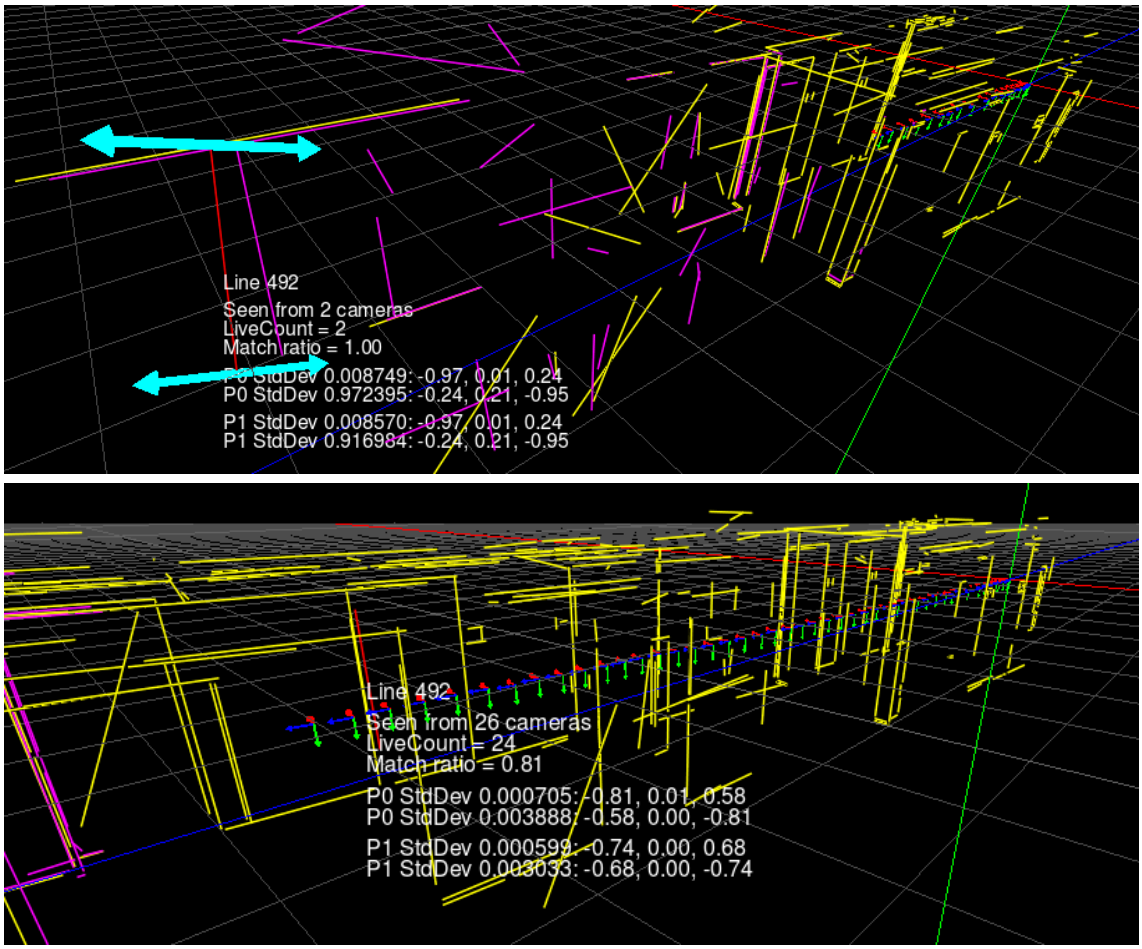



Figure 5.12: The top image shows the initial uncertain triangulation of a line segment. The standard deviation in sightline direction is about $1m$ due to the lack of baseline between the observations (and the initial distance to the camera). As the line segment is repeatedly matched, its location and covariance is restricted by the additional constraints. The lower image shows the same line segment after 26 observations, now having only $0.004m$ standard deviation. Note how inaccurate the more distant stereo matched lines are (magenta). The camera trajectory is shown by the colored coordinate frame.

## 5.6 Discussion

In this chapter we described a complete Line SLAM system that is based on the line segment detection and stereo matching from Section 3.5 (intra-frame matching) and ICML from Section 4.3 for the matching of line segments between keyframes (inter-frame matching). We discussed the general bundle adjustment problem and how it can be solved efficiently. We investigated different line pa-

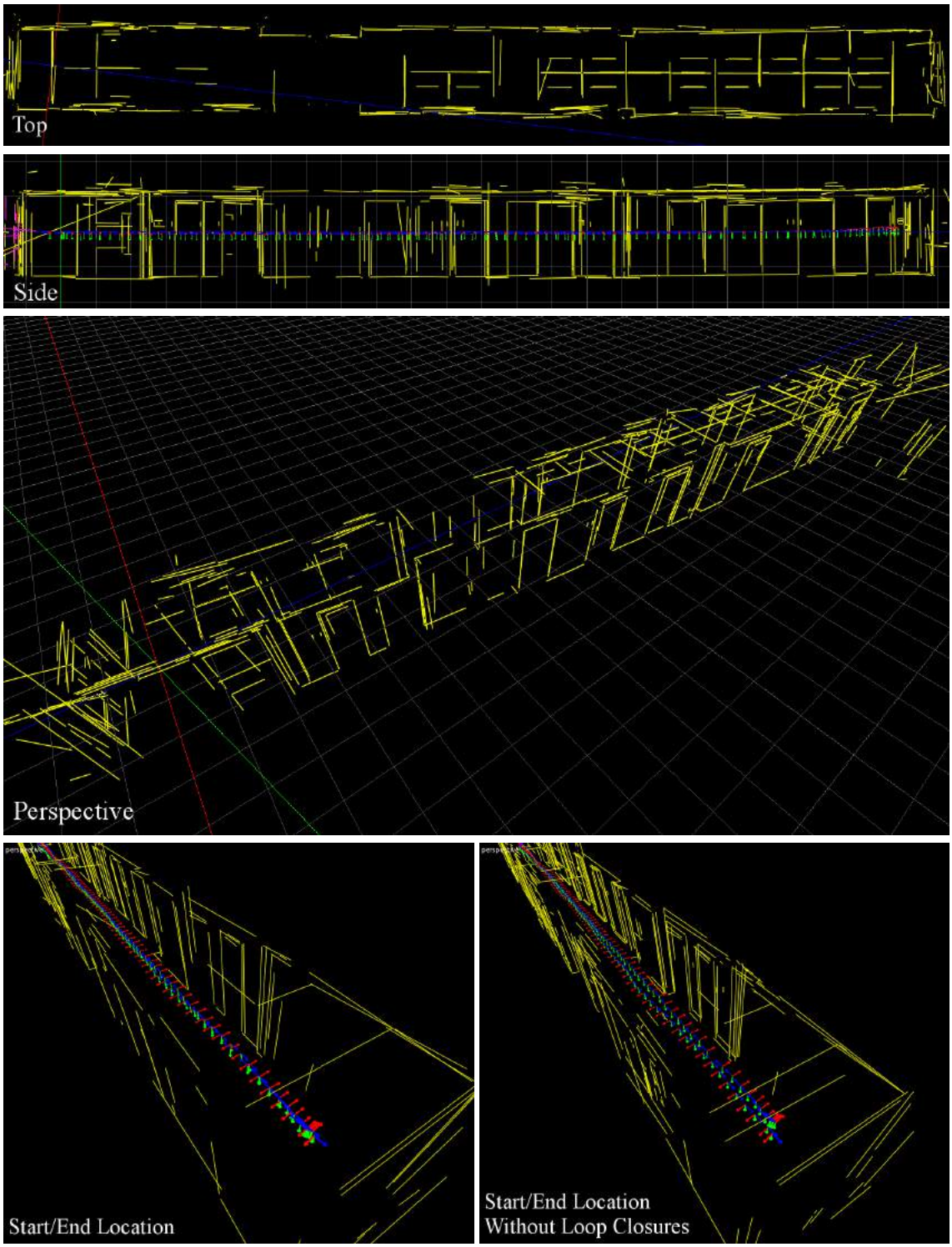



Figure 5.13: Reconstruction result of the Corridor sequence. This experiment demonstrates the loop closure potential of the proposed Line SLAM system, as the geometry is successfully matched during the way back – accordingly undergoing a viewpoint change of 180°. The final reconstruction shows unfragmented line segments for most of the geometry, showcasing successful merging and end point estimation.The bottom row compares the reconstruction at the start/end location with (left) and without (right) loop closures. Note the double contours and trajectory offset in the case of the latter.

rameterizations in the context of estimating a line segment map and proposed the orthogonal relative subspace representation. Since a pure line bundle adjustment only addresses the problem of the estimation of infinite lines, we also proposed several techniques for managing finite line segments. Firstly, as the location of the line segment end points and overall line fragmentation in a given frame can be arbitrary, we handle this strongly non-Gaussian aspect of line segment observations with a heuristic outside of the bundle adjustment. Secondly, it is very important to allow for line segment merging since we cannot assume to find the final parameterization in the first observation. In fact, all experiments had a significant fraction of cases where segment merging was required in order to yield a clean final reconstruction. Finally, we demonstrated the great loop closure potential of the proposed Line SLAM system. Being based only on geometric matching, we are able to match viewpoint changes of $180^\circ$.

Future work includes the extension of the currently implicit loop closures to more robust explicit ones. This will allow for arbitrarily large loops to be closed. Also, a more elaborate keyframe selection can be used in order to speed up mapping by using a lower number of frames overall. The next chapter will extend on the presented Line SLAM system by extracting surfaces from the reconstructed line segment map.

# Chapter 6

# Surface Reconstruction

In this chapter, we propose a novel multi-view method for surface reconstruction from matched line segments with applications to robotic mapping and image-based rendering. Starting from the 3D line maps that were created in Line SLAM as described in the previous chapter, plane hypotheses are formed for all non-collinear and sufficiently coplanar line segment pairs. The surface that is spanned by two segments is used to immediately prune hypotheses that do not pass a sight line occlusion check to keep the initial plane number tractable. After further merging, exhaustive intersections are computed in an efficient way to yield a maximally informative surface representation. Finally, robustified visibility constraints are used to recover a dense surface mesh that is a pessimistic representation of the free space, which is desirable for path planning applications. The presented system is a complete and automatic solution suitable for mapping an environment in real-time scenarios like robotic exploration. We demonstrate the performance of our algorithm on several indoor scenes with varying complexity. This chapter is based on the corresponding publication (Witt and Mentges, 2014) and was created in close collaboration with my coauthor.

## 6.1 Introduction

In many applications, geometric reconstruction from multiple views is an important problem. For image-based rendering and 3D scanning, accurate and visually pleasing models are desired, whereas higher level robotic planning requires concise geometrical representations of an environment. While great progress has

been made with feature point based structure from motion (SfM) techniques in recent years, the reconstruction of scenes containing non-textured surfaces is still a major challenge. Many man-made environments do not even provide sufficient feature points to recover the camera motion – as shown in Chapter 4 – not to mention surface reconstruction. Even partial lack of texture quickly gives rise to holes with conventional techniques (Furukawa and Ponce, 2010). For robotic planning, this is usually not acceptable, which is why active vision systems (e.g. laser scanners) are mostly employed. However despite the lack of point features, the structural information of sparsely or non-textured scenes is still visually deducible from edges (by which we mean intensity gradient maxima). Using piecewise straight line segments as a compact edge representation, we aim to reconstruct completely untextured scenes which are only partially visible in each frame as is the case during indoor exploration.

The proposed surface reconstruction algorithm builds upon the presented Line SLAM system from the previous chapter. The resulting map of line segments along with their visibility graph and covariances is the input for the presented algorithm. Our surface reconstruction approach tries to leverage the maximal information content from the input data by even incorporating planes that derive from only two line segments (in contrast to statistical consensus of multiple primitives). We generate plane hypotheses from coplanar line segment pairs and immediately check visibility of other line segments in the quad area that the candidate segments span. This early pruning is done to keep the plane count tractable. The end point covariances that are obtained from bundle adjustment are used to assign line segments to hypotheses and subsequently merge planes based on common children and a coplanarity criterion. The intersection of the remaining planes yields a three-dimensional grid for which every face is initially assumed to be solid. The free space is determined by robustified visibility constraints that the sight lines from different camera poses to the line segments impose.

Our main contributions are three-fold. First, we propose a novel method for generating plane hypotheses from line segments in $N$ views that is capable of recovering planes from just two coplanar line segments while not creating an intractable number of false hypotheses. This enables the reconstruction of completely untextured scenes for which surfaces may only be partially visible. Secondly, an efficient plane intersection scheme based on a scene graph is defined by

several simple rules: 1. intersections are created for all common line segments between planes, 2. line segments have to be enclosed by intersection points which is enforced by auxiliary intersections that 3. are found in planes that fulfill a spatial or graph-based adjacency criterion. Thirdly, we propose a robust occlusion score for convex faces and the sight triangle that is spanned by a camera origin and line segment end points. This score considers lateral and depth penetration severity to determine the navigable free space from imperfect measurements.

Accordingly, due to the resulting concise geometric representation, the high efficiency that enables real-time use and pessimistic assumptions about the free space, our method is especially useful for robotic exploration.

## 6.2 Related Work

A frequently chosen approach to planar reconstruction is plane sweeping (Furukawa et al., 2009; Gallup et al., 2007; Manessis et al., 2000; Sinha et al., 2009), where plane hypotheses are generated at locations of high feature density along a limited and more or less constrained set of sweeping directions. In (Manessis et al., 2000) the camera image plane is swept across the scene and at every depth location where features were found, a Delaunay triangulation is performed. Because of the triangulation, reconstructed geometry is only indirectly related to the scene geometry and more importantly: reconstruction of every plane in the scene requires a large number $n$ of different perspectives, convergence is shown for $n \to \infty$.

Another approach is to sweep across principal directions of the scene. One way is to assume a Manhattan world as in (Flint et al., 2010; Furukawa et al., 2009; Gallup et al., 2007) with obvious limitations considering scene geometry. More flexibly, non-orthogonal vanishing directions serve as sweeping directions in (Sinha et al., 2009). However, vanishing directions with a low frequency of occurrence may be challenging. Furthermore, plane hypotheses defined only by parallel lines are ignored, which can lead to problems in some environments (e.g. corridors).

Generally speaking, plane sweeping techniques impose rather strong restrictions to the solution space and it is difficult to determine meaningful sweeping directions while not missing relevant but under-implied ones (e.g. untextured

surfaces of which only few edges are detectable).

In (McLauchlan et al., 2000), junction points are searched among input line segments to generate plane hypotheses from connected line pairs. Faces are only created across looping sequences of junctions and thus many faces of the scene may be missing in the reconstruction, since not every junction or even line may be visible or detectable.

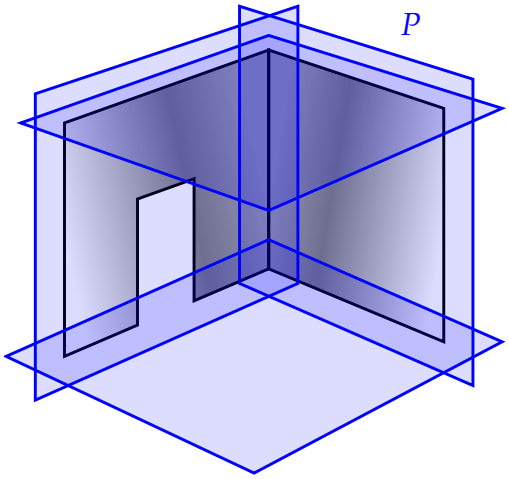
In (Gallup et al., 2010), input images are divided in planar and non planar areas. For planar reconstruction, a RANSAC based scoring method is employed to fit arbitrarily oriented plane hypotheses to dense depth input point data. However, such input may not be available in untextured scenes.

An approach that leverages image-space neighborhood of edge pixels for generating plane hypotheses is presented in (Tomono, 2012). Seed points are distributed across the input image according to a Voronoi diagram of the detected edges, for which 3D coordinates have been determined in a previous SLAM step. For each seed point, RANSAC is used to select the best hypothesis for its supporting edge pixel neighborhood. However, it seems that a fixed threshold irrespective of error estimates is used for hypothesis scoring which is problematic for surfaces in a larger range of distances. Multiple views are reconstructed individually and then stitched together. This makes a reconstruction of textureless regions at image borders difficult.

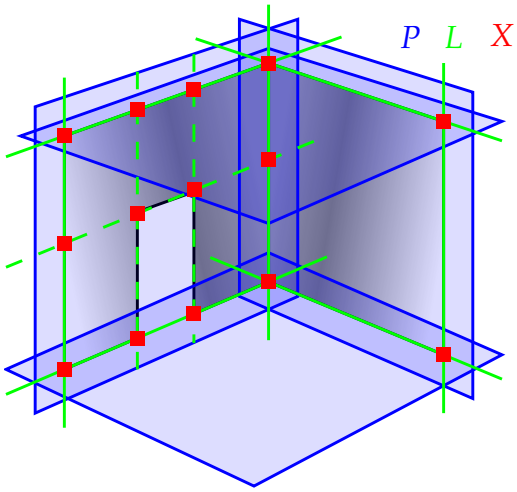
Our approach leverages information from multiple views while considering individual line segment error covariances. Plane intersections provide consistent and concise geometry without unnecessary holes. The use of line segments as sparse input allows our method to be fast enough for real-time applications.

## 6.3 Surface Reconstruction Algorithm

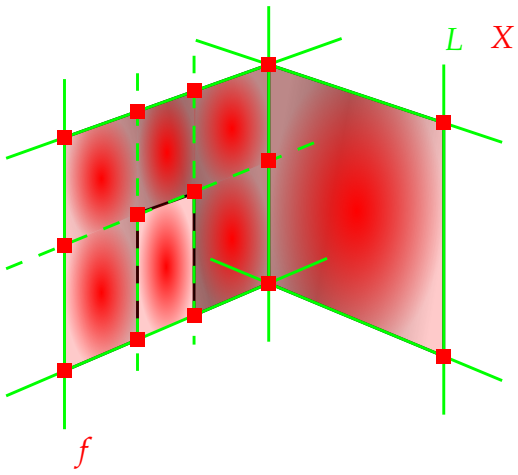
The motivation behind our algorithm is that line segments are concise and geometrically expressive features with which many man-made scenes can be described. The number of planes that can be derived from $n_S$ line segments is $O(n_S^2)$ while from $n_p$ points, $O(n_p^3)$ planes are possible. Pairs of line segments even allow to immediately assess coplanarity as a plane likelihood criterion, whereas any 3 points are coplanar. Another beauty lies in the hinting towards 3D edges. We utilize these hints to compute plane intersections based on common line seg-



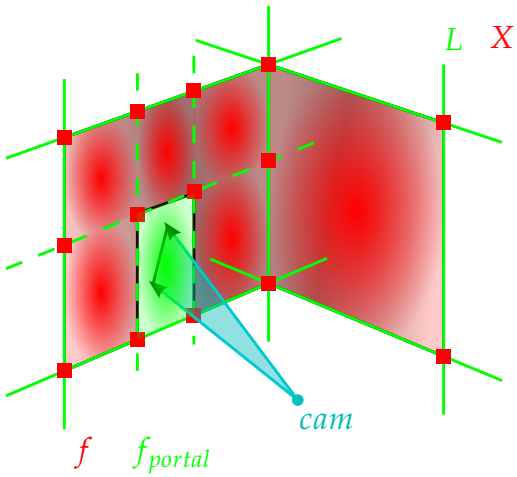

(a) Plane Search ⇒ Planes $P$ for every surface containing $\geq 2$ coplanar segments.




(b) Plane Intersection ⇒ Interconnected grid of intersection lines $L$ and points $X$.




(c) Face creation ⇒ A solid mesh of faces $f$ is generated based on the intersection grid. Faces may span free space.




(d) Occlusion Check ⇒ Classification of faces into solids and permeable portals. Output: surface mesh (polygons).

Figure 6.1: The shown reconstruction pipeline visualizes the steps from generating planes, over the creation of an intersection grid based on line segment relations to the extraction of faces that are culled based on a visibility constraint.

ments between different planes to extract volumes. Figure 6.1 gives an overview of our pipeline from generating plane hypotheses, over plane intersection (based on line segments), the extraction of faces from the intersection grid and finally culling faces that do not pass our robustified visibility constraint.

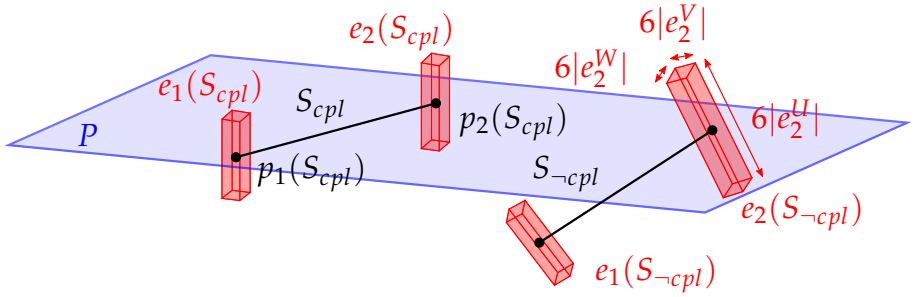



Figure 6.2: A line segment $S$ is considered coplanar to a plane $P$, if both error volumes $e_{1,2}(S)$ of its end points $p_{1,2}(S)$ intersect with $P$.

### 6.3.1 Line Segments and Coplanarity

Each line segment $S$ reconstructed by the Line SLAM is defined by its end points $p_1(S)$ and $p_2(S)$ in Euclidean coordinates and corresponding error description. For simplification, the errors of the segment end points are represented by bounding cuboids $e_{1,2}(S)$ around the Gaussian error ellipsoids, spanned by the error bases $\{e_{1,2}^U, e_{1,2}^V, e_{1,2}^W\}$ which correspond to the eigenvectors of the point covariance matrices estimated during the SLAM optimization procedure. The resulting error cuboids are scaled to $3\sigma$ in each direction to obtain volumes, which contain the ideal points to a high certainty of 99.7%.

For a line segment $S$ to be considered coplanar to a plane $P$, the condition

$$cpl(S,P) = (e_1(S) \cap P \neq \emptyset) \wedge (e_2(S) \cap P \neq \emptyset) \tag{6.1}$$

has to be fulfilled, i.e. both of the described error cuboids $e_{1,2}(S)$ have to intersect with $P$. Accordingly, two segments $S_{1,2}$ are called coplanar, if

$$cpl(S_1,S_2) = \exists P : cpl(S_1,P) \ \wedge \ cpl(S_2,P). \tag{6.2}$$

When searching for plane hypotheses among combinations of segments, first, a provisional normal $n' = dir(S_1) \times dir(S_2)$ is computed, where $dir(S)$ denotes the normalized segment direction. The error base vectors of each segment point are then projected onto $n'$ to obtain the error $e^{n'} = |e^{U,n'}| + |e^{V,n'}| + |e^{W,n'}|$ in direction of $n'$. The reciprocal square of the projected errors $e^{n'}$ is now used to perform a weighted least squares fit of a plane to the four segment points of the combination. If the fitted plane intersects all error cuboids, a coplanar segment

combination and thus a valid plane hypothesis is found.

Prior to the coplanarity check, segment combinations are checked for collinearity using a similar projection of the error bases onto the relevant end point distance vectors, and checking the error projections for overlap, which implies collinearity. If a combination $S_1$, $S_2$ is collinear, the condition $cln(S_1, S_2)$ holds true. Collinear segments are not considered when checking for coplanarity to ensure geometrical stability of the constructed plane hypotheses.

Line segments $S$ coplanar to a plane $P$ are called the *child segments* of $P$, represented by the set

$$cld_S(P) = \{S \mid cpl(S, P)\}. \tag{6.3}$$

Correspondingly, for each segment $S$ a set $pnt_P(S)$ of *parent planes* is defined, containing all planes to which $S$ is coplanar.

### 6.3.2 Frames and Visibility

For each frame $F$, $cld_S(F)$ defines a set of the visible segments as obtained from the Line SLAM. Analogously, for each segment $S$ a set $pnt_F(S)$ of *parent frames* is defined, which includes all frames $F$, wherein $S$ has been observed. From this, we derive a set of parent frames

$$pnt_F(P) = \bigcup_{S \in cld_S(P)} pnt_F(S) \tag{6.4}$$

for each plane $P$, which is the union of the child segments' parent frames.

The sets $cld_S(F)$ and $pnt_F(P)$ form a graph (Figure 6.3), which is used to constrain interference of geometry by visibility. $pnt_F(P)$ has the important property of providing information about overlapping geometry between frames and thus about regions where frame-wise geometry has to be seamed (Figure 6.3), which is e.g. used to restrict merging and intersecting of two planes $P_1$ and $P_2$ based on the presence of *common parent frames* $pnt_F(P_1) \cap pnt_F(P_2)$ .

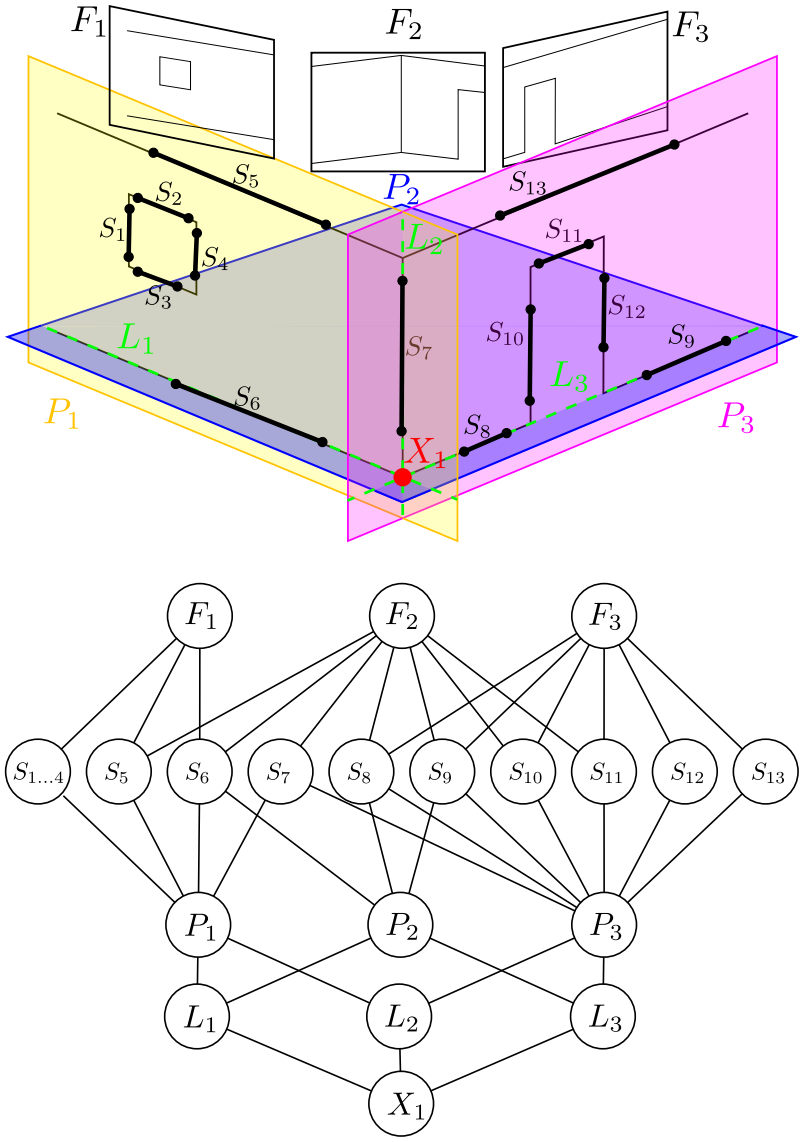



Figure 6.3: A simple scene with three camera frames $F_{1...3}$ from which 13 line segments $S_{1...13}$ are visible. The planes $P_{1...3}$ generate the intersection lines $L_{1...3}$ and intersection point $X_1$ (note that the ceiling plane that $S_5$ and $S_{13}$ would generate has been omitted for visual clarity). The graph represents the resulting relations of visibility and geometry, which are created and assessed during reconstruction.

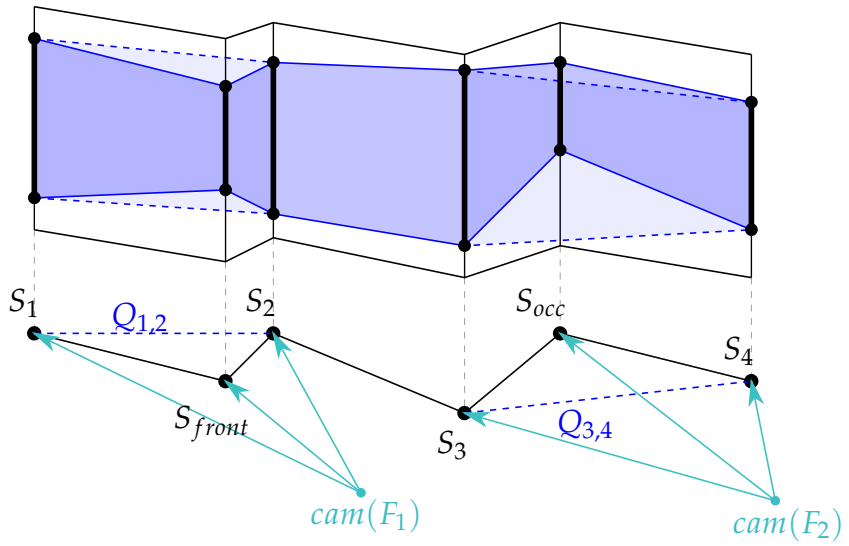



Figure 6.4: Immediate occlusion and contour check for plane hypotheses. The false hypothesis defined by $S_1, S_2$ crosses solid volume and fails the contour test because its spanning quad $Q_{1,2}$ gets occluded by $S_{front}$ and $cpl(S_1, S_{front}) \wedge cpl(S_{front}, S_2)$. This pairwise coplanarity requirement ensures the discarded hypothesis will get enclosed by the resulting planes. The false hypothesis between $S_3, S_4$ crosses free space and will be discarded since its spanning quad $Q_{3,4}$ occludes $S_{occ}$, which may be of arbitrary orientation. Note that most of the 10 possible false hypotheses between the shown parallel line segments have been left out for clarity.

### 6.3.3 Finding Meaningful Planes

#### Frame-wise Plane Search

First, for each frame $F$, coplanar line segment combinations are searched among all $S \in cld_S(F)$. Especially the presence of many parallel segments in the scene can lead to large numbers $O(|cld_S(F)|^2)$ of false hypotheses, since every pair of parallels defines another plane. On the other hand, parallel segments cannot be ignored, since in many situations, we found parallel contours to be the only source of information, such as for untextured corridor floors. To make use of parallels without generating too many false hypotheses, each combination found to be coplanar has to pass a pair of tests to immediately discard hypotheses spanned across free space or solid volume.

To discard hypotheses crossing free space, the spanning quad $Q_{3,4}$ of a combination $S_{3,4} \in cld_S(F)$ is checked for occluding other visible line segments $S_{occ} \in cld_S(F)$ (Figure 6.4). For each occlusion, a score is computed from length, posi-

tion and average back distance of the occluded part of $S_{occ}$, which is accumulated across frames $F$ with $S_{3,4} \in cld_S(F)$. Face occlusion checks and scoring will be explained in detail within Section 6.3.5. If the accumulated score exceeds a threshold, the hypothesis will be discarded. By this approach we achieve robustness against sporadic false occlusions, caused by segment outliers (e.g. reflections).

A similar *contour check* is performed to detect hypotheses spanned across solid volume. Now, it will be checked for $Q_{1,2}$, if it *gets occluded* by a segment $S_{front} \in cld_S(F)$ (Figure 6.4). This frontal occlusion of $Q_{1,2}$ is scored and processed analogously to the back occlusion case. The potentially dangerous process of discarding hypotheses because of features *in front of* the spanning quad $Q_{1,2}$ is made robust by an additional requirement. Such hypotheses may only be discarded, if

$$cpl(S_{front}, S_1) \ \wedge \ cpl(S_{front}, S_2), \tag{6.5}$$

that is the occluding line segment $S_{front}$ has to be coplanar with *both*, $S_1$ and $S_2$. This means, the discarded hypothesis will be *enclosed* by the two resulting hypotheses between $S_1$,$S_{front}$ and $S_2$,$S_{front}$.

False hypotheses may remain undetected, if e.g., $Q_{1,2}$ of a coplanar segment combination is of small extent compared to average distance between segments visible across the scene. However, at this stage, it is not required to discard every false hypothesis, since the final main occlusion check of the algorithm will eliminate non-solid faces generated within false planes.

**Frame-wise Plane Merging**

After plane hypotheses $P \in cld_P(F)$ passing the early occlusion and contour check were found, for each frame $F$ all segments $S \in cld_S(F)$ are checked for coplanarity with the generated hypotheses $\in cld_P(F)$ and added to the respective sets of plane child segments $cld_S(P)$.

Now, that we obtained complete information about segment coplanarity for the current frame, we can make use of this to perform a specific merging of planes, based on common child segments between them. A combination of planes

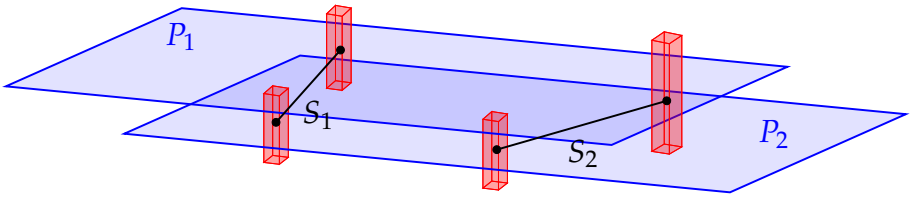



Figure 6.5: Plane Merging. A combination of planes $P_{1,2}$ will get merged, if they have at least one non-collinear segment combination $S_{1,2}$ in common and do not lose any child segments after refitting to the united child segments.

$P_1, P_2 \in cld_P(F)$ will get merged, if it fulfills the condition

$$\begin{aligned} mrg(P_1, P_2) \; &= \exists S_1, S_2 \in cld_S(P_1) \cap cld_S(P_2) : \neg cln(S_1, S_2) \\ &\wedge \forall S \in cld_S(P_1) \cup cld_S(P_2) : cpl(S, P_{mrg}), \end{aligned} \tag{6.6}$$

i.e. if they have at least one *non-collinear* combination of segments in common and the resulting merged plane $P_{mrg}$ still contains all of the united child segments, where $P_{mrg}$ is obtained from performing a weighted least squares fit across the united child segments. When performing merging on a subset of planes, those are sorted by the number of possible merge candidates and merged with descending candidate count. This procedure is iterated until no more mergings are possible.

### Frame-overlapping Plane Merging

After plane hypotheses have been found and merged frame-wise, a further assignment of segments to planes and merging of planes is performed, based on the requirement that they own common parent frames. First, segments $S$ for which $pnt_F(S) \cap pnt_F(P) \neq \emptyset \wedge cpl(S, P)$ are added to the child segments $cld_S(P)$ of the respective planes $P$. Second, planes $P_1, P_2$ having common parent frames $pnt_F(P_1) \cap pnt_F(P_2) \neq \emptyset$ get merged if they fulfill $mrg(P_1, P_2)$. Through this approach, we obtain geometry continuously seamed across adjacent frames, but do not perform any merging of unrelated geometry, which could lead to an unnecessary loss of local detail at higher computational cost.

The sets of common parent frames between planes $pnt_F(P_1) \cap pnt_F(P_2)$ and segments $pnt_F(S_1) \cap pnt_F(S_2)$ can be determined efficiently by searching them among the intersections between the sets of child segments $cld_S(F)$ and child planes $cld_P(F)$ of each frame $F$. In general, a frame based approach is more ef-

ficient, instead of complexities $O(n_P^2)$ for plane merging and $O(n_P \cdot n_S)$ for child segment assignment, we achieve $O(n_F \cdot |cld_P(F)|)$ and $O(n_F \cdot |cld_P(F)| \cdot |cld_S(F)|)$ respectively, where $n_F$ is the number of processed frames $F$. $n_P$ and $n_S$ are the total number of known planes and line segments, where usually $n_P \gg |cld_P(F)|$ and $n_S \gg |cld_S(F)|$.

### 6.3.4 Efficient Robust Plane Intersection

After plane hypotheses have been constructed, it has to be determined, in which way planes get intersected with each other to form a geometrical and topological grid of intersection points $X$ connected along intersection lines $L$.

#### Regular intersections

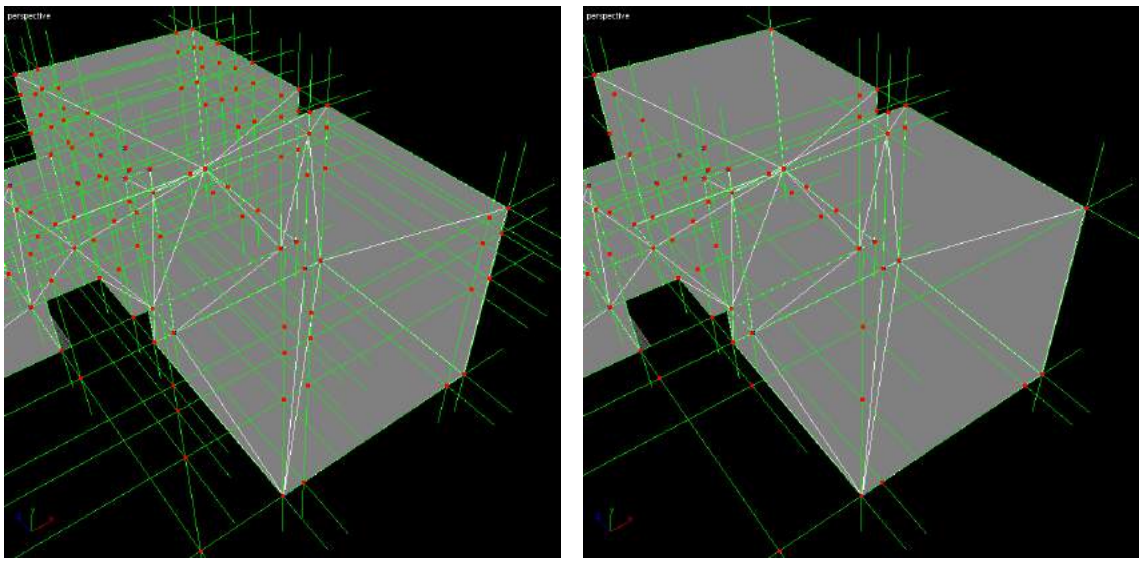
A natural choice for an intersection criterion is the presence of a *common child segment* among two planes, since such a segment corresponds to an observed real edge of the environment geometry. Thus, a combination of planes $P_1, P_2$ will get intersected, if the condition

$$\begin{aligned} cmn(P_1, P_2) &= (cld_s(L_{1,2}) \neq \emptyset) \\ \text{with} \qquad cld_S(L_{1,2}) &= cld_S(P_1) \cap cld_S(P_2) \end{aligned} \tag{6.7}$$

is fulfilled, where $L_{1,2}$ denotes the intersection line between $P_1$ and $P_2$ and $cld_S(L_{1,2})$ is the set of child segments of $L_{1,2}$, generally defined by

$$cld_S(L) = \bigcap_{P \in pnt_P(L)} cld_S(P). \tag{6.8}$$

Through this, every *observed* environment edge will get represented by an intersection line $L$ within the constructed grid. After planes have been intersected according to (6.7), intersections points $X$ are calculated plane-wise from combinations of lines $L_{1,2}$. Note that restricting the plane intersection leads to the existence of intersection points $X$ with less than 3 parent planes $pnt_P(X)$ and lines $pnt_L(X)$ connected to it, while they still intersect geometrically in $X$. This leads to an intersection grid, only containing intersection points, collinear to observed scene edges represented by segments (Figure 6.6).



(a) (b)

Figure 6.6: Comparison of (a) exhaustive intersection and (b) under requirement of common child segments between intersecting planes in the ideal case of observed segments for every scene edge.

Line segments $S_{pln}$ for which $|pnt_P(S_{pln})| = 1$, i.e. which only have a single parent plane, are called *planar segments*. Such segments stem from intensity gradients on planar surfaces, e.g. wood paneling on a wall. To represent those borders, artificial intersection lines $L_{pln}$ are constructed from the projection of those planar segments onto their parent plane, which in turn is intersected regularly with all other intersection lines of the parent plane.

### Line Segment Enclosement

Since we do not assume perfect line detection for every environment edge, we cannot rely on the presence of line segments as cues for plane intersections. This section explains the fundamental rule that we use to determine the need for auxiliary intersections (*after* performing regular intersections according to (6.7)). The main idea is the enforcement of *line segment enclosement* by intersection points $X$ along all intersection lines $L$ (see Figure 6.7). This is motivated by the fact, that the final mesh is a 3D-grid of connected intersection point pairs and only these sections of intersection lines can make up the face edges of the final reconstruction. Accordingly, since we assume that every line segment is relevant, we have

to enforce enclosement by intersection points to assure that all line segments are represented in the resulting reconstruction.

Only the outermost segment end points $p_{min,max}(S \in cld_S(L))$ along each intersection line $L$ need to be checked for enclosement, since the line segments in between are naturally enclosed as well. Therefore, an intersection line $L$ is *enclosed* if and only if the condition

$$\begin{aligned} enc(L) \; &= \exists X_1 \in cld_X(L) : \mu_L(X_1) \geq \mu_L(p_{max}) \\ &\wedge \exists X_2 \in cld_X(L) : \mu_L(X_2) \leq \mu_L(p_{min}) \\ &\wedge |pnt_L(X_{1,2})| \geq 3 \; \wedge \; |pnt_P(X_{1,2})| \geq 3 \end{aligned} \tag{6.9}$$

is true, where $\mu_L(p)$ is the scalar line parameter of a point $p$ that corresponds to the closest point on $L$, $cld_X(L)$ denotes the set of child intersection points along $L$ and $pnt_L(X)$ the set of parent intersection lines crossing in an intersection point $X$.

The sub predicate $|pnt_L(X_{1,2})| \geq 3 \; \wedge \; |pnt_P(X_{1,2})| \geq 3$ requires an enclosing intersection point to stem from the intersection of at least 3 planes, instead of just from intersections between 2 or more intersection lines of a single plane (e.g. of planar line segments). The reason for this (as shown in Figure 6.7) is that only one of the $L$-adjacent faces $f$ would be closed if an intersection point $X$ is not a three-dimensional, but a planar intersection. However, an auxiliary intersection of the parent planes $pnt_P(L)$ with a further plane $P_{aux}$ always yields enclosing intersection points $X_{aux}$ fulfilling this requirement.

**Auxiliary Intersections**

If $enc(L)$ evaluates to false for any line $L$, the best plane $P_{aux}$ for an auxiliary intersection has to be found. In general, we want to generate the tightest possible enclosement that minimizes the distance between $X_{aux}$ and an unenclosed segment end point $p(S)$. For this, we search among the planes of the following groups:

1. common parent frame: $pnt_F(L) \cap pnt_F(P_{aux}) \neq \emptyset$,
2. spatially close: $\exists P \in pnt_P(L) : close(L, P, P_{aux})$,
3. artificial scene bounding planes.

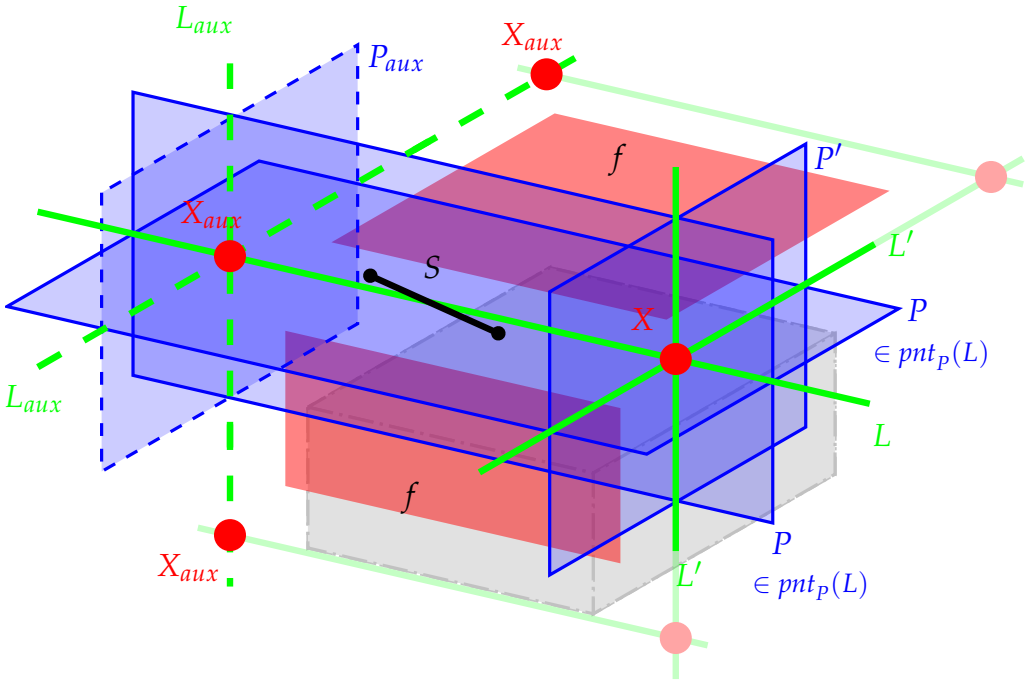



Figure 6.7: To obtain a consistent 3D intersection line grid that represents all segments $S$, it is essential to enforce intersection points $X$ on both segment sides. The lack of intersection points is usually a result of undetected line segments (e.g. due to lighting or occlusion). We solve this problem by searching for an auxiliary plane $P_{aux}$ to generate the missing intersection lines $L_{aux}$ and points $X_{aux}$ (dashed). The requirement for intersection points to have 3 parent lines and planes guarantees that all $S$-adjacent faces $f$ can be closed and thus 3D volumes are terminated by planes to all sides.

While the first group of planes is close in terms of their scene graph relation, the spatial closeness of planes has to be defined. Since a plane $P$ is spatially related to its child segments, we define the spatial adjacence of planes by their segment bounding boxes

$$\begin{aligned} close(L, P \in pnt_P(L), P_{aux}) \; &= ||(L \cap P_{aux}) - C(P)|| < \lambda \cdot R(P) \\ &\wedge \; ||(L \cap P_{aux}) - C(P_{aux})|| < \lambda \cdot R(P_{aux}), \end{aligned} \tag{6.10}$$

where $C(P)$ denotes the center and $R(P)$ the radius of the bounding box of all child segment end points $p_{1,2}(S \in cld_S(P))$ and $\lambda$ serves as a safety factor (chosen dependent on the observed/expected line segment density).

Often, surfaces like untextured walls extend beyond the field of view and

cannot be enclosed by planes that where generated from the visible geometry. If those faces shall be reconstructed (as is desirable in most cases, especially robotics), artificial bounding planes in all 6 principal directions need to be introduced. We parameterize these planes to form a bounding box around all detected line segments, scaled by a safety factor.

#### Face Creation

After obtaining an intersection grid from the found plane hypotheses, convex faces $f$ are searched along the planar sub grids of intersection lines by employing a left turning search.

### 6.3.5 Main Occlusion Check And Solid Surface Extraction

Now, we have obtained a 3D-intersection-grid spanned with faces, that subdivides the reconstructed geometry into separate volumes. Since we initially assume that all faces are solid, we recover the free space by enforcing the visibility constraint. However, hard culling of faces that are penetrated by sight lines can be problematic with real world data. Reflections, inaccuracies or outliers could quickly destroy geometry. We use a robustified occlusion score $occ(f)$ that factors in the severity in lateral and depth direction to assess the likelihood that a face $f$ actually needs to be culled. After all faces have been scored in this way, the free volume is obtained by thresholding. By this approach we obtain a pessimistic estimation of the navigable space that surrounds the robot.

#### Segment Sightline Projection

For every parent frame $F \in pnt_F(P)$ of a plane $P$ all visible child segments $S \in cld_S(F)$ with at least one end point behind $P$ are projected onto $P$, with the exception of plane child segments $S \in cld_S(P)$. The sight line projection $S^P$ of a segment $S$ with end points $p_{1,2}(S,F)$ is defined by the segment between the projected end points $p^P_{1,2}(S,F)$, which are obtained by intersecting the sight lines

$$L^s_{1,2}(\lambda) = cam(F) + \lambda(p_{1,2}(S,F) - cam(F)), \quad \lambda \in [0;1] \tag{6.11}$$

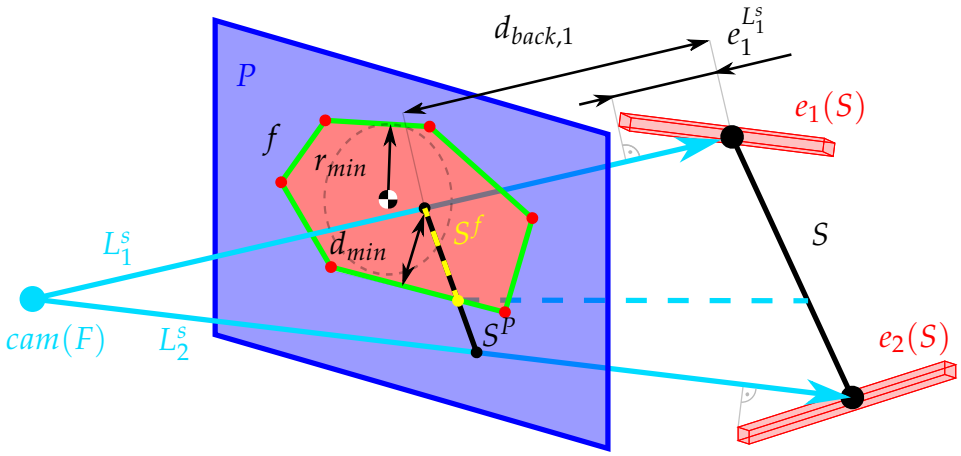



Figure 6.8: Every face $f$ is checked for occlusions of line segments. Our occlusion score uses the border distance $d_{min}$ and penetration depth $d_{back}$ to compute an occlusion severity $\in [0, 1]$. The border distance $d_{min}$ is normalized by the maximal inner face circle radius $r_{min}$ and $d_{back}$ by the segment error in sight line direction.

with $P$, where $cam(F)$ is the camera position for frame $F$. Segments intersecting $P$ are trimmed to the part behind $P$ before sight line intersection. Figure 6.8 visualizes the involved geometry. Note, that every segment $S$ can have different and multiple sets of end points $p_{1,2}(S, F)$ per parent frame $F$, since different parts of a segment can be visible in each frame, e.g. due to occlusions or unfavorable lighting.

Once the projection $S^P$ of a segment $S$ onto $P$ is computed, it will be used to perform the occlusion check for all faces $f \in cld_f(P)$, i.e. all faces among the planar sub grid of intersection lines $\in cld_L(P)$.

### Occlusion Score

To perform the check for a single face $f$, first, the face-overlapping sub segment $S^f$ is determined by intersecting $S^P$ with the border edges of $f$. If $S^f$ exists, an occlusion score

$$occ(f, S) = \frac{d_{overlap}(S^f, f) \cdot d_{back}(S^f, S, f)}{|pnt_F(f)|} \tag{6.12}$$

is computed, where $d_{overlap}$ is an *overlap score* $\in [0; 1]$ weighting the face border distance of the planar overlap of $S^f$ with $f$ and $d_{back}$ is a *back distance score* $\in [0; 1]$

weighting the average back distance of the part of $S$ behind $S^f$ with respect to the camera $cam(F)$. The score is normalized by the number of parent frames $|parent_F(f)|$ of $f$, i.e. the number of frames the face has been visible in, where $parent_F(f)$ is defined by

$$pnt_F(f) = \bigcap_{L \in L(f)} \underbrace{\bigcap_{P \in pnt_P(L)} pnt_F(P)}_{pnt_F(L)}, \tag{6.13}$$

wherein $L(f)$ denotes the set of lines defining the border of $f$ and $parent_F(L)$ the parent frames of a line $L$.

**Overlap Score**

Let $e_i(f)$ be the edges of a face $f$ spanned between intersection points $X$ , then the score

$$d_{overlap}(S^f, f) = \min \left\{ 1;\ d_{min}(S^f, f)/r_{min} \right\} \tag{6.14}$$

with

$$d_{min}(S^f, f) = \min_{e \in e_i(f)} \left\{ \max_{p \in \{p_{1,2}(S^f)\}} \{d(p, e)\} \right\} \tag{6.15}$$

relates border distance of $S^f$ to the radius $r_{min}(f)$ of the biggest circle around the centroid $ctr(f)$ of $f$. The value $d_{min}(S^f, f)$ is the minimal distance of $S^f$ to any of the face edges $e_i(f)$, where $d(p, e)$ is the distance of an end point $p$ of $S^f$ to an edge $e \in e_i(f)$. By relating to $r_{min}(f)$, $d_{max}$ is weighted evenly along every direction for faces of varying aspect ratio.

**Back Distance Score**

By linear interpolation between the end points of a segment $S$, the back distances $d_{back}(S^f)$ and the sight line parallel errors $e_{back}(S^f)$ of the segment part behind $S^f$ (i.e. the part occluded by $f$) are computed from the back distances $d_{back,1,2}$ and sight line parallel errors $e_{1,2}^{L_{1,2}^s} = \sum_{R=U,V,W} |e_R^{L_{1,2}^s}|$ of $S$. Let $\bar{d}_{back}(S^f)$ be the mean back distance and $\bar{e}_{back}(S^f)$ the mean error behind $S^f$, then the back distance score

is defined as

$$d_{back}(S^f, S, f) = \min\left\{1; \frac{\bar{d}_{back}(S^f)}{\bar{e}_{back}(S^f)}\right\}. \tag{6.16}$$

Visually spoken, $occ(f, S)$ expresses the depth error relative back distance, weighted by the amount of overlap between the truncated segment projection $S^f$ and face $f$, averaged over the frames the face $f$ is visible in. By this averaging, $occ(f, S)$ gains robustness against occurrences of implausible sight lines, e.g. caused by segment outliers.

After having computed $occ(f, S)$ for each face $f$, we can now robustly determine non-solid portals $f_{portal}$ by thresholding with $occ_{max}$. In our experiments $occ_{max} = 0.25$ proved to deliver good results. This translates to an ideal occlusion with $occ(f, S) = 1$ needing to occur every $4th$ frame a face has been visible in.

**Complexity**

With the present implementation, the check is performed by iterating linearly over all faces of a plane, yielding an execution time $O(|cld_f(P)|)$ per plane. This could be reduced to $O(\log|cld_f(P)|)$ by employing a binary space partitioning of the planar grids, ideally using the existing intersection lines as the partitioning hyperplanes.

## 6.4 Experimental Results

For the experiments, we used the same sequences and stereo camera as in the previous chapters. The experiments were run on a single core of an Intel Core i7 with 2.8 GHz without GPU acceleration. Table 6.1 lists reconstruction time measurements (excluding Line SLAM).

The Room sequence from Figure 6.9 is the most extreme example of scene sparsity that we present in this evaluation. The white walls have large areas with bright moving specularities which may even provide false information for dense stereo or multi-view reconstruction approaches (Furukawa and Ponce, 2010; Newcombe et al., 2011). Due to the small distance to the wall, the camera is only able to capture excerpts of the whole geometry (see the frustum plot that shows the

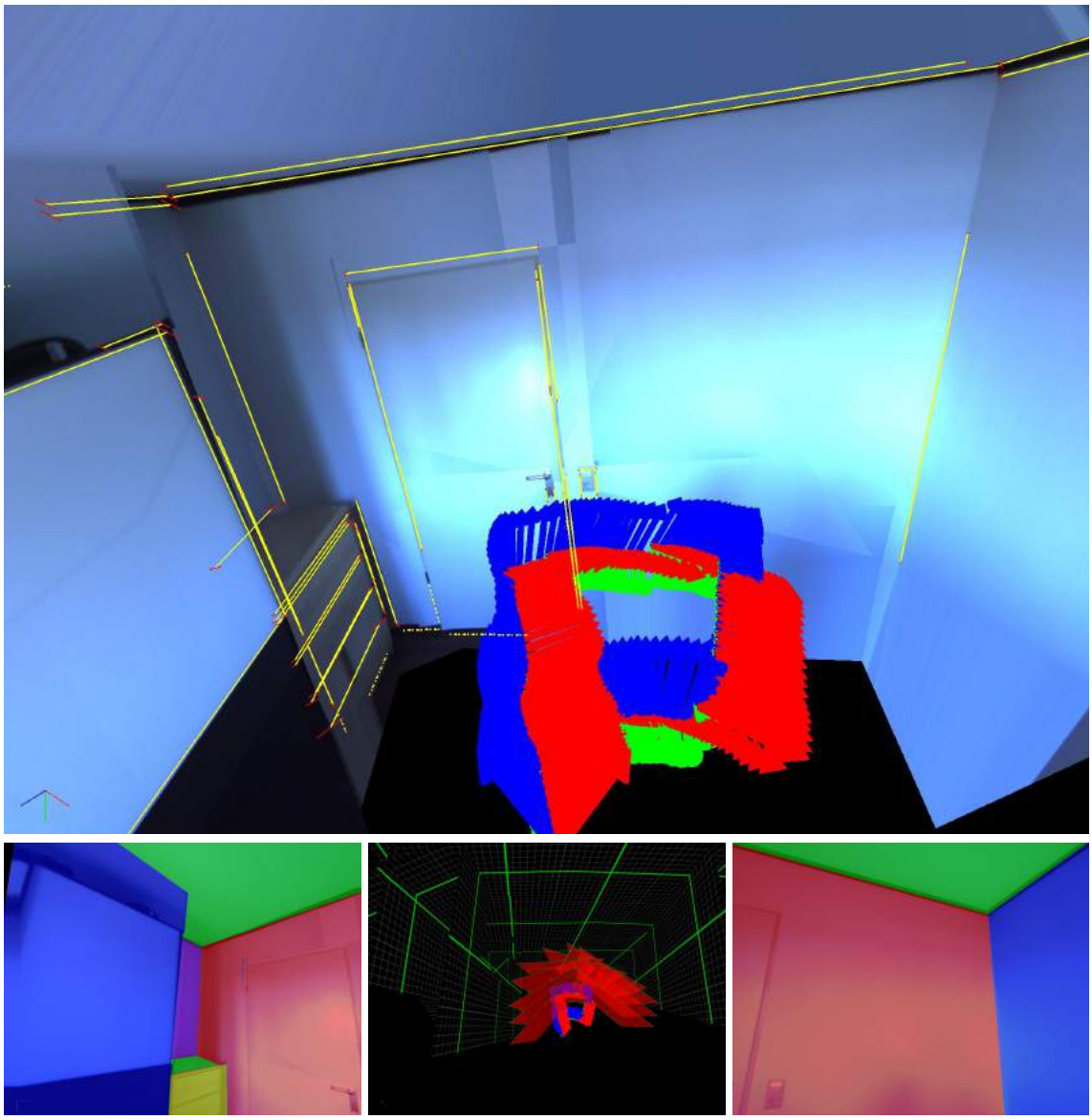


Figure 6.9: The Room sequence has virtually no texture, large areas with specularities and all frames have only partial scene visibility. The image row shows the pose/frustum plot (center) and two input images that have been colored based on the normal direction of reconstructed planes. The upper image shows a rendering of the complete reconstructed scene with overlaid camera poses (blue arrow is the look at direction) and detected line segments (with error cuboids in red).

trajectory from above). The plots with colored plane direction overlay exemplarily show two frames of the sequence. The large textureless border regions are especially challenging to reconstruct for other methods. The number of frames could be significantly reduced in this case, but we generally favor large numbers

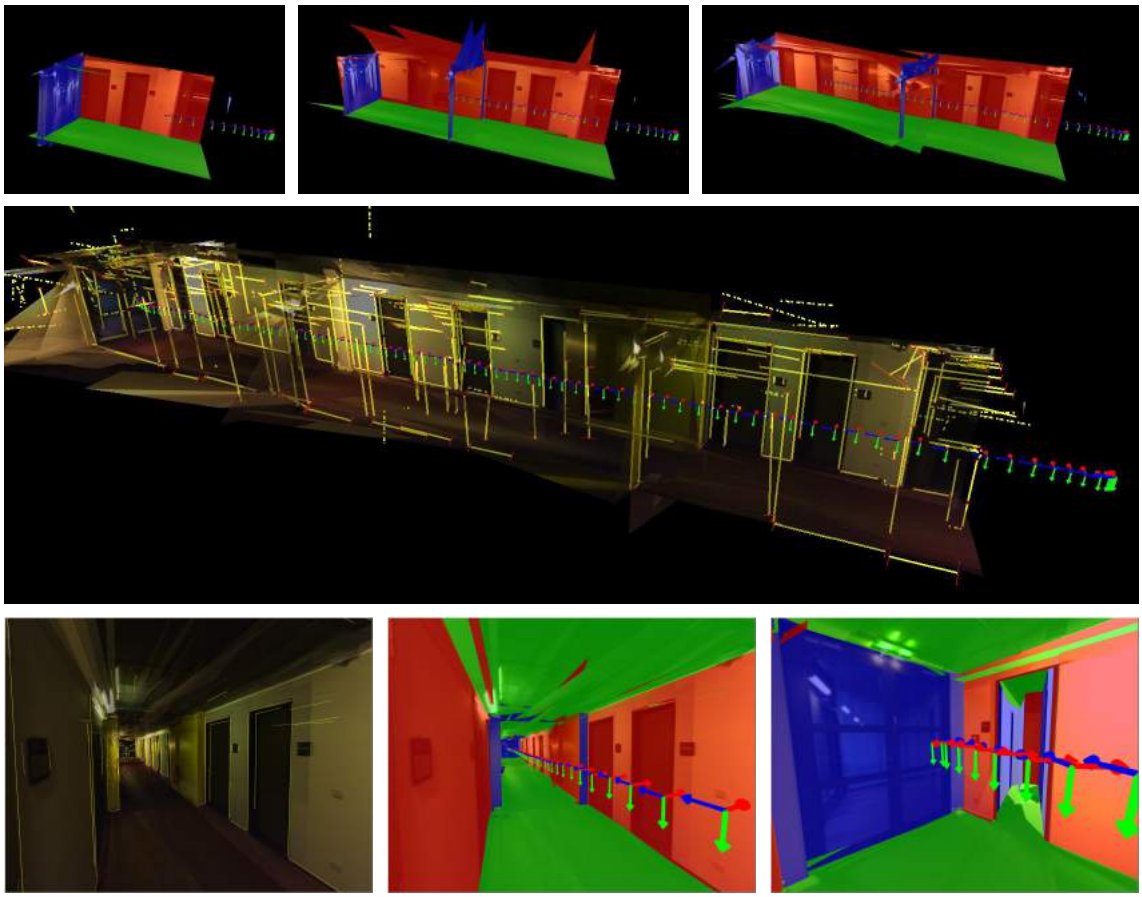


Figure 6.10: The first row shows reconstructions of the Corridor sequence for increasing numbers of frames, as one would compute during robotic exploration. The middle plot shows the texture-mapped result with overlaid line segments (backfaces culled, as in the top row). The bottom row shows renderings from novel viewpoints inside the corridor (coloring shows plane normal direction). Note the open door in the bottom right rendering.

of sight lines to be able to cull more non-solid faces in our visibility test. Due to the low geometric complexity, the average computation time was only 0.5ms per frame.

The first half of the Corridor sequence consists of 90 frames from a 24m long trajectory in a moderately textured environment with reflections in the metallic ceiling and ceramic floor (see Figure 6.10). The top image row illustrates how the reconstruction evolves during exploration and new sight lines continuously carve out the free volume. The reconstruction successfully recovers the two 20cm ledges in the middle of the corridor (blue plane direction color) and the open door with portions of the dark room behind it (right image in bottom row).

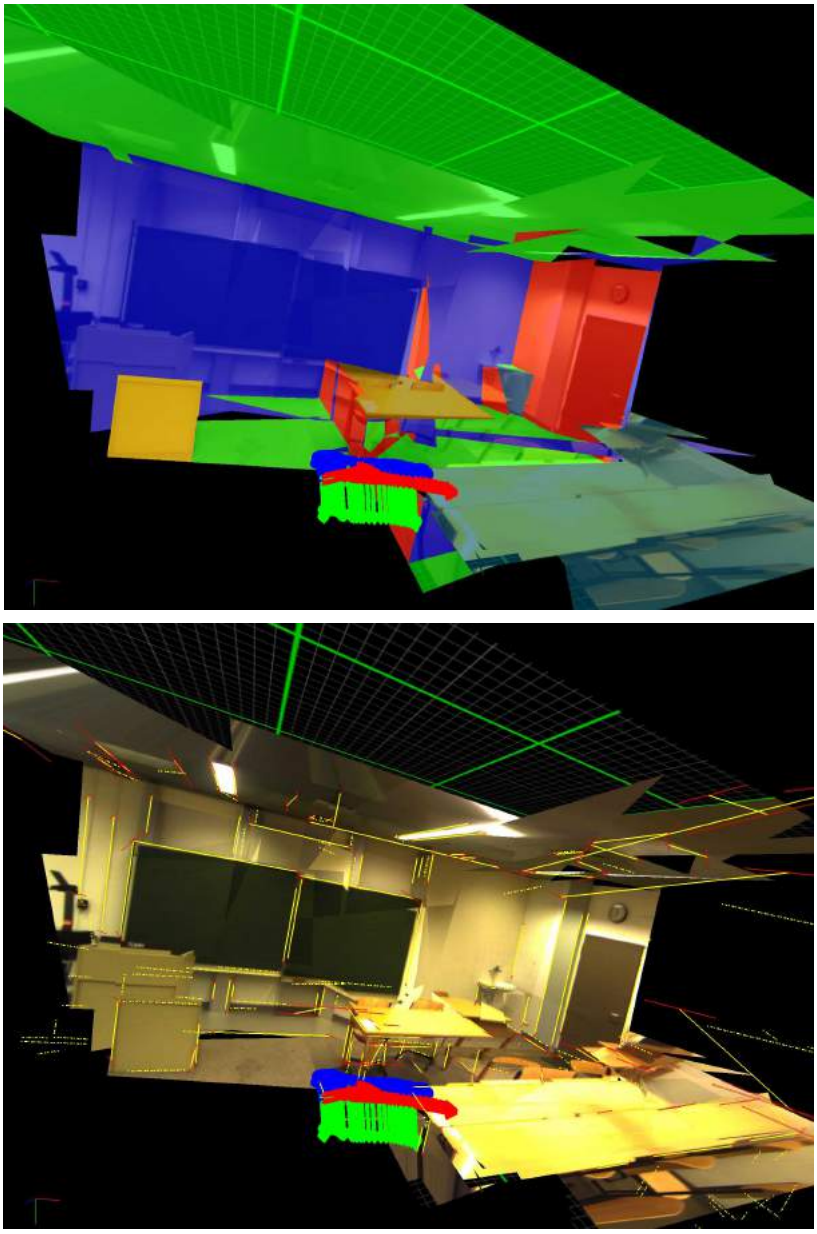


Figure 6.11: The upper rendering visualizes the different plane normal directions in the reconstructed Big Room sequence. The free space is not entirely accurate, as some non-solid faces are not violating the visibility constraint from the given frames. However, the reconstruction is still usable for robotic planning, and as a robot continues to explore, it will generate more sight lines to carve out the free volume.

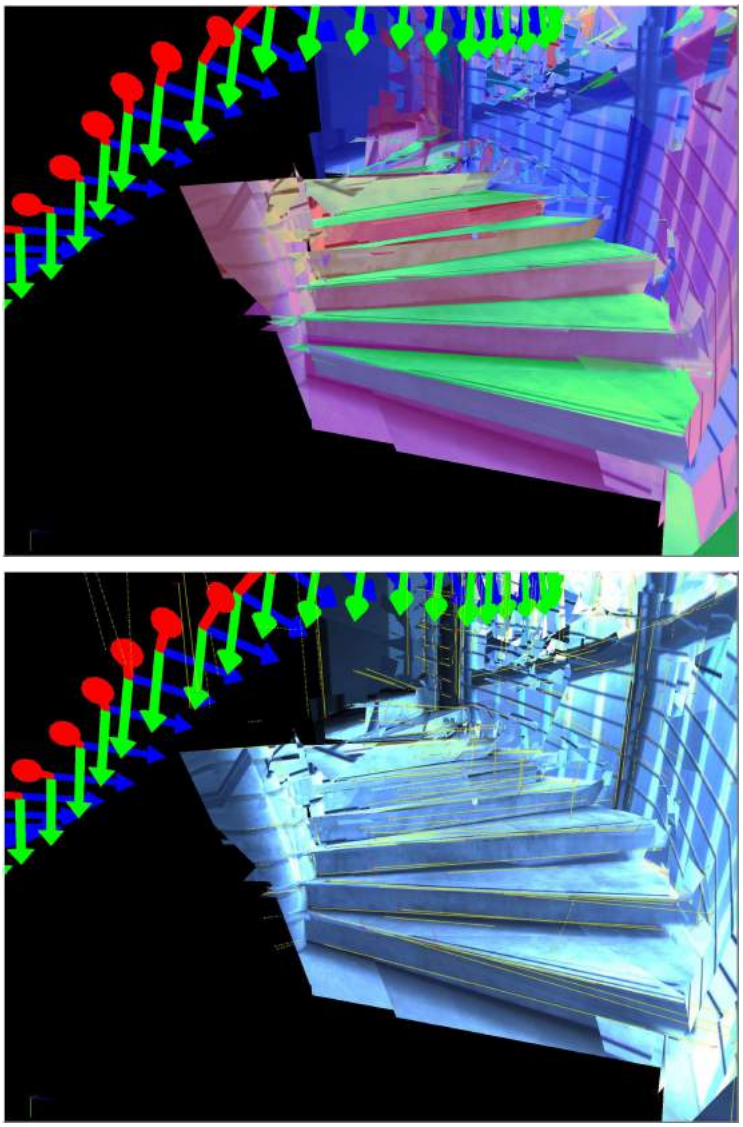


Figure 6.12: Renderings of a reconstructed spiral staircase. Note the curved wall approximated by tangential planes. The coloring reflects the plane normal orientation.

The Big Room scene in Figure 6.11 is geometrically more complex. It is challenging that the line segment errors are relatively large due to the increased viewing distance and short camera trajectory (see the red error cuboids). However, this is a common situation for robots. Also, due to various occlusions and lighting, our Line SLAM cannot track all line segments, which leads to an incomplete line map. While the resulting reconstruction is not entirely accurate, the correct planes are found and a robot could continue exploration while generating addi-

Table 6.1: Execution Times per Frame by Pipeline Stage

| | Room | Corridor | Big Room | Spiral Stairs |
|---|---|---|---|---|
| Frames | 239 | 90 | 44 | 40 |
| Plane Search | 0.1ms | 1.6ms | 2.1ms | 0.4ms |
| Intersection | 0.1ms | 1.6ms | 1.1ms | 9.2ms |
| Face Creation | 0.1ms | 1ms | 0.1ms | 7.2ms |
| Occlusion Check | 0.2ms | 3.1ms | 2.8ms | 53ms |
| Total per Frame | 0.5ms | 7.3ms | 6.1ms | 70ms |
| Total | 109ms | 655ms | 265ms | 2.73s |

tional sight lines to improve its estimate of free space.

Finally, the Spiral Stairs sequence demonstrates the reconstruction of curved surfaces and various different plane orientations (Figure 6.12). Of course, this is limited to curved surfaces for which a sufficient number of line segments can be detected. However, the sequence shows that the concept of our algorithm is general enough to be applied to such challenging cases.

## 6.5 Discussion

We presented a fast reconstruction method that can infer geometry from line segments with a pessimistic assumption about the free space. Plane hypotheses are efficiently generated and an intersection grid is created based on child line segment relations. The use of auxiliary plane intersections to close open contours allows reconstructions even when facing partial line detection failures. With this scheme, we try to leverage the maximal information content. However, the quality of the result still depends on the success of the Line SLAM. Accordingly, highly textured objects have to be handled with conventional reconstruction methods. Another issue concerns the visibility constraint. The number of sight lines from line segments can be insufficient in some scenes and we may not be able to cull some non-solid faces that resulted from the intersection step. Future research will explore the addition of a simple dense stereo matching algorithm to generate supplementary sight lines for sufficiently textured regions. This will also allow to reduce the number of required frames significantly. The currently achieved reconstructions are already very useful for structured scenar-

ios and could be even widened in scope, if point features are added to gain a fast hybrid approach.

The field of suitable applications include image-based rendering and other reconstruction tasks where a simple scene model needs to be acquired. The prime focus during development has been on robotic exploration, though. Both the achieved processing speed and the avoidance of holes in the reconstructed geometry lend itself to this problem.

# Chapter 7

# Conclusions

Visual navigation has come a long way. Many approaches exist that are able to solve the SLAM problem robustly in well-textured environments. Also, real-time dense reconstruction has been shown in these settings. In comparison, untextured environments are still a blind spot for many approaches. Striving to contribute complementary methods that provide self-sufficient solutions to such difficult cases, we focused on the utilization of edge information as we showed that the most extreme untextured scenes may not contain any other visual cues.

Having started on edge pixel chains, we found that line segments are a more concise yet informative way to parameterize edges. While line segments condense straight edges especially well, curves can also be expressed. As a higher level primitive, an important advantage of line segments over feature points is their geometric expressiveness. We therefore dedicated the majority of this work to line segment based formulations.

## 7.1 Discussion

Building upon a well-known edge detection algorithm, we proposed approaches for all subsequent stages leading up to our full Line SLAM system with dense reconstruction. First, we discussed the nature of edges in the context of sparse stereo matching and derived efficient methods that take advantage of edge connectivity information. Our comparison showed that the error rates were on a par with much more computationally expensive dense stereo methods. Finally being interested in matched line segments, we ultimately found direct matching

of these primitives far more efficient without having to compromise on quality in our target environments. A prerequisite for this to be successful is solid line segment detection. An important observation in this respect was, that, also here, the utilization of edge connectivity information contributes significantly to both computational efficiency and algorithm performance. In our qualitative comparison on indoor images, our line segment detector showed superior results over state-of-the-art histogram based approaches (i.e. Hough variants).

To solve the motion estimation problem, we proposed a novel reprojection optimization for line segments. We have shown that our 1-to-N matching is more robust than simple 1-to-1 matching and argued that the unreliable nature of line segment end points calls for such an approach. A distinction to most other matching techniques is that ours is purely geometric and thus completely independent of scene texture. Furthermore, since photometric changes do not affect matching (as long as line segments can be detected), given a sufficient pose prior, our matching can cope with large viewpoint changes.

For online motion estimation in an unknown environment our method currently requires a stereo camera. Nevertheless, the reprojection formulation can also be applied to monocular problems like model tracking or localization in a known map. A usecase for the latter could be automated cars which do only operate on prerecorded, verified maps for safety and alignment reasons.

Since many robotic applications do not only require motion estimates, but also the creation of maps and localization therein, we developed a complete line SLAM system. We pointed out the practical nuisances of line segments that need to carefully be addressed. In particular, the continuous estimation of segment end points plays a key role in building up long tracks. As a direct result, the bundle adjustment benefits from fewer and more consistent line segments with an increased number of constraints on average. For the optimization itself we proposed a minimal line parameterization that enables an easy transition to line segments. Furthermore, we demonstrated loop closures under significant viewpoint changes. We have not seen similar results in other visual SLAM systems and believe that appearance-based approaches in general will struggle in such cases.

In comparison to feature point maps, our reconstructed line segment maps have a high geometric expressiveness. This is why we developed a dense surface reconstruction method that leverages the favorable properties of line segments.

Foremost, we argue that the number of permissible plane hypotheses from line segments is orders of magnitude smaller than from feature points. This makes an exhaustive plane search tractable. Hence, we do not need to make any restricting assumptions about the geometry. We even allow planes that originate from only two noncollinear segments. We found this to be essential in our pursuit to recover as much geometry as possible from the few visual cues that untextured environments provide. However, despite the promising results with our sequences, for the current version, the scope of our surface reconstruction method is clearly limited to scenarios that are dominated by straight structures. Cases where the line segment map becomes too incomplete are detrimental – e.g. due to objects with complex shape or strong, high frequency texture. Nevertheless, the fast runtime allows the use for real-time indoor exploration and mapping applications.

## 7.2 Future Work

Widening the scope to a multi-purpose visual navigation system, several possibilities for future research arise. The integration of feature points to form a hybrid system is the first enhancement that comes to mind. This would allow to handle all cases from heavily textured to the sparsest scenes that we successfully handle with the presented methods.

Minor improvements with respect to convergence speed and range in ICML could be achieved by a simplistic line feature matching, e.g. by comparing the average gradient magnitude. Such a weighting of matches would decrease the influence of opposite sign edge matches and steer convergence in the right direction from more coarse initial guesses. First experiments in this direction showed promising results. Considering more sophisticated line features would have a significant impact on processing times and we anticipate that the viewpoint change robustness would decrease considerably.

Moving towards the deployment of our Line SLAM on a robot, the current loop closure implementation has to be revised to an explicit one to be suitable for large loops. For this, a fast geometry-based line segment similarity detector needs to be added. We believe that we will only be able to retain the demonstrated viewpoint independence if we stay within the geometry domain for this purpose. Since our Line SLAM is quite successful in reconstructing unfragmented line seg-

ments, geometric histogram descriptors of the immediate line segment neighbors could be promising. These would then be suitable for a bag-of-words approach to achieve fast and scalable performance.

Another possible field of future research could lie in an adaptation of our system towards monocular SLAM. If an initialization phase is added during which line segments are tracked by e.g. an optical flow technique, we would be able to continue tracking with ICML based on the initial segment reconstructions. New line segments would continuously be added in the same way. However, this naturally restricts the permissible camera trajectories in order to be able to triangulate new line segments. It has already been mentioned in previous research on monocular line SLAM approaches that a purely linear camera motion is not sufficient to initialize all possible edge orientations.

Finally, a tight integration of our dense reconstruction approach into the SLAM pipeline could significantly improve our long-term map management in the face of occlusions. Currently, dense reconstruction is merely a post-processing step. However, we could use all the extracted relations and surface geometry to predict whether line segments should be visible from a certain viewpoint. This way, we could quickly reason that a previously seen object has disappeared and promptly remove it from the map. On the other hand, we will retain line segments that are in our view frustum but occluded by geometry. Another extension would be possible with regards to the robustness to highly textured objects. If a dense stereo reconstruction is used to augment the line-segments as input, hull geometry for unstructured objects (e.g. plants) could be generated in order to yield a complete assessment of free space.

# Appendix A

# Quad-Rotor Robot System Architecture

This chapter presents the results of an effort to create a high performance quad-rotor MAV named iQCopter for research in the field of swarm robotics and vision-based autonomous operation. During the design, navigational and computational capabilities have been of major priority. A distinctive feature of the iQ-Copter is its layered system architecture using a comparably powerful, hard real-time capable x86-computer even in the innermost control loop for maximum transparency and ease of use while a simple fixed-point microcontroller provides a low-level sensor interface and a fallback solution for safety reasons. Finally, a successful system identification and subsequent design of an aggressive $H_\infty$ controller are presented as benchmark cases to demonstrate the performance of the design. This chapter is based on the corresponding publication (Witt et al., 2011b) and was created in close collaboration with my coauthors.

## A.1 Introduction

Micro unmanned aerial vehicles (MAVs) as a research platform have received strong attention in recent years. Whether for swarm experiments in a closed environment with external tracking (How et al., 2008; Michael et al., 2010) or as fully autonomous agents with on board intelligence (Bachrach et al., 2010; Grzonka et al., 2009; Hoffmann et al., 2004; Meier et al., 2011), many different testbeds have been proposed. The quad-rotor setup with its simple construction, vertical

Figure A.1: The iQCopter quadrotor MAV. The highest point is the IMU to achieve the largest possible separation between the actuator currents and the magnetometers. The iQCopter measures 72cm from rotor tip to rotor tip and weighs about 1250 grams including a 3300mAh 3-cell lithium polymer battery.

take off and landing (VTOL) capability and quick response due to its unstable dynamics make it a frequently selected choice for UAV systems.

Our interest is in fully autonomous outdoor and indoor operation with single and swarming MAVs performing object transportation tasks and search-and-rescue operations in disaster areas. This poses demanding requirements on the capabilities of the testbed. High computational performance is required to be able to incorporate vision based simultaneous localization and mapping (SLAM) and obstacle avoidance algorithms while a solid inertial navigation system and capable actuators are needed to realize aggressive control algorithms. The iQCopter (short for intelligent QuadroCopter) has been designed to provide a highly integrated easy to use research testbed that is able to meet these requirements (Witt et al., 2011a).

### A.1.1 Related Work

Several approaches to autonomous research grade quad-rotors and even more projects exist. This section describes a selection of successful projects to cover common choices that are made in the design phase.

Since quad-rotors enjoy great popularity not only in the research community, several open-source projects exist. Grzonka et al. (2009) use an open-source project as a starting point to develop a fully autonomous quad-rotor equipped with a more accurate commercial IMU and a laser range finder executing SLAM on a wirelessly connected computer. An ARM based system is used for on board

control and data streaming. The commercial IMU is limited to 120 Hz.

For the RANGE project of the MIT, a commercial MAV platform is used (Bachrach et al., 2010). Also carrying a laser range finder, but a more powerful computer, it is able to execute SLAM on board of the MAV. In most commercial systems the inner loop is a black box for the researcher. This is convenient in some respect, but can be a struggle if one is interested in accessing signals at the innermost loop at high sampling rates for identification or controller development. This can even be infeasible.

The Stanford STARMAC (Hoffmann et al., 2004) is based on a commercial IMU and a custom airframe. On board, two 16-bit microprocessors are used for control and communication. The utilized IMU is limited to 76 Hz.

The CSIRO X4-flyer (Pounds et al., 2006) is a large custom built research quadrotor with an IMU running at 50 Hz. A 16-bit microcontroller closes the inner loop and streams data via Bluetooth.

For the Pixhawk project of the ETH Zürich, a new system was developed from the ground up including a custom IMU, airframe and integrated powerful x86 computer for on board stereo vision and pattern recognition (Meier et al., 2011). The inner loop with state estimation and controllers runs on an ARM-based microcontroller. The x86 computer is only running soft real-time tasks.

The system in this work is based on a commercial airframe (Conrad Electronic, 2011) equipped with custom electronics. An 8-bit microcontroller acts as a low-level interface to all sensor components. The main architectural difference to the other systems is that loop closure is achieved through a powerful, hard real-time capable x86 computer (running Linux with real-time extensions), allowing to run all filter, control and also non-deterministic algorithms in parallel. A newly developed, script-driven real-time software framework routes signals to reusable and parameterizable modules. For research this setup has several benefits:

- All data flow is completely transparent, configurable and can be recorded at high rates
- The innermost loop can use extensive algorithms and run in double precision
- Modules can be developed for a known environment and be immediately deployed

- Familiar scientific software libraries can be used
- Stable fallback solutions on the 8-bit microcontroller provide safety against implementation errors during trials

The outstanding results of the linear system identification and controller experiments support the claim that a potent system has been designed.

### A.1.2 Outline

This chapter is laid out as follows. The system architecture is outlined in Section A.2 and proceeds with a description of the hardware components in Section A.3. Section A.4 briefly describes the linear system identification with a grey box SIMO model and subsequent design of an aggressive $H_\infty$ controller to demonstrate the performance of the system. Finally a discussion of the results and an outlook on future projects is given in Section A.5.

## A.2 MAV System Architecture

The driving aspects for the iQCopter were performance, size/weight, ease of use, safety/reliability and cost. Foremost a high computational and navigational performance has been identified as being crucial as a step towards fully autonomous operation. Accordingly, a potent x86 board computer and a high-fidelity MEMS IMU have been chosen as the foundation. To adhere to the goal of high navigational performance on the software side a high fidelity state estimator and controller algorithms preferably running with double precision and a sufficiently high sampling rate should be integrated.

Using a fixed-point microcontroller for this task, however, is problematic. Carrying out double precision floating point algorithms is usually highly inefficient or even impossible. Additionally, even if it is possible it introduces a burden for the researcher concerned with the development of complex algorithms to cross-compile for a proprietary platform, where familiar scientific libraries might not work and which offers low transparency during runtime and lacking debugging comfort.

An implementation on powerful computer hardware which runs an operating system is more convenient in that respect but problematic for safety and reliabil-

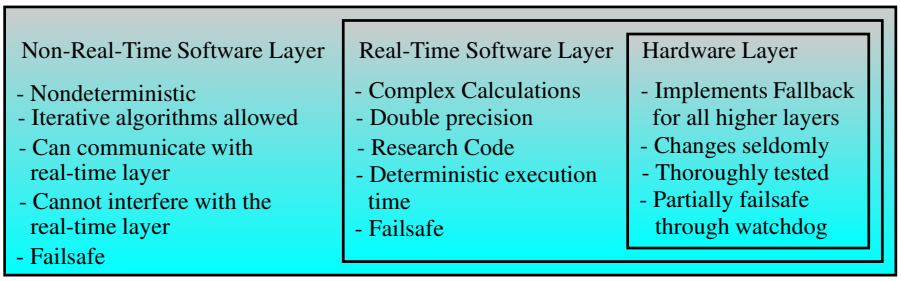



Figure A.2: The layered structure of our system design. Through the fallback mechanisms in the hardware layer, experimental code can be run in the software layer to accelerate research progress without sacrificing safety.

ity reasons. Although erroneous code can also lockup the simplest microcontroller it does not suffer from determinism problems such as a computer running a regular operating system such as Linux or Windows. Another challenge arises with the ease of use of a computer. If it is easy to implement new functionality a researcher tends to do so and of course occasionally introduces conceptional mistakes or implementation errors. For safety critical functionality this is fatal for the whole system in the case of a MAV.

### A.2.1 System Layers

To include the best aspects of a simple microcontroller and a powerful general purpose computer into the concept, the system architecture of the iQCopter consists of three hierarchic layers, each providing certain properties as shown in Figure A.2. The innermost layer is the ”hardware layer” consisting of an 8-bit microcontroller which is connected to all critical sensors, actuators and a long range RF module. It is responsible for timing, the acquisition of sensor data and subsequent transfer to the computer board. This is illustrated in Figure A.3.

Safety, ease of use and performance are guaranteed by the combination of a solid fallback solution running on the fixed-point microcontroller in the hardware layer and the deterministic real-time software layer running more extensive algorithms on the computer. In case the main controller process on the computer is locked up or exited accidentally, the fallback solution automatically takes over within milliseconds. This way even software for the innermost control loop can easily and safely be researched while having the comfort of a familiar environment with advanced debugging and development tools.

Algorithms with nondeterministic execution time like many vision algorithms or iterative optimization are prohibited in the real-time layer, though. This is to be handled in the non-real-time layer. An important point is that a non-real-time task can never interfere with a real-time task even if it stresses the processor to 100%.

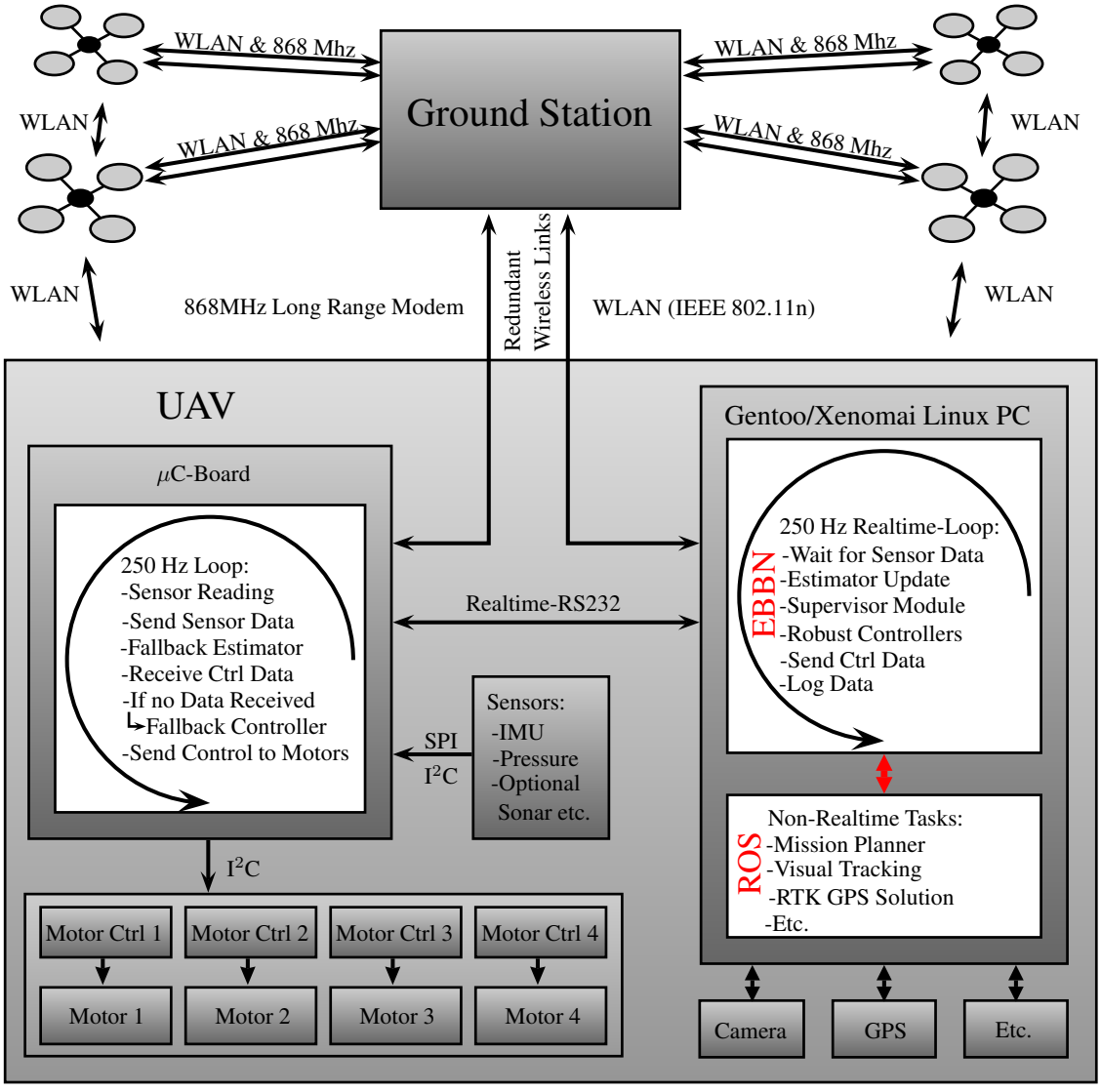



Figure A.3: The system architecture of an iQCopter communicating with multiple MAVs and a ground station. The two 250Hz loops are synchronized via the RS232-Link. If the board computer fails, the $\mu$C-Board is capable of both attitude estimation and control as a fallback solution.

### A.2.2 Real-Time Software Framework

Having introduced the layered system architecture it was necessary to bring the endeavor to achieve high transparency and ease of use to the software level on the computer. This was realized by the development of a novel real-time capable framework named Executable Building Blocks Network (EBBN). It is a comprehensive suite of basic blocks such as math modules, hardware interfacing modules, control and state estimation modules etc. to form a runtime coalition for which the routing and parameterization is completely defined by configuration files.

## A.3 Hardware Setup

### A.3.1 Computer Board

The current iQCopter is equipped with a 1.6GHz Intel Atom embedded computer (consuming about 5W), with plans to upgrade to a laptop-level multicore computer board in the future. The board computer is easily replaceable in this system setup since no hardware specific code is executed on the computer. The ATMEL ATmega128 microcontroller functions as an abstraction layer between sensors and computer. The only hardware requirement for the computer is a RS232 serial port to form the real-time link to the microcontroller. The processing power of the Atom processor is sufficient to run all state estimation and control algorithms in double precision at 250Hz while running a pattern recognition algorithm with 400x300 pixels at about 5Hz in parallel.

The deployed operating system is a real-time enabled Gentoo Linux since the control loop shall be closed via the board computer and thus requires deterministic timing. In order to make the Linux kernel real-time capable it is augmented with the Xenomai real-time framework (Gerum, 2004). During load tests the latencies where measured to never exceed $30\mu s$ which is well below the requirements considering $4ms$ period time.

### A.3.2 Sensors

The most important sensors in a MAV are the inertial sensors like gyroscopes and accelerometers, since they provide the essential data for computation of the

Table A.1: Quantities of the integrated Analog Devices ADIS16405 IMU. It also comprises a 3-axis magnetometer in the same package, weighing only 16 grams in total. Laid out as an intelligent sensor it provides temperature compensation and low pass filtering but not a complete state estimation solution.

attitude. Commercially available complete inertial measurement units (IMU) including sensor data fusion were considered not appropriate for this project due to high price, weight and slow sampling rates (50-100Hz). The development of an IMU based on analog inertial sensors is challenging, though. Temperature sensitivity and high requirements for amplification and A/D-conversion complicate a high-performance solution. As a compromise an integrated intelligent sensor from Analog Devices' iSensor family has been chosen: the ADIS16405 IMU. In a small package it houses high-accuracy temperature compensated MEMS accelerometers, gyroscopes and magnetometers for a reasonable price. Some specifications are given in table A.1. Sensor readings are transferred via a SPI interface. Data is internally acquired at 819.2Hz and filtered by a discrete low-pass configured to 17 taps improving gyroscope noise levels to 0.0045°/sec RMS and changing their bandwidth to 32Hz.

For sensor data fusion a simple yet efficient quaternion based Kalman filter has been implemented, running at 250 Hz. Although the sensor is high-end in the range of MEMS IMUs it is still important to estimate and compensate accelerometer and gyroscope biases during flight for satisfying performance. Additionally, a hard-iron calibration of the magnetometers has been done after the MAV assembly. The magnetometer biases are not estimated at flight time.

Position measurements are supplied either by a Real-Time Kinematic GPS (RTK-GPS) solution that is described by Pilz et al. (2011) or by visual measurements with a single camera or a stereo camera system. The RTK-GPS solution is based on a $\mu$-Blox LEA-6T chip passing raw data to the computer which iteratively determines the solution. At the time of writing both GPS-based and vision-based position control have successfully been tested in first flight tests.

### A.3.3 Wireless Communication

The wireless communication capabilities comprise a 802.11n WLAN solution to provide a high performance mesh network structure between all MAVs and the

ground station and a redundant long range 868Mhz link to provide a fallback solution.

### A.3.4 Flight Frame and Actuators

For the flight frame and motor controllers the commercially available Conrad Quadrocopter 450 (Conrad Electronic, 2011) toy quadrotor was chosen as the basis for the reason of easy and inexpensive availability of spare parts. The motor controllers are commanded via $I^2C$-bus in contrast to standard brushless controllers which only accept a rather inaccurate pulse width modulated signal as input. The motors are Robbe ROXXY 2827-34 brushless outrunner motors which drive 10″x4,5″ airscrews providing a thrust up to 550 gram per motor with a 3-cell lithium polymer battery. The flight time is about 15 minutes with a 3300mAh battery and a total airborne weight of 1250 grams.

## A.4 Performance Benchmark

In the following, two benchmark cases are briefly presented, which demonstrate the ability of the design to achieve remarkable performance with real flight experiments. All practical experiments strongly benefit from the highly modular, transparent and configurable EBBN framework, which has sped up development progress.

### A.4.1 System Identification of the Roll and Pitch Axis

An important step for a research platform is the acquisition of an accurate system model for the purpose of model-based controller design and simulation. Unfortunately this is usually very difficult for quad-rotor MAVs since identification signals cannot be fed into an uncontrolled and unsupervised flying vehicle. For a successful identification one ideally needs complete transparency down to the innermost control loops. This can be a problem with commercially available UAV platforms which only provide a blackbox solution for the stabilizing control loops or only slow sampling rates are provided externally. Even if the controller parameters are available it might pose problems to route the desired signals of a closed system to a data logger.

With EBBN and the presented system architecture, the innermost control loops with all signals are at the disposal of the researcher who can easily route all the desired signals to a data logger or the wireless communication system. It has been helpful to be able to quickly reparameterize or exchange stabilizing controllers during identification.

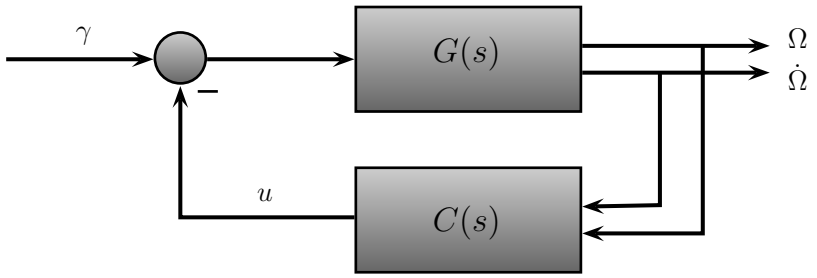



Figure A.4: The closed-loop SIMO identification setup for a single axis. The identification signal $\gamma$ is comprised of 20 periods of a PRBS-sequence to reduce noise influences.

This section cannot comprehensively discuss the system identification for brevity. However, Ljung (1999) provides an excellent reference for closed-loop identification methods and Bouabdallah et al. (2004) for modeling quad-rotors. Instead the successful identification attempt for the roll and pitch axis is exemplarily described. For this, the indirect method is used where a stable closed-loop model is identified including the known controller $C(s)$. Afterwards the unstable open-loop model $G(s)$ can be extracted. Figure A.4 shows the setup for a single axis. The identification signal $\gamma$ is a pseudo random binary sequence (PRBS) signal with a bandwidth of approximately 20Hz and a peak-to-peak amplitude of 0.15, which corresponds to 15% of the total valid motor control range. During the identification and validation experiments the iQCopter was fixed in all but the identification axis to be able to apply several unsupervised periods of the same PRBS-sequence without crashing the vehicle. The identification and validation data sets were acquired by averaging over all periods to reduce noise. Since 20 periods were averaged the signal-to-noise ratio improved by the factor 20. This was necessary since during initial experiments it was found that noise from vibration and process noise were significant.

Figure A.5 shows the final identification result. The model is the result of an iterative identification, since the feedback controller $C(s)$ here is already an $H_\infty$-controller of 8th order which in turn was designed with a model that was ac-

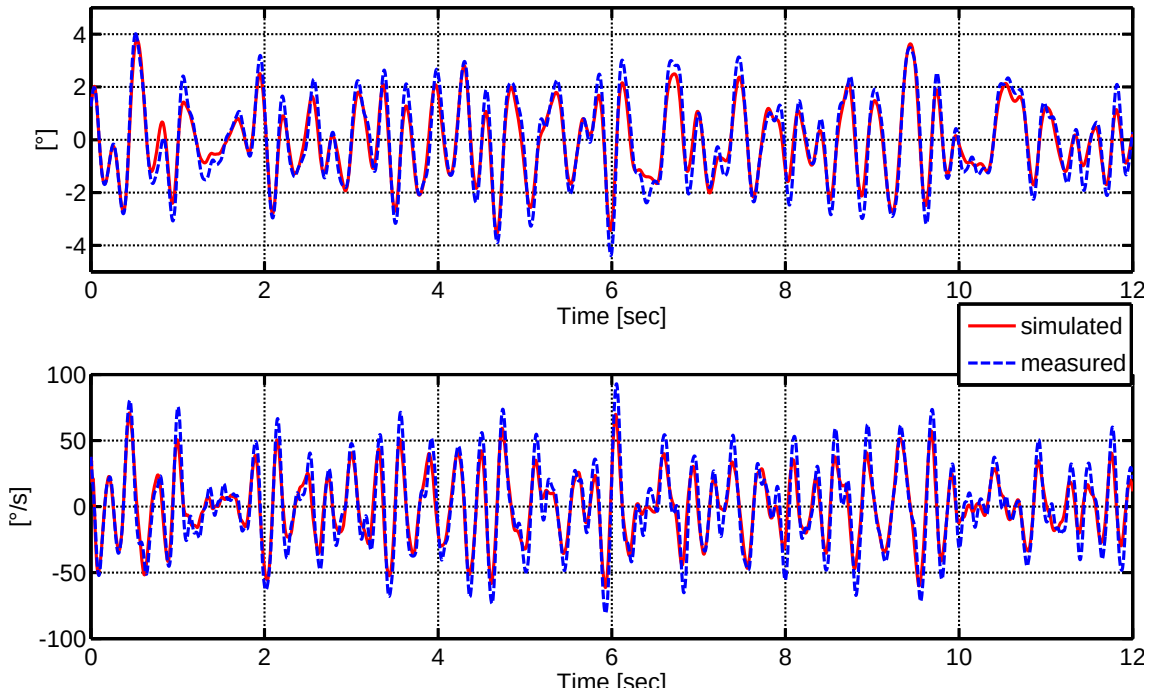



Figure A.5: Validation data set of the final model for the pitch and roll axis. Due to the unstable quad-rotor dynamics the simulation was run with the same stabilizing controller as during the experiment. From the plots of both outputs one can attest the model a very good approximation of the dynamics.

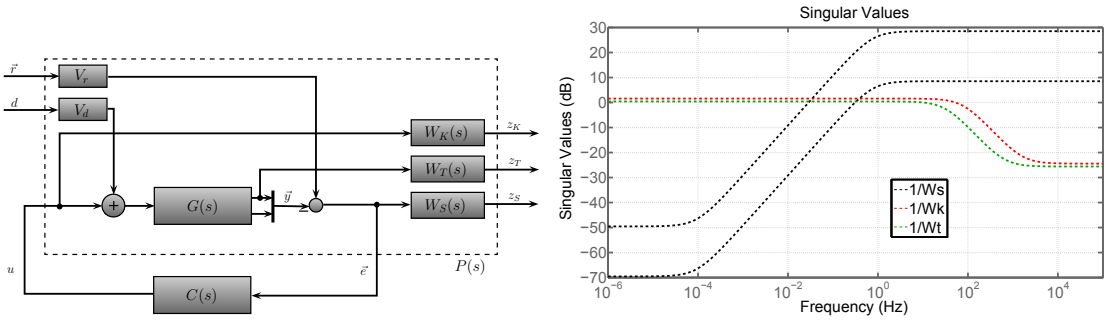
quired similarly, but with a hand-tuned PD-controller. The depicted single input multiple output (SIMO) model has been obtained by optimization of a grey box model with second order motor dynamics, an approximated delay of 32ms and integral behavior for the angular rate and double integral behavior for the angle output:

$$G(s) = \begin{bmatrix} \frac{1}{s^2} \\ \frac{1}{s} \end{bmatrix} \frac{5.31(s+215.93)(s+131.56)}{(s+33.92)(s+13.14)} e^{-0.032s}$$

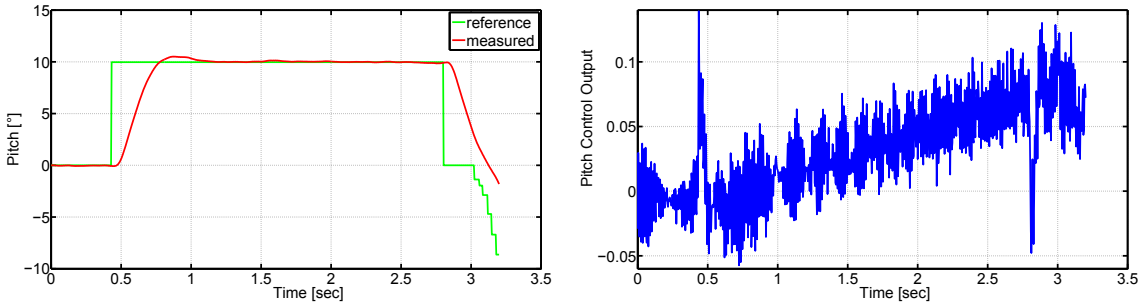
The acquired model resembles the system dynamics very well as will be proven with the design of an aggressive $H_\infty$ attitude controller for the pitch and roll axis in the next section.

### A.4.2 An $H_\infty$ Controller Design for the Roll and Pitch Axis

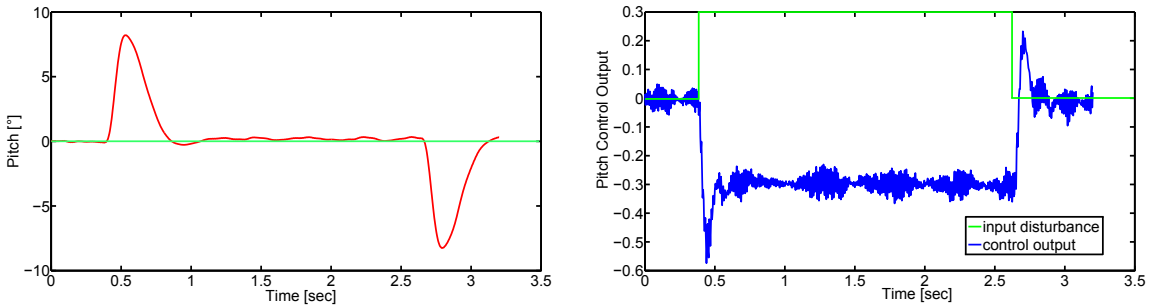
Once a sufficient model has been acquired, model-based controller design techniques can be applied. Many UAVs rely on hand-tuned PID controllers due to the




(a) The generalized plant (left) and the singular values of the shaping functions (right) that were used for the $H_\infty$ design.




(b) Step response of the pitch axis of an airborne iQCopter.




(c) Input disturbance of 0.3 applied to the pitch axis of an airborne iQCopter.

Figure A.6: These plots prove the exceptional control performance of the iQ-Copter with its $H_\infty$ controller. The input disturbance of 30% of the total control range is a rather extreme example of the ability of the controller design.

lack of a good system model. In Pounds et al. (2006), Bouabdallah et al. (2004) and Hoffmann et al. (2004), linear model-based techniques are applied to quad-rotors. Due to model inaccuracies the results are not optimal, though.

The presented controller uses both, the angle and angular rate output, to achieve aggressive controller performance. This has been the driving idea behind the identification of a SIMO model. The generalized plant setup and shap-

ing functions are shown in Figure A.6(a).

The performance of the resulting MISO $H_\infty$ controller for the pitch and roll axes is shown in Figure A.6(b) and A.6(c) which depicts unfiltered plots from a free flight experiment. The improvement over the previously used hand-tuned PID controller has been found to be significant due to the high static gain that is achievable for both reference tracking and input disturbance rejection cases while keeping the overshoot as low as 6%. The rise time was optimized to about 300ms, accordingly the controller cycles about 75 periods meanwhile at 250Hz sampling rate. Although this might seem excessive we have observed that this yields superior results when compared to 50Hz sampling rate. This is probably due to the amplitude discretization that is necessary to generate the motor controller input. The oversampling effectively generates a higher resolution for actuator control. The designed controllers have quickly been incorporated into the UAV as state space models which are loaded from a Matlab generated file easing the transition from simulation to practical experiment.

## A.5 Conclusion

This chapter has presented the iQCopter, a high-performance quad-rotor MAV with a layered system architecture to provide an easy to use testbed with high transparency down to the innermost control loops and mechanisms to ensure operational safety. With the successful model-based design of an aggressive controller it was shown that 1) the inertial measurement solution performs well, 2) the loop closure via a computer at 250Hz was successfully implemented and functions safely, 3) the linear model that was acquired represents the quad-rotor attitude dynamics well and 4) the physical setup with the chosen actuators and motor drivers is sufficient.

# List of Abbreviations

| Abbreviation | Description |
|---|---|
| AGAST | Adaptive and Generic Accelerated Segment Test |
| CPU | Central Processing Unit |
| DOF | Degree Of Freedom |
| EBDP | Edge-Based Dynamic Programming |
| EMCBR | Edge Matching by Confidence Based Refinement |
| FAST | Features from Accelerated Segment Test |
| GPU | Graphics Processing Unit |
| ICL | Iterative Closest Line |
| ICML | Iterative Closest Multiple Lines |
| ICP | Iterative Closest Point |
| KLT | Kanade-Lucas-Tomasi (feature tracker) |
| ME | Mean Error |
| MSE | Mean Squared Error |
| RANSAC | RANdom SAmple Consensus |
| SAC | SAmple Consensus |
| SAD | Sum of Absolute Differences |
| SIFT | Scale-Invariant Feature Transform |
| SLAM | Simultaneous Localization And Mapping |
| SURF | Speeded Up Robust Features |
| WTA | Winner Takes All (matching) |

# Bibliography


Ayache, N. and Faverjon, B., 1987: Efficient registration of stereo images by matching graph descriptions of edge segments. *IJCV*, 1(2), 107–131.

Bachrach, A. G., de Winter, A., He, R., Hemann, G., Prentice, S. and Roy, N., 2010: RANGE-robust autonomous navigation in gps-denied environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1096–1097. IEEE.

Bailey, T. and Durrant-Whyte, H., 2006: Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13(3), 108 – 117.

Baker, H. H., 1982: Depth from Edge and Intensity Based Stereo. Technical Report September, Stanford University.

Bartoli, A. and Sturm, P., 2005: Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. *Computer Vision and Image Understanding*, 100(3), 416–441.

Bay, H., Ferrari, V. and Van Gool, L., 2005: Wide-Baseline Stereo Matching with Line Segments. In *Proc. of CVPR*, 329–336. IEEE.

Besl, P. J. and McKay, N. D., 1992: A method for registration of 3-D shapes. *PAMI*, 14(2), 239–256.

Bosse, M., Newman, P., Leonard, J. and Teller, S., 2004: Simultaneous Localization and Map Building in Large-Scale Cyclic Environments Using the Atlas Framework. *The International Journal of Robotics Research*, 23(12), 1113–1139.

Bouabdallah, S., Noth, A. and Siegwart, R., 2004: PID vs LQ control techniques applied to an indoor micro quadrotor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, 2451–2456. IEEE.

Bradski, G., 2000: The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Canny, J., 1986: A Computational Approach to Edge Detection. *PAMI*, 8(6), 679 – 698.

Chandraker, M., Lim, J. and Kriegman, D., 2009: Moving in stereo: Efficient structure and motion using lines. In *Proc. of ICCV*, 1741–1748. IEEE.

Chen, T. and Wang, Q., 2011: 3D Line Segment Detection for Unorganized Point Clouds from Multi-view Stereo. In *Proc. of ACCV*, 400–411.

Conrad Electronic, 2011: Quadrocopter 450 ARF. `http://www.conrad.de/ce/de/product/208000/QUADROCOPTER-450-ARF-35-MHz/`. [Online, Accessed December 6th 2014].

Davison, A. J., 2003: Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Proc. of ICCV*, 1403–1410. IEEE.

Davison, A. J., Reid, I. D., Molton, N. D. and Stasse, O., 2007: MonoSLAM : Real-Time Single Camera SLAM. *PAMI*, 29(6), 1052–1067.

Dellaert, F., 2012: Factor Graphs and GTSAM : A Hands-on Introduction. Technical Report September.

Denavit, J. and Hartenberg, R. S., 1955: A kinematic notation for lower-pair mechanisms based on matrices. *Trans. of the ASME Journal of Applied Mechanics*, 22, 215–212.

Deriche, R. and Faugeras, O., 1990: 2-D Curve Matching Using High Curvature Points: Application to Stereo Vision. In *Proc. of ICPR*, 240–242.

Devernay, F., 1995: A Non-Maxima Suppression Method for Edge Detection with Sub-Pixel Accuracy. Technical report, INRIA.

Dijkstra, E., 1959: A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269–271.

Dong, Z., Zhang, G., Jia, J. and Bao, H., 2009: Keyframe-Based Real-Time Camera Tracking. In *Proc. of ICCV*, 1538–1545.

Douglas, D. H. and Peucker, T. K., 1973: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica*, 10(2), 112–122.

Eade, E. and Drummond, T., 2009: Edge landmarks in monocular slam. *Image and Vision Computing*, 27(5), 588–596.

Elder, J. H., 1999: Are Edges Incomplete ? *IJCV*, 34(2/3), 97–122.

Fan, B., Wu, F. and Hu, Z., 2010: Line matching leveraged by point correspondences. In *Proc. of CVPR*, 390–397. IEEE.

Fischler, M. and Bolles, R., 1981: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 381–395.

Fitzgibbon, A. and Zisserman, A., 1998: Automatic camera recovery for closed or open image sequences. In *Proc. of the ECCV*, volume 1, 311–326. IEEE.

Flint, A., Mei, C., Reid, I. and Murray, D., 2010: Growing semantically meaningful models for visual SLAM. In *Proc. of CVPR*, 467–474. IEEE.

Furukawa, Y., Curless, B., Seitz, S. and Szeliski, R., 2009: Manhattan-world stereo. In *Proc. of CVPR*, 1422–1429. IEEE.

Furukawa, Y. and Ponce, J., 2010: Accurate, dense, and robust multiview stereopsis. *PAMI*, 32(8), 1362–1376.

Gallup, D., Frahm, J. and Pollefeys, M., 2010: Piecewise Planar and Non-Planar Stereo for Urban Scene Reconstruction. In *Proc. of CVPR*, 1418–1425.

Gallup, D., Frahm, J.-M., Mordohai, P., Yang, Q. and Pollefeys, M., 2007: Real-time plane-sweeping stereo with multiple sweeping directions. In *Proc. of CVPR*, 1–8.

Geiger, A., Lenz, P. and Urtasun, R., 2012: Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proc. of CVPR*, 3354–3361. IEEE.

Geiger, A., Ziegler, J. and Stiller, C., 2011: StereoScan : Dense 3d Reconstruction in Real-time. In *IEEE Intelligent Vehicles Symposium (IV)*, 963–968. IEEE.

Gerum, P., 2004: Xenomai - Implementing a RTOS emulation framework on GNU/Linux. `http://www.xenomai.org/documentation/xenomai-2.3/pdf/xenomai.pdf`. [Online, Accessed December 6th 2014].

Gong, M., Yang, R., Wang, L. and Gong, M., 2007: A performance study on different cost aggregation approaches used in real-time stereo matching. *IJCV*, 75(2), 283–296.

Grzonka, S., Grisetti, G. and Burgard, W., 2009: Towards a navigation system for autonomous indoor flying. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2878–2883. IEEE.

Harris, C. and Stephens, M., 1988: A Combined Corner and Edge Detector. In *Proc. of the Alvey Vision Conference*, 147–152.

Harris, C. G. and Pike, J. M., 1987: 3D Positional Integration from Image Sequences. In *Proc. of the Alvey Vision Conference*, 233–236. Alvey Vision Club.

Hart, P., Nilsson, N. and Raphael, B., 1968: A formal basis for the heuristic determination of minimum cost paths. *TSMC*, 4(2), 100–107.

Hartley, R. I. and Zisserman, A., 2004: *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition.

Helmer, S. and Lowe, D., 2010: Using stereo for object recognition. In *Proc. of ICRA*, 3121–3127. IEEE.

Hirschmüller, H., Innocent, P. R. and Garibaldi, J., 2002: Real-Time Correlation-Based Stereo Vision with Reduced Border Errors. *IJCV*, 47(1), 229–246.

Hofer, M., Wendel, A. and Bischof, H., 2013: Incremental Line-based 3D Reconstruction using Geometric Constraints. In *Proc. of BMVC*, 1–11.

Hoffmann, G., Rajnarayan D.G., Waslander S.L., Dostal, D., Jang, J. S. and Tomlin, C. J., 2004: The Stanford Testbed of Autonomous Rotorcraft for Multi Agent Control (STARMAC). In *The 23rd Digital Avionics Systems Conference*, volume 2. IEEE.

How, J. P., Bethke, B., Frank, A., Dale, D. and Vian, J., 2008: Real-time indoor autonomous vehicle test environment. *Control Systems Magazine, IEEE*, 28(2), 51–64.

Illingworth, J. and Kittler, J., 1988: A Survey of the Hough Transform. *Computer Vision, Graphics, and Image Processing*, 44(1), 87–116.

Jain, A., Kurz, C., Thormählen, T. and Seidel, H.-P., 2010: Exploiting Global Connectivity Constraints for Reconstruction of 3D Line Segments from Images. In *Proc. of CVPR*, 1586–1593. IEEE.

Jeong, W. Y. and Lee, K. M., 2006: Visual SLAM with Line and Corner Features. In *Proc. of IROS*, 2570–2575. IEEE.

Kawai, Y., Ueshiba, T., Ishiyama, Y., Sumi, Y. and Tomitai, F., 1998: Stereo correspondence using segment connectivity. In *Proc. of ICPR*, 648–651. IEEE.

Kim, N. and Bovik, A., 1988: A Contour-Based Stereo Matching Algorithm Using Disparity Continuity. *Pattern Recognition*, 21(5), 505–514.

Klein, G. and Murray, D., 2007: Parallel tracking and mapping for small AR workspaces. In *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 225–234. IEEE.

Klein, G. and Murray, D., 2008: Improving the Agility of Keyframe-Based SLAM. In *Proc. of ECCV*, volume 2, 802–815. IEEE.

Kolmogorov, V. and Zabih, R., 2001: Computing visual correspondence with occlusions using graph cuts. In *Proc. of ICCV*, 508–515. IEEE.

Konolige, K. and Agrawal, M., 2008: FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping. *IEEE Transactions on Robotics*, 24(5), 1066–1077.

Koschan, A. and Rodehorst, V., 1995: Towards real-time stereo employing parallel algorithms for edge-based and dense stereo matching. In *Proc. of IEEE Int. Workshop on Computer Architectures for Machine Perception*, 234–241. IEEE.

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K. and Burgard, W., 2011: G2o: A general framework for graph optimization. In *ICRA*, 1, 3607–3613. University of Freiburg, USA, IEEE.

Li, Z. N., 1994: Stereo correspondence based on line matching in Hough space using dynamic programming. *TSMC*, 24(1), 144–152.

Lindeberg, T., 1998: Edge detection and ridge detection with automatic scale selection. *IJCV*, 30(2), 117–156.

Ljung, L., 1999: *System identification: Theory for the user*. Prentice Hall information and system sciences series. IEEE, Upper Saddle River and NJ, 2. ed. edition.

Lourakis, M. I. A. and Argyros, A. A., 2009: SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software*, 36(1), 1–30.

Lowe, D. G., 1991: Fitting parameterized three-dimensional models to images. *PAMI*, 13(5), 441–450.

Lowe, D. G., 2004: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 91–110.

Mair, E., Hager, G. D., Burschka, D., Suppa, M. and Hirzinger, G., 2010: Adaptive and Generic Corner Detection Based on the Accelerated Segment Test. In *Proc. of ECCV*, 183–196.

Manessis, A., Hilton, A., Palmer, P., McLauchlan, P. and Shen, X., 2000: Reconstruction of Scene Models from Sparse 3D Structure. In *Proc. of CVPR*, volume 2, 2666–2673. IEEE.

Marr, D. and Hildreth, E., 1980: Theory of Edge Detection. *Proceedings of the Royal Society B: Biological Sciences*, 207(1167), 187–217.

Matas, J., Galambos, C. and Kittler, J., 1998: Progressive Probabilistic Hough Transform. In *Proc. of BMVC*, 1–10.

Matas, J., Galambos, C. and Kittler, J., 2000: Robust Detection of Lines Using the Progressive Probabilistic Hough Transform. *CVIU*, 78(1), 119–137.

McLauchlan, P., Shen, X., Manessis, A., Palmer, P. and Hilton, A., 2000: Surface-Based Structure-from-Motion using Feature Groupings. In *Proc. of ACCV*, 699–705. IEEE.

Medioni, G. and Nevatia, R., 1985: Segment-based stereo matching. *Computer Vision, Graphics, and Image Processing*, 31(1), 2–18.

Mei, X., Sun, X., Zhou, M., Jiao, S., Wang, H. and Zhang, X., 2011: On building an accurate stereo matching system on graphics hardware. In *ICCV Workshops*, 467–474. IEEE.

Meier, L., Fraundorfer, F. and Pollefeys, M., 2011: The intelligent flying eye. *SPIE Newsroom*.

Meltzer, J. and Soatto, S., 2008: Edge Descriptors for Robust Wide-Baseline Correspondence. In *Proc. of CVPR*, 1–8. IEEE.

Michael, N., Mellinger, D., Lindsey, Q. and Kumar, V., 2010: The GRASP Multiple Micro-UAV Testbed. *IEEE Robotics & Automation Magazine*, 17(3), 56–65.

Nedevschi, S., Oniga, F., Danescu, R., Graf, T. and Schmidt, R., 2006: Increased accuracy stereo approach for 3D lane detection. In *IEEE Intelligent Vehicles Symposium 2006*, 42–49. IEEE.

Newcombe, R. A., Lovegrove, S. J. and Davison, A. J., 2011: DTAM: Dense Tracking and Mapping in Real-Time. In *Proc. of ICCV*. IEEE.

Nistér, D., Naroditsky, O. and Bergen, J., 2004: Visual odometry. In *Proc. of CVPR*, volume 1, 1–8. IEEE.

Ohta, Y. and Kanade, T., 1985: Stereo by Intra-and Inter-Scanline Search Using Dynamic Programming. *PAMI*, 7(2), 139–154.

Pilz, U., Gropengieß er, W., Walder, F., Witt, J. and Werner, H., 2011: Quadrocopter Localization Using RTK-GPS and Vision-Based Trajectory Tracking. In *Proc. of ICIRA*, 12–21.

Pounds, P., Mahony, R. and Corke, P., 2006: Modelling and Control of a Quad-Rotor Robot. In *Proceedings of the Australasian Conference on Robotics and Automation*. IEEE.

Robert, L. and Faugeras, O. D., 1991: Curve-based stereo: Figural continuity and curvature. In *Proc. of CVPR*, 57–62. IEEE.

Roberts, K., 1988: A new Representation for a Line. In *Proc. of CVPR*, 635–640. IEEE.

Rosten, E. and Drummond, T., 2006: Machine learning for high-speed corner detection. In *Proc. of ECCV*, 430–443.

Scharstein, D. and Szeliski, R., 2002: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1), 7–42.

Schindler, G., Krishnamurthy, P. and Dellaert, F., 2006: Line-Based Structure from Motion for Urban Environments. In *Third International Symposium on 3D Data Processing Visualization and Transmission*, 846–853. IEEE.

Schmid, C. and Zisserman, A., 1997: Automatic line matching across views. In *Proc. of CVPR*, 666–671. IEEE.

Schmid, C. and Zisserman, A., 2000: The geometry and matching of lines and curves over multiple views. *IJCV*, 40(3), 199–233.

Shi, J. and Tomasi, C., 1994: Good features to track. In *Proc. of CVPR*, 593–600. IEEE.

Sibley, G., Mei, C., Reid, I. and Newman, P., 2010: Vast-scale Outdoor Navigation Using Adaptive Relative Bundle Adjustment. *International Journal of Robotics Research*, 29(8), 958–980.

Sinha, S. N., Steedly, D. and Szeliski, R., 2009: Piecewise planar stereo for image-based rendering. In *Proc. of ICCV*, 1881–1888. IEEE.

Smith, P., Reid, I. and Davison, A., 2006: Real-time monocular SLAM with straight lines. In *Proc. of BMVC*, volume 1, 17–26. IEEE.

Strasdat, H., Montiel, J. M. M. and Davison, A. J., 2010: Real-time monocular SLAM: Why filter? In *Proc. of ICRA*, 2657–2664. IEEE.

Sumi, Y., Kawai, Y., Yoshimi, T. and Tomita, F., 2002: 3D Object Recognition in Cluttered Environments by Segment-Based Stereo Vision. *International Journal of Computer Vision*, 46(1), 5–23.

Taylor, C. J. and Kriegman, D. J., 1995: Structure and motion from line segments in multiple images. *PAMI*, 17(11), 1021–1032.

Tomono, M., 2009a: Detailed 3D mapping based on image edge-point ICP and recovery from registration failure. In *Proc. of IROS*, 1164–1169. IEEE.

Tomono, M., 2009b: Robust 3D SLAM with a stereo camera based on an edge-point ICP algorithm. In *Proc. of ICRA*, 4306–4311. IEEE.

Tomono, M., 2010: 3D localization based on visual odometry and landmark recognition using image edge points. In *Proc. of IROS*, 5953–5959. IEEE.

Tomono, M., 2012: Image-based planar reconstruction for dense robotic mapping. In *Proc. of ICRA*, 3005–3012. IEEE.

Triggs, B., McLauchlan, P., Hartley, R. and Fitzgibbon, A., 2000: Bundle adjustmenta modern synthesis. In B. Triggs (Editor), *Vision algorithms*, volume 1883 of *Lecture notes in computer science*, 298–372. IEEE, Berlin.

Ulusoy, I., Halici, U. and Hancock, E., 2004: Probabilistic phase based sparse stereo. In *Proc. of ICPR*, 84–87 Vol.4. IEEE.

Von der Heydt, R., Zhou, H. and Friedman, H. S., 2000: Representation of stereoscopic edges in monkey visual cortex. *Vision research*, 40(15), 1955–1967.

Wang, L., Neumann, U. and You, S., 2009: Wide-baseline image matching using Line Signatures. In *Proc. of ICCV*, 1311–1318. IEEE.

Werner, T. and Zisserman, A., 2002: New Techniques for Automated Architectural Reconstruction from Photographs. In *Proc. of ECCV*, 541–555.

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I. and Tardós, J., 2009: A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous Systems*, 57(12), 1188–1197.

Witt, J., Annighöfer, B., Falkenberg, O., Pilz, U., Weltin, U., Werner, H. and Thielecke, F., 2011a: TUHH Quadrokopter Projekt. `http://www.tu-harburg.de/quadrokopter`. [Online, Accessed December 6th 2014].

Witt, J., Annighöfer, B., Falkenberg, O. and Weltin, U., 2011b: Design of a High Performance Quad-Rotor Robot Based on a Layered Real-Time System Architecture. In *Proc. of ICIRA*, 312–323.

Witt, J. and Mentges, G., 2014: Maximally Informative Surface Reconstruction from Lines. In *Proc. of ICRA*, 2029–2036. IEEE.

Witt, J. and Weltin, U., 2012a: Robust Real-Time Stereo Edge Matching by Confidence-based Refinement. In *Proc. of ICIRA*, 512–522. Springer.

Witt, J. and Weltin, U., 2012b: Sparse Stereo by Edge-Based Search Using Dynamic Programming. In *Proc. of ICPR*, 3631–3635. IEEE.

Witt, J. and Weltin, U., 2013: Robust Stereo Visual Odometry Using Iterative Closest Multiple Lines. In *Proc. of IROS*, 4164–4171. IEEE.

Zeki, S., Watson, J. D. G., Lueck, C. J., Friston, K. J., Kennard, C. and Frackowiak, R. S. J., 1991: A Direct Demonstration of Functional Specialization in Human Visual Cortex. *The Journal of Neuroscience*, 11(3), 641–649.

Zhou, H., Zou, D., Pei, L., Ying, R., Liu, P. and Yu, W., 2015: StructSLAM : Visual SLAM with Building Structure Lines. *IEEE Transactions on Vehicular Technology*, 1–12.

# PUBLICATIONS

J. Witt and G. Mentges, "Maximally Informative Surface Reconstruction from Lines", *IEEE International Conference on Intelligent Robots and Applications* (**ICRA**), 2014

J. Witt and U. Weltin, "Robust Stereo Visual Odometry Using Iterative Closest Multiple Lines", *IEEE/RSJ International Conference on Intelligent Robots and Systems* (**IROS**), 2013

J. Witt and U. Weltin, "Sparse Stereo by Edge-Based Search Using Dynamic Programming", *International Conference on Pattern Recognition* (**ICPR**), 2012

J. Witt and U. Weltin, "Robust Real-Time Stereo Edge Matching by Confidence-Based Refinement", *International Conference on Intelligent Robotics and Applications* (**ICIRA**), 2012

O. Falkenberg, J. Witt, U. Pilz, U. Weltin, H. Werner, "Model Identification and $\mathcal{H}_\infty$ Attitude Control for Quadrotor MAV's", *International Conference on Intelligent Robotics and Applications* (**ICIRA**), 2012

U. Pilz, W. Gropengießer, F. Walder, J. Witt, H. Werner, "Quadrocopter Localization Using RTK-GPS and Vision-Based Trajectory Tracking", *International Conference on Intelligent Robotics and Applications* (**ICIRA**), 2011

J. Witt, B. Annighöfer, O. Falkenberg, "Design of a High Performance Quad-Rotor Robot Based on a Layered Real-Time System Architecture", *International Conference on Intelligent Robotics and Applications* (**ICIRA**), 2011

J. Witt, H. Werner, "Approximate Model Predictive Control for Nonlinear Multivariable Systems", *Model Predictive Control*, InTech, 2010

J. Witt, M. Dunbabin, "Go with the Flow: Optimal AUV Path Planning in Coastal Environments", *Australasian Conference on Robotics and Automation* (**ACRA**), 2008

J. Witt, S. Boonto, H. Werner, "Approximate Model Predictive Control of a 3-DOF Helicopter", *IEEE Conference on Decision and Control* (**CDC**), 2007