Structure from Motion with Higher-level Environment Representations

Zhirui Wang

A thesis submitted for the degree of
Master of Philosophy of
The Australian National University

September, 2018

Declaration

This thesis is an account of research undertaken between March 2016 and September 2018 at The College of Engineering & Computer Science, The Australian National University, Canberra, Australia.

Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

Zhirui Wang
September, 2018

Acknowledgements

This thesis is about the work I did in the last two and a half years. I really appreciate for what I have learned during the period, it is a valuable experience for studying abroad. During the journey, I learned the way about thinking in research and understand the meaning of my work. As an international student, the teaching style is far different from my hometown but it can give the student a wider mind while also making a more creative thought. Thanks to my supervisor Laurent Kneip who guided me all the way through my research. Moreover than that, he's personality also has a positive influence of me in life. For me, he is not only a teacher but also a good friend.

To my beloved father and mother, without whom I'll never be able to finish my degree. Thanks to my mom Ying Xiong for her love and support. Thanks to my dad Yonghui Wang who is my guidance in life.

Abstract

Computer vision is an important area focusing on understanding, extracting and using the information from vision-based sensor. It has many applications such as vision-based 3D reconstruction, simultaneous localization and mapping(SLAM) and data-driven understanding of the real world. Vision is a fundamental sensing modality in many different fields of application.

While the traditional structure from motion mostly uses sparse point-based feature, this thesis aims to explore the possibility of using higher order feature representation. It starts with a joint work which uses straight line for feature representation and performs bundle adjustment with straight line parameterization. Then, we further try an even higher order representation where we use Bezier spline for parameterization. We start with a simple case where all contours are lying on the plane and uses Bezier splines to parametrize the curves in the background and optimize on both camera position and Bezier splines. For application, we present a complete end-to-end pipeline which produces meaningful dense 3D models from natural data of a 3D object: the target object is placed on a structured but unknown planar background that is modeled with splines. The data is captured using only a hand-held monocular camera. However, this application is limited to a planar scenario and we manage to push the parameterizations into real 3D. Following the potential of this idea, we introduce a more flexible higher-order extension of points that provide a general model for structural edges in the environment, no matter if straight or curved. Our model relies on linked Bézier curves, the geometric intuition of which proves great benefits during parameter initialization and regularization. We present the first fully automatic pipeline that is able to generate spline-based representations without any human supervision. Besides a full graphical formulation of the problem, we introduce both geometric and photometric cues as well as higher-level concepts such overall curve visibility and viewing angle restrictions to automatically manage the correspondences in the graph. Results prove that curve-based structure from motion with splines is able to outperform state-of-the-art sparse feature-based methods, as well as to model curved edges in the environment.

Contents

List of Figures

Chapter 1

Introduction

During evolution, almost all creatures living in a light rich environment have developed eyes. This truth proves eyes are highly efficient information collection sensor and vision can provide a rich source of information on the surrounding environment. For humans, one can barely move in a complex environment with eyes closed. Similarly, for machines, computer vision is a very important area where we investigate automatic environment understanding. Vision based simultaneous localization and mapping is a popular area under computer vision where it does environment mapping together with localization. Using a camera can provide a rich source of information that the geometric models hereby established can be formed in several ways minimizing either geometric or photometric errors. Meanwhile, due to the development of the smartphone, the cost of cameras has becomes negligibly small, especially compared to the significantly more expensive lidar sensors. All these advantages raise the interest of the industry for replacing the original high-cost laser based SLAM solution with visual SLAM.

While the common environment representation in structure from motion is mostly given by a sparse point cloud, few of them investigate higher order representation like straight lines. This thesis aims at exploring the possibility of using higher order spline for both environment representation and localization. Compared with sparse point cloud, edges provide more data and thus must lead to higher accuracy. A common way to exploit the potential of edges while still enabling comfortable matching of features and exploitation of incidence relations is given by relying on straight lines. Many works have proven the feasibility of purely line-based structure from motion [2], or even hybrid architectures that rely simultaneously on points and lines [3, 4, 5]. The latter, in particular, have successfully demonstrated superior accuracy in comparison to purely point-based representations. The problem with lines is that they do not represent a general model for describing the 3D location of imaged edges; They are limited to specific, primarily man-made environments in which straight lines are abundant, either in the form of occlusion boundaries or in the form of appearance boundaries in the texture.

More general approaches in which 3D information for curved edges is recovered have recently been demonstrated in the online, incremental visual localization and mapping community. Works such as [6] and [7] notably reconstruct semi-dense depth maps for all image edges. The depth estimates are updated and propagated from frame to frame. While certainly very successful in terms of an economic generation of enhanced map information, the representation is only local and therefore highly redundant in nature. A unique, global representation of the environment optimized jointly over all observations is not provided. Ignoring works that only focus on small scale object reconstruction [8], the same accounts for fully dense approaches that estimate depth over the entire image [9].

In this thesis, we are targeting towards a higher order representation of the environment

for camera tracking. Specifically, we have explored the possibility of using straight lines and Bezier splines in the environment. Compared with sparse point based methods, it is proved that using spline based feature in bundle adjustment can further improve the tracking accuracy as well as giving a more meaningful map.

1.1 Contribution

This thesis contains the work I did in the last two and a half years. During this period, the main contribution focus on the representation of the environment using higher order parameterization. The contribution to the community is listed as below:

- A way of parameterizing straight lines in 3D space and an optimization algorithm that minimizes the re-projection error over multiple view points.
- The use of Bezier splines for modeling the curves in 3D, which provides a combination of advantages in particular during the initialization of the Bezier spline parameters.
- The first complete framework that automatically extracts, matches, initializes, and optimizes a purely curve-based environment representation without any human intervention. The optimization does not involve lifting, and hence remains fast on standard architectures.
- Novel photometric and geometric criteria for verifying correspondences. In particular, our method uses color images, and the photometric error is evaluated in the HSV space. The quality of the correspondences is further reinforced by checking the consistency of edge identities and viewing directions.
- We present a full pipeline for 3D object model reconstruction with calibrated hand-held monocular cameras. The pipeline is composed of a free-form curve based structure-from-motion module coupled to a user-assisted graph cut implementation for object segmentation and a space-carving back-end for providing simple reconstructions from the obtained camera poses.

1.2 Papers

1.2.1 Published

Wang, Z., Kneip, L., Towards Space Carving with a Hand-Held Camera, International Conference on Computer Vision Systems, 2017

1.2.2 Under Review

Wang, Z., Kneip, L., Fully automatic structure from motion with a spline-based environment representation, Asian Conference on Computer Vision, 2018

Yu, P., Wang, C., Wang Z., Yu J., and Kneip, L., Line-Based Relative Pose for Light-Field Cameras, Robotics and Automation Letters (RAL) with ICRA presentation option

1.3 Thesis Outline

This thesis is divided into six chapters, chapter 2 gives some background information about this thesis as well as the motivation for my research. Chapter 3 describes a fast implementation of line-based bundle adjustment, which was developed in the context of a joint collaboration on line-based visual SLAM with light-field cameras, and provides a natural starting point for the main contributions in this thesis. I mainly participate in the optimization of the re-projection and my work is demonstrated in the chapter. Chapter 4 the main contribution of this work, which is spline-based environment modeling and localization for both the planar and the completely general 3D case. Chapter 5, concludes with an application of the bezier spline based parameterization where we do a 3D reconstruction of a small object using the estimated poses. Finally, chapter 6 gives a conclusion of the thesis.

- **Chapter 2** This chapter gives some background information for structure from motion and demonstrate some state-of-art pipelines. In the end of the chapter, I explain the motivation for the present, higher-order model based representations.

- **Chapter 3** This chapter explains the straight line based structure from motion pipeline I developed in the context of a light-field camera based localization and mapping platform. I'll demonstrate how do we parameterize the straight line as well as the optimization using straight line feature.

- **Chapter 4** As a combination of two works, this chapter is divided into two major sections while sharing some common information at the beginning. I will explain the work I did in both of my papers as well as the results.

- **Chapter 5** This chapter shows a application of my works using planar based bezier spline parameterization. We do a space carving based 3D reconstruction where the input poses is obtained using a structure from motion pipeline with planar spline based bundle adjustment.

- **Chapter 6** gives a conclusion of all my works during my master degree. It discuss the contribution and limitations of my work in that period while also list some future works that can be address for following work.

Chapter 2

Background & Motivation

This chapter will give a brief introduction of the basic in structure from motion as well as demonstrate the motivation of researching in this direction.

2.1 Structure From Motion

Structure from motion(SFM) is a photogrammetric range imaging technique under computer vision which aims at recovering 3D information from Multiple 2D images. The main target is usually divided into two parts: one is to reconstruct the 3D model of the scene while another one is the recovery of the motion of the camera position. Reconstructing a 3D model from multiple planar projections is a classical *inverse problem* with a long-standing history in computer vision. The solution typically involves three steps. The first one is given by the extraction of stable features from the images, the second one by the establishment of correspondences between features in different images, and the third one by the exploitation of incidence relations that permit the recovery of camera poses and 3D structure. We denote a stable feature with points in the image that can be reliably extracted from different viewpoints while always pointing at the exact same point in 3D. Ignoring complex cases such as occlusions and apparent contours, these are all the image points for which a local extremum in the first order derivative can be observed. This—and the intuition behind line drawings—have to lead to the initial belief that the most useful features in an image are simply all the edges. However, later research has shown that point correspondence are not only much easier to establish but also easier to be used as part of incidence relations from which even closed-form solutions to camera resectioning and direct relative orientation can be derived [10, 11]. Point-based paradigms, therefore, are the dominating solution to the structure from motion problem. In this section, we will explain the model we use for the camera as well as giving a background theory of the Structure from motion technique.

2.1.1 Camera Model

We use the pinhole model to describe the projection of a point into the camera. In the pinhole camera model, the camera is described as a box with a small hole in the front side. Each point in the object is projected into the image based on the principle of collinearity, where the points are projected by a straight line through the camera center (the hole in the front side). This model can be described with a coordinate of (O, x, y, z) where O represents the camera center. Parameter x and y form a plane that is parallel to with the image plane while the z-axis is perpendicular to the image plane.

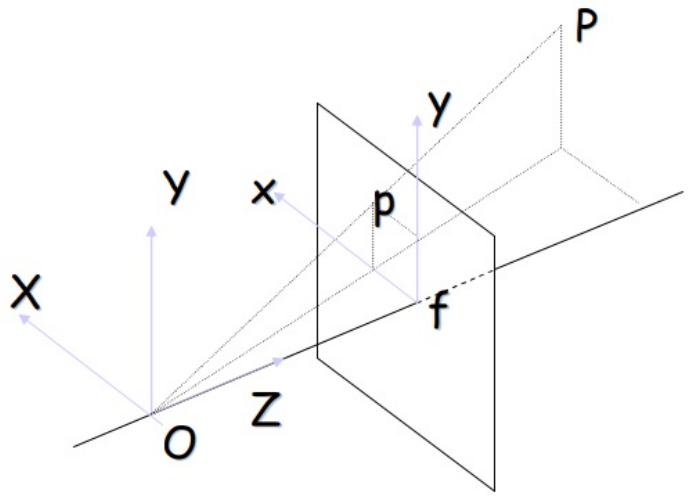


Figure 2.1: Example of a pinhole model

Intrinsic parameters are used to describe the perspective projection inside the camera. In order to get the relationship between a point in camera coordinate $\mathbf{P_c}(x, y, z)^T$ and the corresponding point on camera retina $\mathbf{p}(x, y, 1)^T$, an intermediate plane is introduced. It is parallel to the cameraś retina with a unit distance. The points on the normalized plane are described in homogeneous coordinate as $\hat{p}(u, v)^T$ and the corresponding homogenous projection is described as

$$\hat{p} = \frac{1}{z} P_c \tag{2.1}$$

and the point on the image plane is describe as :

$$\mathbf{p} = \mathbf{K}\hat{\mathbf{p}} \tag{2.2}$$

Where the matrix $\mathbf{K}$ is described as the camera matrix:

$$\mathbf{K} = \begin{bmatrix} \mathbf{f_x} & alpha_c \mathbf{f_x} & \mathbf{c_x} \\ 0 & \mathbf{f_y} & \mathbf{c_y} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

The vector **fc** is the focal length in pixel unit while the vector **cc** represent the principal point coordinates in pixel unit. $alpha_c$ is the skew coefficient defining the angle between the x and y pixel axes stored in the scalar.

2.1.2 Camera tracking and Mapping

The orientation and position of the camera is represented as a 3×3 rotation matrix $\mathbf{R}$ and a 3×1 translation matrix $\mathbf{t}$ while $[\mathbf{R}|\mathbf{t}]$ describes the extrinsic camera parameter matrix. The transformation of a point $\mathbf{P}$ from Camera 1 to Camera 2 can be defined as:

$$\mathbf{P}' = \mathbf{R_1^2}\mathbf{P} + \mathbf{t_1^2} \rightarrow \hat{\mathbf{P}}' = \begin{bmatrix} \mathbf{R_1^2} & \mathbf{t_1^2} \\ 0 & 1 \end{bmatrix} \hat{\mathbf{P}} \tag{2.4}$$

where $\hat{\mathbf{P}}$ is the point in homogeneous coordinate and $\mathbf{P}^{'}$, $\hat{\mathbf{P}}^{'}$ are the projected position respectively.

In order to derive the transformation matrix, the first step is to match correspondences between images. As previously described, traditional methods extract sparse features as key-points and do feature matching between key-points sets to obtain correspondences. There are many types of features can be used here like ORB[12], SURF[13], BRISK[14]...Each type of feature will derive a description of the feature and the matching can be done by comparing the similarity between the descriptors. To obtain the geometric relation between two images directly without the need for retrieving the depth of points, the fundamental matrix is introduced. The fundamental is a 3×3 matrix where

$$\mathbf{0} = \mathbf{x}^{'}\mathbf{F}\mathbf{x} \tag{2.5}$$

$\mathbf{x}$ is a point in one image while $\mathbf{x}^{'}$ is the correspondence point in another image. We further introduce the essential matrix where:

$$\mathbf{E} = \mathbf{K^{-1}}\mathbf{F}\mathbf{K} \tag{2.6}$$

And the essential matrix $\mathbf{E}$ can be decomposed into $\mathbf{R}, \mathbf{t}$ as described in [15]. However, for monocular cameras, the translation only determines the direction of the real translation and the scale is not observable.

For a sparse feature based algorithm, the mapping is usually done by generating a point cloud of the feature points. In order to obtain the 3D position from a 2D observation, another observation is required and we do triangulation to estimate the depth of the point. Assume the position of a point $\mathbf{P}$ in the coordinate in the first camera, and the transformation from the first camera to the second camera is $[\mathbf{R}|\mathbf{t}]$.The observation of $\mathbf{P}$ in both camera is $\mathbf{p_1}$ and $\mathbf{p_2}$. The position of the point $\mathbf{P}$ can be easily obtained by solving the equation:

$$\Lambda_2\mathbf{K^{-1}}\mathbf{p_2} = \Lambda_1 \begin{bmatrix} \mathbf{R_1^2} & \mathbf{t_1^2} \\ 0 & 1 \end{bmatrix} \mathbf{K^{-1}}\mathbf{p_1} \tag{2.7}$$

where Λ_1 and Λ_2 are the scaling factor of the point observation $\mathbf{p_1}$ and $\mathbf{p_2}$. The equation can be reorganized in to the format of $\mathbf{A}\mathbf{x} = \mathbf{0}$ and solved using SVD decomposition as shown in [15]. After triangulation, the 3D position of the points and the camera position is usually further optimized using bundle adjustment. Bundle adjustment is a non-linear refinement algorithm which minimizes the overall cost of the errors. For most sparse feature based method the cost function is derived as the re-projection error of the 3D points into the images. The re-projection of a 3D point i into a camera j can be derived as:

$$\mathbf{p(i,j)}^{'} = \mathbf{K} \begin{bmatrix} \mathbf{R_j} & \mathbf{t_j} \\ 0 & 1 \end{bmatrix} \mathbf{P_i} \tag{2.8}$$

As the observation of point i in camera j is $\mathbf{m_{ij}}$ and the image is on the retina plane where $\mathbf{p_0}[3] = 1$ the final reprojection error can be written as

$$\mathbf{E}_{i,j} = || \frac{\mathbf{p(i,j)}^{'}}{||\mathbf{p(i,j)}^{'}||} - \mathbf{m_{ij}} || \tag{2.9}$$

Assume we have $\mathbf{n}$ 3D points in a world coordinate with $\mathbf{m}$ views. For each 3D point

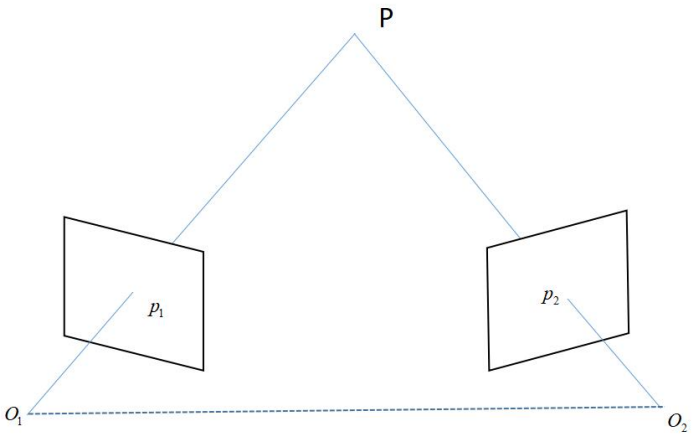


Figure 2.2: Ideal relationship of the triangulation with two camera

$\mathbf{P_i}$ observed in $\mathbf{q_i}$ views where the relative transformation from the world coordinate to the observing camera coordinate j is $\mathbf{T_j}$. The final equation of the bundle adjustment can be written as

$$argc_{min}\{\mathbf{P}_1...\mathbf{P}_n, \mathbf{T}_1...\mathbf{T}_n\} = \sum_{i=0}^{i<n} \sum_{j=0}^{j<q_i} \mathbf{E}_{i,j} \tag{2.10}$$

2.2 State-of-Art

Sparse feature based structure from motion is a very popular direction in research and there are many successful systems in the literature. This section gives some brief introduction of some start-of art structure from motion pipelines.

ORB-SLAM[16] is an open source sparse feature based monocular SLAM pipeline. It uses ORB features and runs in real time. It provides a mostly stable result in small scale, large scale indoor and outdoor environments. It is the first real-time visual SLAM pipeline that works in large scale environments since also executing active loop closure. More specifically, the framework uses the same type of features for local invariant keypoint tracking and matching, as well as large scale place recognition. It proves that bundle adjustment over a strong network of correspondence with reasonably good initialization can lead to highly accurate pose estimation results in real-time. Besides that, ORB SLAM has a robust strategy to maintain the tracking. It balances accuracy and efficiency by maintaining a local map which reduces the size of the bundle adjustment while still providing sufficient information for stable local tracking. Even when tracking is lost, it maintaining a re-localization algorithm which tries to re-localize the camera by comparing to previous views using a bag of word matching. However, there are some disadvantages of using ORB features, as they are fast but also cheap. Compared against other features like SURF, ORB can be very unstable and easily lead to a tracking lost. Also, as they directly use nonlinear refinement to recover poses, the initial position must be close to its real position to avoid local minima. To reduce the influence of this problem, they assume relative translation between subsequent frames to be constant and predict the position for the next frame using the previous transformation. However, this can easily break for a highly dynamic camera and lead to a tracking lost.

Bundler [17][18] is another open source library for Structure from motion. Different from ORB-SLAM, bundler doesn't require continuity of the input which means the input does not necessarily capture by the same camera in a sequence. This gives the advantage that bundler can use any image found online for the reconstruction. For each image, it computes the 3D camera position and registers it to the map. With the given transformation between cameras, the 3D position of the image point can be recovered. In bundler, the scene is reconstructed with points and lines in the image. Bundler uses SIFT features which means it can't run in real-time and it can take a lot of time depending on the size of the dataset.

2.3 Motivation

Curve-based geometric incidence relations have since ever intrigued the structure-from-motion community. For example, rather than solving the relative pose problem from points correspondences [10, 11], early works such as [19] and Lateron [20] and [21] looked into the possibility of using curves and surface tangents to solve the stereo calibration problem. However, the presented constraints for solving geometric computer vision problems are easily influenced by noise, and not practically useful. In order to improve the quality of curve-based structure from motion, further works, therefore, looked at special types of curves such as straight lines and cones, respectively [22, 23].

Our primary interest is the solution of structure-from-motion over many frames and observations. Point-based solutions are very mature from both theoretical [15] and practical perspectives [24]. However, point-based representations are somewhat unsatisfying as they simply do not present a complete, visually appealing result. It is therefore natural that the structure-from-motion community has been striving for higher-level formulations that are able to return fully dense surface estimates [8]. Fully dense estimation is however very computationally demanding, which is why a compromise in the form of line based representations has also received significant attention from the community [2, 25]. Lines are however not a general model for representing the environment, they fail in environments where the majority of edges are bent. [26, 27, 28, 29] provide a solution to this problem by introducing curve-based representations of the environment, such as sub-division curves, non-rational B-splines, and implicit representations via 3D probability distributions. However, they do not exploit the edge measurements to improve on the quality of the pose estimations as well, as they do not optimize the curves and poses in a joint optimization process.

Full bundle adjustment over general curve models and camera poses has first been shown in [30]. The approach, however, suffers from a bias that occurs when the model is only partially observed. [31] discusses this problem in detail, and presents a lifting approach that transparently handles missing data. [32] solves the problem by modeling curves as a set of shorter line segments, and [33] models the occlusions explicitly. While [31] is the most related to our approach, the lifted formulation is computationally demanding, the work does not discuss the fully automatic establishment of a correspondence graph that would enable fully automatic incremental structure-from-motion. Our main contribution is an efficient, fully automatic solution to this problem, thus enabling automatic curve-based structure from motion in larger scale environments.

Further related work can be found in the online visual SLAM community, which equally aimed at finding an efficient, general compromise between point-based [34, 16, 35] and dense [9] formulations. While [3, 4, 5] have again looked at using lines (primarily through

a combination with points), recent works such as [36, 6, 37, 7, 38, 39] have also successfully realized visual SLAM pipelines based on general edge features by estimating depth along all edges that can identified in the images (e.g. by applying Canny-edge extraction [40]). While these methods do represent fully automatic pipelines, their results do rely on a global representation of curves, and consequently, fail to jointly optimize over poses and structure.

Chapter 3

Line based Structure from Motion

While most monocular visual SLAM frameworks rely on sparse key-points, it has long been acknowledged that lines also represent a higher-order feature with high accuracy and has been proven that they have good repeatability and abundant availability in man-made environments. However, the back-projection of a 2D line measurement is a plane that intersects with the camera center. For any 2D-2D line correspondence between two views, these planes will generally intersect in a 3D line, independently of the relative camera pose. That means a correspondence between two points of view will have no constraint to the camera poses which means at least three views are required to recover the geometric relationship between cameras and this can then be recovered using trifocal tensor geometry [41]. The present chapter also introduces line-based bundle adjustment, the next logical extension to the point-based case.As indicated in [42], using a single camera is most likely insufficient for doing purely line based motion estimation which ,in the end, make us to build a light-field camera for experiment. The goal here essentially consists of real-time visual simultaneous localization and mapping (VSLAM) using line based features and only a system overview is shown in figure 3.1. Instead of using a monocular camera, we use a light-field camera with nine cameras where each of the cameras is calibrated and the relative poses between the cameras is fixed. This chapter is organized as follow: section 3.1 gives a brief introduction of the trifocal tensor geometry which can be used to bootstrap structure-from-motion problems based on lines. Then, in section 3.2, we will demonstrate our back-end line-based bundle adjustment.

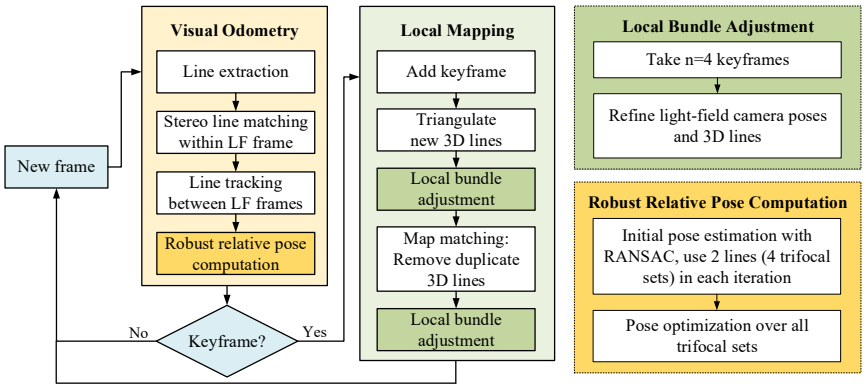


Figure 3.1: The system is originally designed for light-field cameras. Here only provide a brief overview of tri-focal tensor based bootstrapping in the front-end and the representation of lines and line-based bundle adjustment in the back-end

3.1 Short review of trifocal tensor geometry

A trifocal tensor describe the geometry relationship of a straight line observed in three views[15]. Suppose a line $\mathbf{L}$ in 3D space is imaged as the corresponding triplet $\mathbf{m} \leftrightarrow \mathbf{m'} \leftrightarrow \mathbf{m''}$ across three perspective views, which are indicated in Figure 3.1 by their centers $\mathbf{C}$, $\mathbf{C'}$, and $\mathbf{C''}$. Trifocal tensor geometry describes the existence of three 3×3 matrices $\mathbf{S_i}$ (or alternatively a cubic tensor of size 3) that permit the reconstruction of the three coordinates of the line measurement in the first view, respectively. Calling the result q, this line transfer is given by

$$\mathbf{q}^T = \mathbf{m}^{'T}[\mathbf{S_1}, \mathbf{S_2}, \mathbf{S_3}]\mathbf{m}^{''} \tag{3.1}$$

Since $\mathbf{q}$ ideally has to be proportional to the measured line in the first view $\mathbf{m}$ we can easily derive the trifocal tensor incidence relation

$$\mathbf{m}^{'\mathbf{T}}[\mathbf{S_1}, \mathbf{S_2}, \mathbf{S_3}]\mathbf{m}^{''}[\mathbf{m}]_\times = \mathbf{0}^T \tag{3.2}$$

The trifocal tensor can be described as a function of the three camera matrices $\mathbf{P}, \mathbf{P}^{'}$, and $\mathbf{P}^{''}$. Before proceeding, we first move to normalised coordinates given that we assume to be in the calibrated case.We start with normalising the line measurements $\mathbf{m}$ that satisfy the equation $\mathbf{x}^T\mathbf{m} = 0$ for any image point $\mathbf{x}$ along the line. Defining $\mathbf{K}$ to be the matrix of intrinsic camera parameters, the equation can be easily rewritten as

$$\mathbf{x}^T\mathbf{m} = 0 \Rightarrow \mathbf{x}^T\mathbf{K}^{-T}\mathbf{K}^T\mathbf{m} = 0 \rightarrow (\mathbf{K}^{-1}\mathbf{x})^T(\mathbf{K}^T\mathbf{m}) = 0 \tag{3.3}$$

The normalised line coordinates are therefore given by $\mathbf{l} = \mathbf{K}^\mathbf{T}\mathbf{m}$, and it is easily recognised that they also correspond to the normal vector of the 3D plane spanned by the 3D line L and the camera center $\mathbf{C}$. Choosing the first view to be the reference, the camera matrices in the calibrated case are given by

$$\mathbf{P} = [\mathbf{I}|\mathbf{0}],\ \mathbf{P}^{'} = [\mathbf{R}^{'}|\mathbf{t}^{'}],\ \mathbf{P}^{''} = [\mathbf{R}^{''}|\mathbf{t}^{''}] \tag{3.4}$$

where $\mathbf{t}^{'}, \mathbf{R}^{'} = [\mathbf{r}_1^{'}\ \mathbf{r}_2^{'}\ \mathbf{r}_3^{'}]$ and $\mathbf{t}^{''}, \mathbf{R}^{''} = [\mathbf{r}_1^{''}\ \mathbf{r}_2^{''}\ \mathbf{r}_3^{''}]$ are the euclidean parameters that permit the transformation of points from the first to the second and the first to the third camera frame. Given its general form taken from [15], the trifocal tensor $[\mathbf{T}_1\ \mathbf{T}_2\ \mathbf{T}_3]$ that satisfies the incidence relation in the calibrated case

$$(\mathbf{l}^{'T}[\mathbf{T}_1,\ \mathbf{T}_2,\ \mathbf{T}_3]\mathbf{l}^{''})[\mathbf{l}]_\times = \mathbf{0}^T \tag{3.5}$$

is finally given by

$$\mathbf{T}_i = \mathbf{r}_i^{'}\mathbf{t}^{''T} - \mathbf{t}^{'}\mathbf{r}_i^{''T} \tag{3.6}$$

Let us assume a line $\mathbf{L}$ has been observed in n cameras where the extrinsic calibration parameters are given by $\mathbf{c}_i$ while the relative pose between any two camera is described as $\left\{\mathbf{t}_i^j, \mathbf{R}_i^j\right\}$, which transform points from the i^{th} view to j^{th} view. The geometry is illustrated in Figure 3.1. Any three distinct views, say $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$, selected from the observations need to satisfy the trifocal tensor incidence relation (3.5). Picking the first view of the trifocal set to be the reference frame, and assuming that the transformation to the second view already known, it is easy to see that the trifocal tensor constraints become linear in $\mathbf{t}^{''}$ and $\mathbf{R}^{''}$. Each line correspondence, therefore, gives us three linear constraints on the

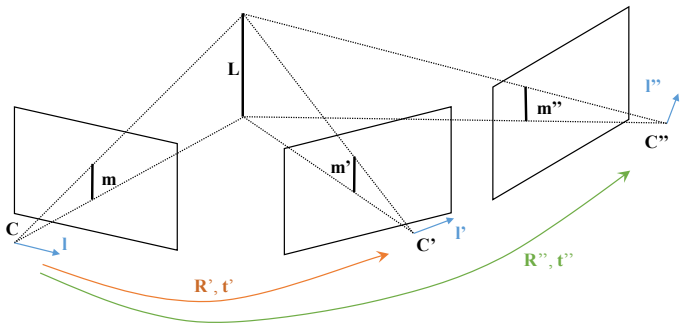


Figure 3.2: Multiple-view geometry of 3D line observations.

relative pose between the third and the first view of the trifocal set. It is however clear that a single line correspondence is not enough to fully constrain the relative pose. We use in total four line correspondences, from which we can obtain 4×3 constraints on the motion parameters. With a total of 12 equations that are linear in the parameters $\mathbf{t}$ and $\mathbf{R}$, we obtain a first solver by a simple linear solution, which can be used for bootstrapping our solution. Note that, in order to ensure that the rotation is a proper rotation matrix, we project the linear solution onto the nearest orthonormal matrix via a simple SVD. The lines can be recovered by extracting the two-dimensional null-space of the normalized line measurements in the image plane.

3.2 Parameterization and back-end optimization

As we already explained the front-end, this section shows how do we parameterize the lines as well as how we optimize them during bundle adjustment. Our local mapping module performs optimization of all variables over a window of n most recent keyframes. This includes the poses of the frames as well as all parameters of 3D lines for which there exist at least two observations inside the window of considered frames. The procedure is commonly known as local or windowed bundle adjustment. After a first execution terminates, we run a map matching procedure in which we seek further correspondences between the current frame and the 3D lines, directly. After all correspondences have been added, we conclude with another round of local bundle adjustment.

An important question when applying bundle adjustment is how exactly to parametrize the optimized variables. Taking the camera pose in the calibrated case as an example, the difficulty lies in making sure that the orientation variable remains a point on the special-orthogonal group $\mathbb{SO}(3)$, either through a minimal, implicit parametrization or by explicitly adding side-constraints to the optimization. In order to remain free of side-constraints, we simply choose the minimal Cayley parameters to represent the orientation of each frame. Furthermore, in order to ensure that we remain far away from 180 rotations for which $||$ [x y z]$^T ||\rightarrow \infty$, we optimize pose changes from the initial absolute pose of each frame rather than the absolute pose directly.

We also need to decide how to parametrize the 3D lines, which are commonly represented by 6 Plücker line coordinates, but only have 4 degrees of freedom. The Plücker line coordinates are notably given by a line direction vector $\mathbf{d}$ with a unit norm and a moment vector $\mathbf{m}$ that is orthogonal to $\mathbf{d}$. As suggested in [15], a minimal parametrization could be given by two points on two planes, which would however leave us with the difficulty of defining these two planes as well as the inability to attain lines that are parallel to one of

the two planes without being contained in it. With strong ties to [2], we propose a less restrictive minimal parametrization of the 3D lines for back-end optimization. Since $\mathbf{d}$ and $\mathbf{m}$ are orthogonal, they can notably be parametrized as the first two columns of a rotation matrix that again is represented as a function of Cayley parameters [43]. The fact that the moment vector $\mathbf{m}$ does not have unit norm is furthermore reflected by multiplying the second column of this matrix by a scale parameter s. We obtain

$$\begin{bmatrix}\mathbf{d}, & \mathbf{m}\end{bmatrix} = \frac{1}{1+x^2+y^2+z^2}\begin{bmatrix}\begin{bmatrix}1+x^2-y^2-z^2\\ 2xy+2z\\ 2xz-2y\end{bmatrix}, & s\begin{bmatrix}2xy-2z\\ 1-x^2+y^2-z^2\\ 2yz+2x\end{bmatrix}\end{bmatrix}. \tag{3.7}$$

We furthermore switch to local optimization by adding a starting orientation $\mathbf{R_0} = [\mathbf{d_0}\frac{\mathbf{m_0}}{||\mathbf{m_0}||}\left(\mathbf{d_0}\times\frac{\mathbf{m_0}}{||\mathbf{m_0}||}\right)]$and a starting line moment $\mathbf{s_0}$ to the parametrization, thus resulting in

$$[\mathbf{d},\mathbf{m}] = \mathbf{R_0}\frac{1}{1+x^2+y^2+z^2}\begin{bmatrix}\begin{bmatrix}1+x^2-y^2-z^2\\ 2xy+2z\\ 2xz-2y\end{bmatrix}, & (\mathbf{s_0}+\mathbf{s})\begin{bmatrix}2xy-2z\\ 1-x^2+y^2-z^2\\ 2yz+2x\end{bmatrix}\end{bmatrix} \tag{3.8}$$

As required, this parametrization has only 4 degrees of freedom, and implicitly enforces all side-constraints on the Plücker line coordinates.

We conclude the exposition of our line-based bundle adjustment back-end by explaining the reprojection error of every 3D line into every frame where a measurement is available. Measurements are notably given by two endpoints $\mathbf{p}_1$ and $\mathbf{p}_2$ while our 3D lines are parametrized as infinitely long. The reprojection error is simply given by the sum of the two squared orthogonal distances between the endpoints and the reprojected line. For simplicity, we sample two points from the lines as $\mathbf{P}_1 = \mathbf{d}\times\mathbf{m}$ and $\mathbf{P}_2 = \mathbf{d}\times\mathbf{m}+\mathbf{d}$ where $\mathbf{d}$ and $\mathbf{m}$ derived from equation 3.8. For a line $\mathbf{L}_i$ observed in the j^{th} camera where the projection from the world coordinate to the camera coordinate derived as π_j, the direction of the projected line can be derived as

$$\mathbf{n_{ij}}^{'} = [n_x, n_y]^T = \mu(\pi_j(\mathbf{p}_2) - \pi_j(\mathbf{p}_1)) \tag{3.9}$$

where $\mu(\mathbf{x})$ return the normalized vector $\mathbf{x}$

$$\mu(\mathbf{x}) = \frac{\mathbf{x}}{||\mathbf{x}||} \tag{3.10}$$

The normal direction of the projected line can be derived as

$$\mathbf{n_{ij}} = [-n_y, n_x]^T \tag{3.11}$$

With a reference point of the projected line can be founded as $\mathbf{C} = \frac{\pi(\mathbf{p}_2)-\pi(\mathbf{p}_1)}{2}$ the cost of the reprojected line becomes

$$e_{ij} = [\mathbf{n_{ij}}^T(\mathbf{p}_1 - \mathbf{C})]^2 + [\mathbf{n_{ij}}^T(\mathbf{p}_2 - \mathbf{C})]^2 \tag{3.12}$$

And the final cost becomes

$$E = \sum_{ij}\mathbf{e}_{ij} \tag{3.13}$$

Note that, in contrast to [3, 44], we employ a genuine, virtually infinite line representation in 3D rather than two end-points. In turn, we utilize end-points where they naturally occur, namely for our measurements extracted in the image plane. This appears to be a more logical definition, as end-points defined in 3D could easily collapse and produce a zero residual.

3.3 Result

As illustrated in [45], even a stereo camera has difficulties in solving the line-based structure from motion problem (depending on the orientation of the baseline). Therefore, we used a light-field camera with a 3x3 matrix arrangement of cameras. Comparing our complete light-field SLAM pipeline on common benchmark datasets is difficult since none of them provides images captured by a plenoptic camera. Fortunately, the realistic synthetic benchmark given by the ICL-NUIM datasets [1] enables us to render novel views as captured by a light-field camera matrix. We define a virtual 3×3 camera matrix with 0.1m baselines while keeping the standard intrinsic parameters unchanged and render new views for two living room sequences for which sufficient lines can be observed. We compare our framework against multiple state-of-the-art solutions from the literature [16, 46, 47, 48, 49, 50]. We evaluate the absolute trajectory error using the tools provided by the TUM RGBD-SLAM benchmark [51] (see reference for detailed definition), and summarize the median values over 50 runs in Table 3.1. Example views of the environment with overlaid line reconstructions as well as the final trajectory are illustrated in Figure 3.4(b). Note that the sequences have no obvious loop, hence the question of whether or not loop closure is performed in the pipeline is irrelevant.

Drift is hardly observable, and the high accuracy is underlined by numbers. We rival even state-of-the-art RGBD-SLAM solutions, and outperform all comparison algorithms on sequence *kt2* by returning a sub-centimeter median absolute trajectory error. Further details about the results obtained by our method including the mean, median, maximum, and minimum errors of both the absolute trajectory error and the relative pose error can be found in Table 3.2. Even if changing the intra-light-field baselines to a different value, sub-centimeter accuracy remains achievable (though we confirm that there is of course only one optimal baseline for a given average scene depth).

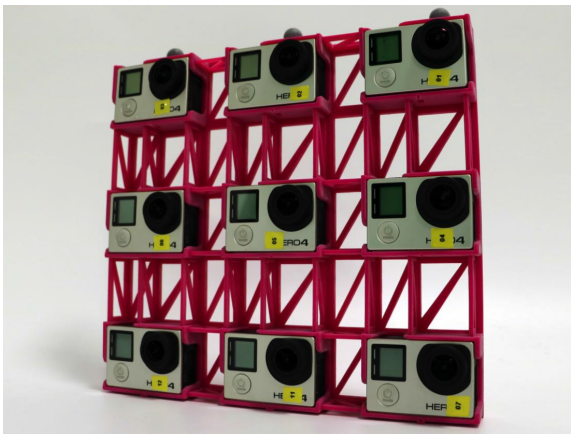


Figure 3.3: The lightfield camera used in this project

Sequence	Ours(0.1m baseline)	ORB-SLAM2 (stereo)[16]	ORB-SLAM2(RGBD)[16]	Elastic Fusion [46]
kt1	1.9	8.6	17.1	0.9
kt2	0.7	6.8	1.9	1.4
Sequence	DVO-SLAM [47]	Endres et al. [48]	MRSMap [49]	Kintinuous [50]
kt1	2.9	0.8	22.8	0.5
kt2	19.1	1.8	18.9	1.0

Table 3.1: Median absolute trajectory errors averaged over 50 runs on ICL-NUIM sequences *kt1* and *kt2*.

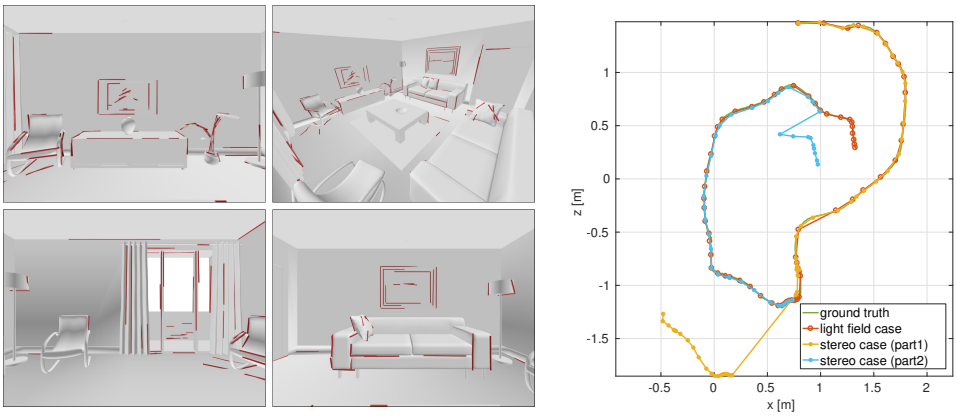


Figure 3.4: Example 3D lines reconstructed by our light-field SLAM pipeline on the synthetic ICL-NUIM living room dataset *kt2*. The right figure shows a ground-floor projection of the estimated trajectory overlaid on groundtruth. The figure furthermore includes results obtained by using only two sub-views of the light-field camera with diagonal baseline. The trajectory splits into two parts because of tracking loss.

	mean	median	max	min
	kt1 (baseline = 0.1m)			
ATE RMSE [cm]	2.27	1.85	7.16	0.92
RPE trans.RMSE [cm]	3.65	3.24	12.39	1.48
RPE rot.RMSE [deg]	1.00	0.87	3.64	0.39
	kt2 (baseline = 0.1m)			
ATE RMSE [cm]	0.85	0.74	3.54	0.36
RPE trans.RMSE [cm]	1.53	1.34	6.03	0.69
RPE rot.RMSE [deg]	0.32	0.30	0.95	0.10
	kt2 (baseline = 0.15m)			
ATE RMSE [cm]	1.78	1.57	5.68	0.51
RPE trans.RMSE [cm]	2.95	2.71	8.21	1.00
RPE rot.RMSE [deg]	0.73	0.66	2.09	0.15

Table 3.2: Detailed results for our method on ICL-NUIM sequences *kt1* and *kt2*.

Chapter 4

Bezier spline based parameterization and bundle adjustment

As already demonstrated the method of using straight lines as a higher order parameterization of the environment in the previous chapter, there are obvious limitation of straight lines: it is impossible to represent curves which are commonly distributed in nature environments. In this chapter, we explore the possibility of using a even higher order representation of the environment where we use Bezier splines to achieve free form curve parameterization. As a combination of the two paper mentioned in Section 1, this chapter is divided into two self-contain sub-chapter where the first one works on a simpler scenario which assumes that contours are placed on a planar scene and the second one focuses on a fully free form contour representation. This chapter is organized as follow: We first give a general background information of Bezier in Section 4.1, Section 4.2 then presents the contribution of using Bézier spline with a planar scene. In the end, Section 4.3.2 demonstrate how our main contribution of the Bézier spline based structure from motion pipeline.

4.1 Bézier splines as a higher-order curve model

Different from the traditional map representation which uses points for map representation, we aim at using Bézier splines and thus reach a more complete, higher order representation of the environment. A Bézier-spline is a continuous curve expression parametrized as a function of control points and a continuous curve parameter t. Every point on the curve can be obtained by referring to a unique value of t. The general definition of a Bézier spline is:

$$\mathbf{B}(t) = \sum_{i=0}^{k} b_{i,k}\mathbf{P}_i, \tag{4.1}$$

where $b_{i,k} = \binom{k}{i} t^i(1-t)^{k-i}$ and $\binom{k}{i} = \frac{k!}{i!(k-i)!}$. $\mathbf{P_i}$ is the $\mathbf{i}^{th}$ control point of the Bézier curve, and k is the order of the curve.

Besides compactness, the choice of Bézier splines is motivated by their simple geometric meaning: the first and last control points are simply the beginning and ending point of the curve while the directions from the first control point to the second and the fourth control point to the third are equal to the local gradient at the beginning and the end

of the curve. As we will see, this clear geometric meaning facilitates initialization and regularization of parameters. Furthermore, Bézier curves parameters are invariant with respect to affine transformations, which means that the transformation of a Bézier curve between different coordinate frames is done by simply applying the same transformation to its control points. Furthermore, Bézier splines provide implicit smoothness and a scale invariant density of points along the curve by simply adjusting the step-size of t.

4.1.1 Smooth polybéziers

In our work, we use cubic Bézier splines for representing curves as they are a powerful representation allowing for independent spatial gradients at the beginning and the end of the curve. Cubic splines employ four control points $\mathbf{P_i}$, hence $k = 3$, and

$$\mathbf{B}(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2t\mathbf{P}_1 + 3(1-t)t^2\mathbf{P}_2 + t^3\mathbf{P}_3. \tag{4.2}$$

A single cubic Bézier spline is unable to fit arbitrarily complex contours. We overcome this problem by borrowing a simple idea from computer graphics: composite, piece-wise Bézier curves (i.e. so-called polybéziers). As indicated in Figure 4.1, polybéziers separate a contour into multiple segments where every segment is represented by a single Bézier spline of limited order. Composite splines stay continuous by simply sharing the ending point of a segment with the starting point of the subsequent segment. Moreover, in order to maintain smoothness, we furthermore make sure that the gradient at the end of one segment coincides with the spatial gradient at the beginning of the next.

Imposing these constraints is simply done by making the control points a function of other latent variables that are shared among neighbouring segments. On one hand, the continuity simply requires the first control point of one segment to be equal to the last control point of the previous segment. Let $\mathbf{P}_0^i \dots \mathbf{P}_3^i$ be the control points of segment i. Furthermore, let the first control point of a segment also be denoted by $\mathbf{Q}_i$. We obtain $\mathbf{P}_0^i = \mathbf{Q}_i$ and $\mathbf{P}_3^i = \mathbf{Q}_{i+1}$. On the other hand, sharing the gradient is made possible by explicitly introducing the local direction at the beginning of each segment, denoted $\mathbf{v}_i$. $\mathbf{v}_i$ is a 3-vector constrained to unit-norm, which means it is a spatial direction with two degrees of freedom only. Since the second control point by definition lies in the direction of the local gradient at the first control point, it may be parametrised as $\mathbf{P}_1^i = \mathbf{Q}_i + d_1^i\mathbf{v}_i$. In order to guarantee smoothness, the third control point of segment i in turn becomes a function of the gradient at the beginning of the next segment, i.e. $\mathbf{P}_2^i = \mathbf{Q}_{i+1} - d_2^i\mathbf{v}_{i+1}$.

In summary, an arbitrary curve is given by a sequence of parameters $\mathbf{Q}_i$, $\mathbf{v}_i$, and $\{d_1^i, d_2^i\}$. By sharing parameters between neighboring segments, a 3D composite Bézier curve is represented as a sequence of cubic Bézier segments where

$$\mathbf{P}_0^i = \mathbf{Q}_i,\ \mathbf{P}_1^i = \mathbf{Q}_i + d_1^i\mathbf{v}_i,\ \mathbf{P}_2^i = \mathbf{Q}_{i+1} - d_2^i\mathbf{v}_{i+1},\ \mathbf{P}_3^i = \mathbf{Q}_{i+1}. \tag{4.3}$$

We parametrize the normal vectors $\mathbf{v}_i$ minimally by making them a function of two rotation angles $\boldsymbol{\theta}_i$, which avoids the addition of side-constraints during optimization. However, in order to facilitate optimisation and avoid the gimbal lock, the parametrisation is local about an initial direction expressed by a pre-rotation $\mathbf{R}_0$, i.e. $\mathbf{v}_i = \mathbf{R}_0\mathbf{v}(\boldsymbol{\theta}_i)$.

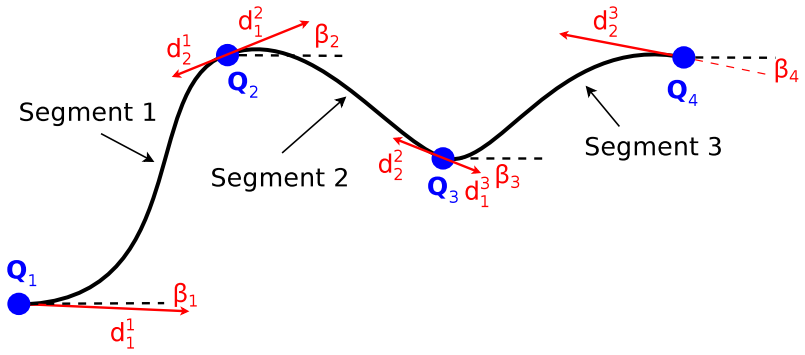


Figure 4.1: Planar example of a smooth curve parametrization with 3 Bézier spline segments.

4.2 Planar Bézier Spline

We start with a simple scenario where all contours are placed on a planar scene. We use a sparse point-based method to bootstrap the problem and giving an initial guess of the camera poses. Then we use a Bézier spline based method to further refine the accuracy of the pose estimation. As previously described, we represent each curve with a sequence of Bézier segments rather than one long segment with many control points. We extract the edges in the images using canny edges detection [40] and split them into segments of the same length of pixels. For planar scenario, there is no need to represent the curves in full 3D space so we replace the $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$ in equation 4.3 with only one gradient ϕ_i and ϕ_{i+1}. Therefore, an example of a curve with 3 Bézier spline segments on a planar case can be indicated in Figure 4.1. Let us denote the sequence of starting and ending points of the curve segments with $\{\mathbf{Q}_1, \cdots, \mathbf{Q}_n\}$. Let segment i simply be the segment between $\mathbf{Q}_i$ and $\mathbf{Q}_{i+1}$. The control points $\left\{\mathbf{P}_1^i, \cdots, \mathbf{P}_4^i\right\}$ of segment i are then given by:

$$\mathbf{P}_0^i = \mathbf{Q}_i,\ \mathbf{P}_1^i = \mathbf{Q}_i + d_1^i \begin{bmatrix} cos\phi_i \\ sin\phi_i \end{bmatrix},\ \mathbf{P}_2^i = \mathbf{Q}_{i+1} - d_2^i \begin{bmatrix} cos\phi_{i+1} \\ sin\phi_{i+1} \end{bmatrix},\ \mathbf{P}_3^i = \mathbf{Q}_{i+1} \tag{4.4}$$

4.2.1 Incremental sparse initialization

The entire sparse initialization part relies on local invariant keypoint extraction and matching in order to compute an initial guess of camera poses as well as the relative positioning of the ground plane. In case of a planar scene, the mapping of points from one image to another can be easily expressed by a homography transformation [15]. We, therefore, bootstrap the sparse computation by first collecting three frames with sufficient frame-to-frame disparity (called *keyframe*), and then find their relative pose by a robust computation of the *planar homography* between these frames. We use three frames because the subsequent decomposition of the homography into relative rotation, translation, and plane normal leads to two possible solutions [52]. In order to resolve the ambiguity, we simply compute two planar homographies for distinct pairs of frames of the first three keyframes and then compare the similarity of the plane normal vectors after rotating them into one common frame.

Once the first relative poses, as well as the normal vector and the depth of the background plane, are identified, we can define a world frame such that the normal vector

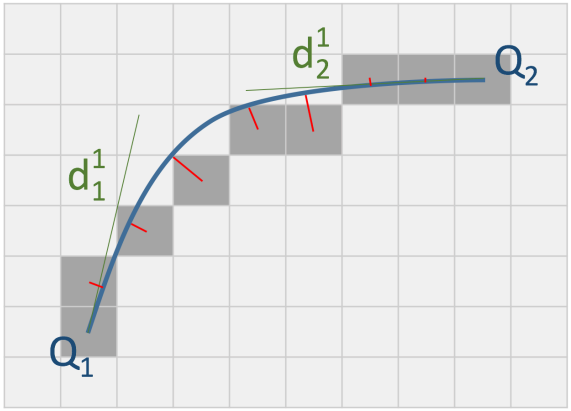


Figure 4.2: Example of a spline segment. The spline initialization consists of minimizing the sum of orthogonal distances from each point on the edge to the spline (in red) over d_1^1 and d_2^1.

of the ground plane is aligned with the vertical axis $\mathbf{e}_z$, and the height of the plane is $z = 0$. All points that have been identified as inliers during the robust planar homography computation can furthermore—since effectively lying on the ground plane—be initialized by simply intersecting the spatial measurement ray from any of the first three keyframes with the ground plane. Subsequent frames can now be aligned to the existing sparse background model by simple robust camera resectioning. In our work, we use P3P [53] embedded into Ransac [54] followed by nonlinear optimization to align subsequent frames with the existing model.

We keep defining and storing keyframes during tracking whenever the frame-to-frame disparity exceeds a given threshold. We furthermore keep adding new 3D points to the background model as parts of the background may have been occluded in earlier frames. We notably use the result from the silhouette extraction to define whether or not a point belongs to the background. Points keep being added to the background by simply intersecting spatial rays with the background plane. The sparse tracking and mapping part is concluded by a windowed bundle adjustment implementation that optimizes the 6 DoF pose of every keyframe, as well as the 2 DoF position of every point on the background plane.

4.2.2 Bézier spline parameter initialization

The initialization of a Bézier spline segment turns out to be easy. We first initialize them in the image. An edge is defined as a set of pixels $\mathbf{C} = \{\mathbf{p}_1, \cdots, \mathbf{p}_l\}$. The first and last control points of each Bézier spline segment (i.e. the sequence $\{\mathbf{Q}_1, \cdots, \mathbf{Q}_n\}$) are simply initialized by subsampling $\mathbf{C}$ with regularly spaced points. The density of points is defined by considering the local curvature: The higher the curvature, the smaller the distance between $\mathbf{Q}_i$ and $\mathbf{Q}_{i+1}$. The local gradients $\{\phi_i, \cdots, \phi_{i+1}\}$ can also be estimated straight from the data. The only parameters that remain to be initialized are d_1^i and d_2^i. They can be initialized for each segment individually by searching along the gradient directions near $\mathbf{Q}_i$ and $\mathbf{Q}_{i+1}$.

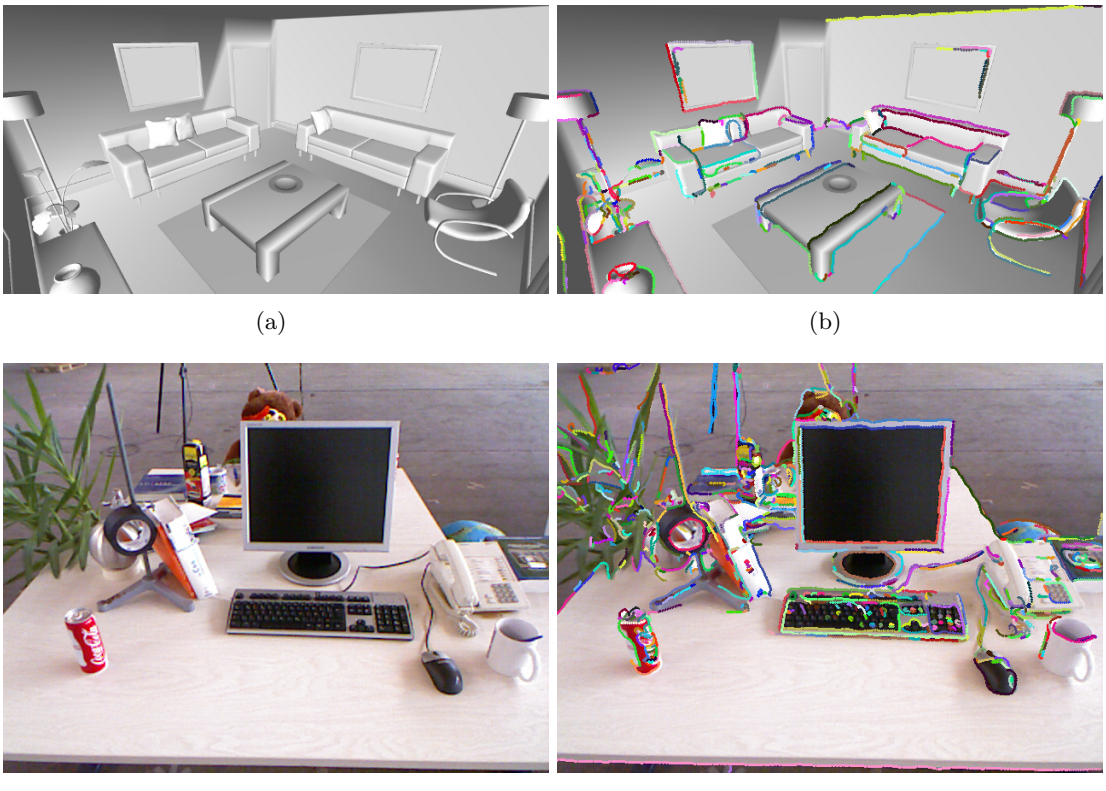
Substituting (4.4) in (4.2) and considering $\mathbf{Q}_i$, $\mathbf{Q}_{i+1}$, ϕ_i, and ϕ_{i+1} to be fixed, each Bézier spline segment $\mathbf{B}_i$ can be regarded as a function of the three scalars d_1^i, d_2^i and t. If $\mathbf{C}_i$ furthermore denotes the subset of p pixels belonging to segment i, the objective for

initializing the remaining parameters of each segment can be formulated as

$$\left\{\hat{d}_1^i, \hat{d}_2^i\right\} = \underset{d_1^i, d_2^i}{argmin} \sum_{j=1}^{p} \underset{t}{min} \|\mathbf{C}_i[j] - \mathbf{B}_i(d_1^i, d_2^i, t)\|^2 \tag{4.5}$$

The energy is also illustrated in Figure 4.2. The minimum location for this energy is simply found by applying Gauss-Newton with numerical Jacobian computation. However, the objective is not entirely trivial to compute, as the derivation depends on an internal minimization over the curve parameter t to find the nearest point on the spline. We re-derive the values for t before each Gauss-Newton iteration and notably do so via a simple 1D bisection search. It is intuitively clear that—under the assumption of sufficiently small curvature—finding the optimal t is likely a convex problem, which causes the bi-sectioning search to converge very quickly.

The Bézier spline parameters are first computed in a reference frame and then projected into the ground plane by again intersecting the rays corresponding to the control points in the image with the ground plane itself. As explained above, due to certain transformation invariance properties of Bézier splines, this projection results in sufficiently good initial values for the subsequent global optimization.



(a) (b)

(c) (d)

Figure 4.3: The original image is shown in the left while the Bézier spline fitting result is shown on the right

4.2.3 Global Optimization

The poses and the background model are finally optimized in a joint curve-based bundle adjustment implementation. Let us assume that there are in total n Bézier spline segments. We define the vector $\mathbf{b}_i = \begin{bmatrix}\mathbf{Q}_i^T & \mathbf{Q}_{i+1}^T & \phi_i & \phi_{i+1} & d_1^i & d_2^i\end{bmatrix}^T$ as the vector of parameters defining the Bézier spline $\mathbf{B}_i$, which may hence be written as a function $\mathbf{B}(\mathbf{b}_i, t)$. It is clear that many of the parameters are shared among different segments, but we ignore this here for the sake of a simplified notation. Let us furthermore assume that we have m camera poses and that the pose of each camera is parametrized by the 6-vector π_j. The objective of our global optimization may then be formulated as minimizing the distance between reprojected samples of each Bézier spline segment and their closest edge points in each one of the keyframes. If $\mathbf{C}^j$ denotes all pixels along an edge in keyframe j, and $\eta(\mathbf{x}, \mathbf{C}^j)$ a function that returns the nearest neighbour of $\mathbf{x}$ within $\mathbf{C}^j$, the optimization objective may be formulated as

$$\left\{\hat{\pi}_1, \cdots, \hat{\pi}_m, \hat{\mathbf{b}}_1, \cdots, \hat{\mathbf{b}}_n\right\} = \tag{4.6}$$
$$\underset{\pi_1,\cdots,\pi_m,b_1,\cdots,b_n}{argmin} \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{t=0}^{9} \| f_{\pi_j}\left(\mathbf{B}(\mathbf{b}_i, 0.1\cdot t)\right) - \eta\left(f_{\pi_j}\left(\mathbf{B}(\mathbf{b}_i, 0.1\cdot t)\right), \mathbf{C}^j\right) \|^2.$$

Note that $f_{\pi_j}(\mathbf{x})$ here denotes the transformation from a world frame into the image plane of a camera with pose parameters π_j. Besides the extrinsic pose parameters, this function also makes use of a suitable camera model with known, pre-calibrated intrinsic parameters (omitted again for the sake of the simplicity of the notation). As can be observed, the internal sum iterates over t and—through the multiplication with 0.1—produces 10 homogeneously distributed samples for reprojection error computation on each Bézier spline segment. We keep this number fixed, although—in the future—we plan to investigate an adaptive number of sampling points depending on the length of the spline segment. Missing data and occlusions that potentially cause outlier residuals are handled by adding a robust Huber norm to the computation.

In order to compute the nearest neighbour efficiently, we follow [55] and precompute a nearest-neighbour map for the edges of each keyframe. We fix the nearest neighbour point during the numerical Jacobian matrix computation, because a small change in the reprojected location could otherwise lead to very large residual changes due to an unexpected, significant change of the nearest neighbour if operating at the center between two distinct curves. This furthermore requires a projection of the residual vectors onto the local gradient direction. The interested reader is invited to look at [55] for further details.

4.2.4 Experimental results

We evaluate the planar Bézier spline based modelling using a self-captured dataset of a planar target with a curve-rich texture. We print out a structured texture—a zebra pattern—for use as a ground. We use a consumer grade, hand-held camera to simply capture a continuous sequence of images while moving around the target. We evaluate our spline-based background model by comparing it against the edges directly extracted from the original image of the background pattern we printed out. We furthermore analyze the residual error and the optimized trajectory before and after spline-based bundle adjustment.

There is no ground truth pose information, which makes it difficult to evaluate the

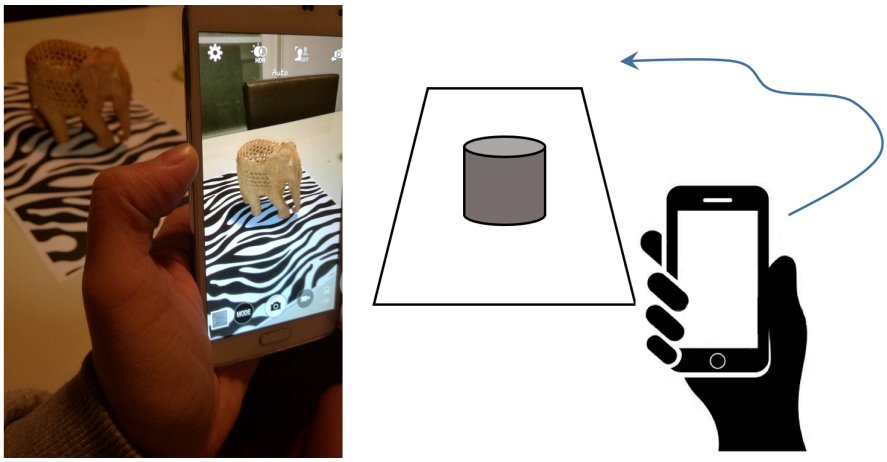


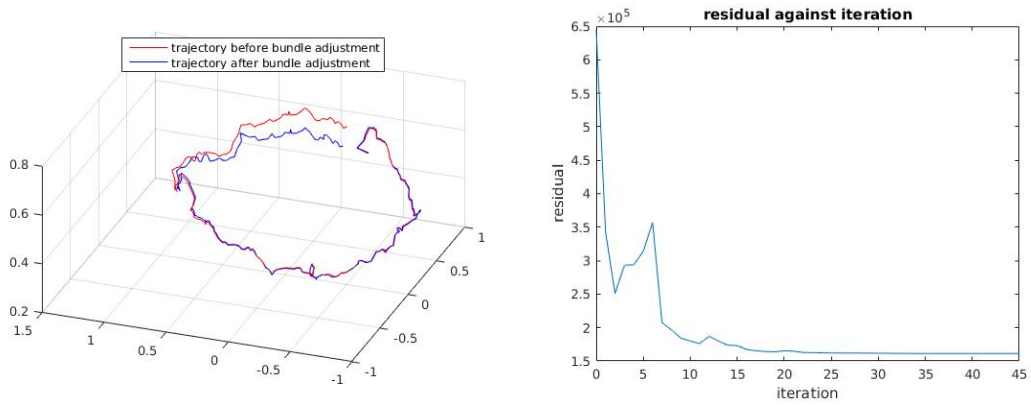
Figure 4.4: Data Collection for the experiment

improvement of the camera poses quantitatively. However, as can be observed in Figure 4.5(a), the change in the camera poses is quite substantial. Figure 4.5(b) furthermore shows the overall residual error during each iteration of the non-linear refinement. As can be observed, the residual generally goes down and converges at the end. From this result, we can conclude that the camera poses are optimized as well.

We solve the non-linear least squares problem (4.6) using the Levenberg-Marquardt implementation of the Ceres-Solver library [56]. The latter is an open source C++ library that is able to solve large-scale non-linear optimization problems. The performance of the optimizer is stable as long as the initial guess of the camera poses is not too far off.

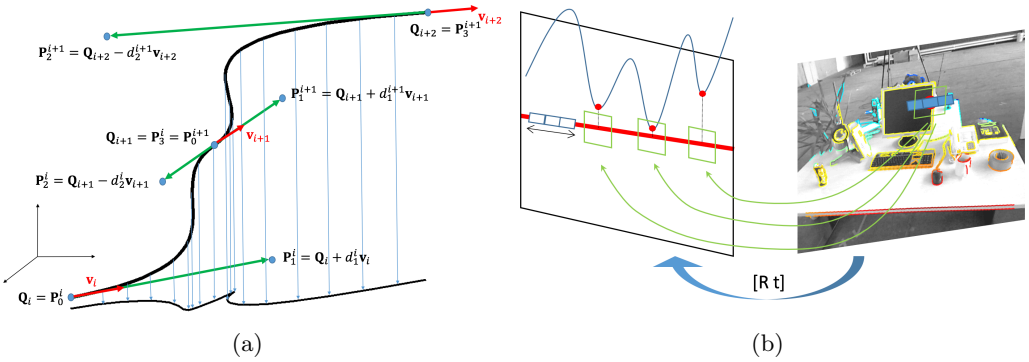
4.2.5 Discussion

The core contribution of our work consists of a successful demonstration of how to use free-form curve models for modeling the environment, and how this can possibly help to increase the accuracy of regular monocular structure from motion. This is demonstrated through the improved dense reconstructions we obtained from our space carving framework. This



(a) (b)

Figure 4.5: (a): Camera poses before and after optimization. (b): Development of the overall residual during the non-linear refinement of the curves.



(a) (b)

Figure 4.6: Left: Two segments of a smooth 3D Bézier curve. Some of the optimization variables (the control points $\mathbf{Q}_i$ and the gradient directions $\mathbf{v}_i$) are shared among adjacent segments. Right: Epipolar matching with 1D patches. The three best local minima are sub-sequently disambiguated by 2D patch matching. The colors in the right frame indicate the depth of pixels and thus show an example result of semi-dense matching.

work is currently working in a simple senario where most of the background contours are observed in every key-frames and occulision is rarely happens so the following work is obvious: we intend to use the presented Bézier spline parametrization for refining general non-planar sparse structure-from-motion results and modeling more complex environment.

4.3 3D Bézier Spline

As the following work of section 4.2, we are aiming to achieve a fully automatic structure from motion pipeline that using free from Bézier spline for both environment representation and bundle adjustment. In section 4.3.1, we will demonstrate how do we initialize the problem as well as how to we initialize the Bézier spline based mapping. The overall system flow will be shown in subsection 4.3.2 as well as a detailed tracking strategy. The experimental result is shown in section 4.3.3 and a conclusion of this works is given in subsection 4.3.4

4.3.1 Initialization of Bézier splines

As we will see in Section 4.3.2, our method relies on a sparse technique to first initialize the poses of all frames in a video sequence. We proceed by extracting Canny-edges [40] in each image and grouping them into curves based on simple connectivity and thresholding of local curvature. We then initialize the depth of each pixel on an edge by using a variant of the semi-dense epipolar tracking method presented in [36]. For each pixel within each group, we perform the following steps to recover the depth:

- Find a good reference frame for stereo matching by considering the length of the baseline and the parallelism between the epipolar direction and the local image gradient.
- Extract the epipolar line in the reference frame.
- Perform a 1D search for photometric consistency along the epipolar line by comparing 1D image patches.

- Short-list the three best local minima.
- For each local minimum, perform a 2D patch comparison to find the best.
- Minimise the photometric error via sub-pixel refinement of the disparity along the epipolar line.
- Take a robust average among all the depths recovered within a local window.

The procedure is visualised in Figure 4.6(b). Knowing camera poses as well as semi-dense depth maps in each frame, it now becomes possible to initialize the 3D position of points along the curve expressed in world coordinates $\mathbf{p}_w$

$$\mathbf{p}_w = \mathbf{R}_w(d\mathbf{K}^{-1}\mathbf{p}) + \mathbf{t}_w \tag{4.7}$$

where $\mathbf{R}_w$ and $\mathbf{t}_w$ are the rotation and translation transforming a point $\mathbf{p}$ from the camera to the world coordinate frame, $\mathbf{K}$ is the matrix of intrinsic camera parameters, and d if the depth of the point along the principal axis. We transform all points for which a depth has been recovered into 3D, and furthermore separate the contours into segments such that each segment has roughly the same number of pixels. We furthermore assume that each segment can now be correctly represented by a cubic Bézier spline. We finally estimate the control points of each segment by counting the number of pixels composing the segment, sampling an equal number of values for t distributed homogeneously within the interval $(0, 1)$, and assuming that the Bézier spline evaluated at those continuous curve parameter values leads exactly to the hypothesised world point. Let us assume that there are m world points. Under the above assumption, the curve parameter for each one of the points becomes

$$t_k = \frac{k}{m-1}, k \in \{0, \ldots, m-1\} \tag{4.8}$$

Using (4.2), we can then find the control points by constructing and solving the linear problem

$$\begin{pmatrix} (1-t_0)^3\mathbf{I} & 3(1-t_0)^2t\mathbf{I} & 3(1-t_0)t^2\mathbf{I} & t_0^3\mathbf{I} \\ \cdots & \cdots & \cdots & \cdots \\ (1-t_{m-1})^3\mathbf{I} & 3(1-t_{m-1})^2t\mathbf{I} & 3(1-t_{m-1})t^2\mathbf{I} & t_{m-1}^3\mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \end{pmatrix} = \begin{pmatrix} \mathbf{p}_{w,0} \\ \cdots \\ \mathbf{p}_{w,m-1} \end{pmatrix} \tag{4.9}$$

Note that the left-hand matrix only depends on a discrete number of homogeneously sampled values for t, and therefore can be computed upfront. To conclude the initialisation, the Bézier segments are grouped into curves following the same order of the original segments extracted in the image. To enforce continuity and smoothness in the initialised curve, the first control point of each segment is simply replaced by the last control point of the previous segment, and the direction at the link point and the distance from the link point are set to

$$\begin{aligned} \mathbf{v}_i &= 0.5 * (\mathbf{P}_4^{(i-1)} - \mathbf{P}_3^{(i-1)})/\|\mathbf{P}_4^{(i-1)} - \mathbf{P}_3^{(i-1)}\| \\ &+ 0.5 * (\mathbf{P}_2^i - \mathbf{P}_1^i)/\|\mathbf{P}_2^i - \mathbf{P}_1^i\| \\ d_1^i &= \|\mathbf{P}_2^i - \mathbf{P}_1^i\| \\ d_2^i &= \|\mathbf{P}_4^i - \mathbf{P}_3^i\|. \end{aligned}$$

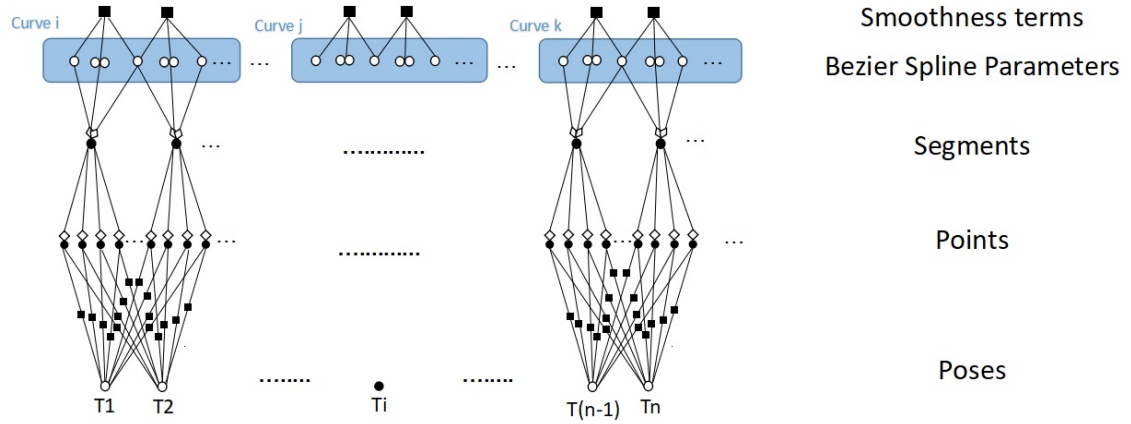


Figure 4.7: Factor graph of our optimisation problem. Curves are composed of Bézier segments, which in turn are sampled to return 3D points. The latter are reprojected into frames if a correspondence between this segment and that frame exists. The curve parameters are not directly optimised, but depend on latent variables some of which are shared among neighbouring segments (bordering control points, as well as curve directions in those points).

4.3.2 Fully automatic, spline-based structure from motion

This section explains the structure of our curve-based structure-from-motion problem, which can notably be formulated as a graph optimisation problem. The first part of the section assumes that the structure of this graph is already initialised, and in turn focusses on how the individual registration costs as well as the overall bundle adjustment are computed. The second part of the section then presents the detailed flow-chart of our incremental structure-from-motion, and in particular provides all the details on the automatic management of the graph structure (i.e. correspondences between segments and frames).

Curve-based structure-from-motion as a graphical optimisation problem

The overall factor graph of our optimisation problem is illustrated in Figure 4.7. Our map is given as a set of composite Bézier curves, each one being composed of one or multiple cubic Bézier splines, which we call here *segments*. However, as explained in the previous section, in order to ensure continuity and smoothness in the curves, we do not optimise the control points of the segments directly. The control points of the segments are dependent on latent variables, which potentially are even shared across neighbouring segments. With respect to Figure 4.7, these parameters are simply called *Bézier spline parameters*. In order to prevent the latter from collapsing or drifting off into unobservable directions, we have our first cost terms added to the graph, which are regularisation constraints on the Bézier spline parameters. We then sample points from each segment, and reproject them into frames for which a correspondence has been established. The second type of cost term occurs here in the form of our 3D-to-2D curve registration loss. The correspondence management will be discussed later, here we focus on the cost terms and the actual optimisation of the graph.

The parametrization is very similar to the previous work with a few modification that suit for 3D curve representation. Let us define the vector $\mathbf{b}_i = \begin{bmatrix}\mathbf{Q}_i^T & \mathbf{Q}_{i+1}^T & \boldsymbol{\theta}_i^T & \boldsymbol{\theta}_{i+1}^T & d_1^i & d_2^i\end{bmatrix}^T$ as the vector of parameters defining the Bézier spline

$\mathbf{B}_i$, which may hence be written as a function $\mathbf{B}(\mathbf{b}_i, t)$. We assume to have n splines. Let us furthermore assume that we have m camera poses and that the pose of each camera is parametrized by the 6-vector $\delta\pi_j$ that expresses a local change with respect to the original pose $\pi_{j,0}$. Let $f_{\delta\pi_j+\pi_{j,0}}(\cdot)$ be a function that transforms a point from the world frame into the image plane of a camera at position $\delta\pi_j + \pi_{j,0}$. The function f assumes and uses known camera intrinsic parameters. Let $\mathbf{C}^j$ denote all pixels along an edge in keyframe j, and $\eta(\mathbf{x}, \mathbf{C}^j)$ a function that returns the nearest pixel on an edge (i.e. within $\mathbf{C}^j$) to a reprojected image location $\mathbf{x}$. The final objective of the global optimisation is given by

$$\left\{\delta\hat{\pi}_1, \cdots, \delta\hat{\pi}_m, \hat{\mathbf{b}}_1, \cdots, \hat{\mathbf{b}}_n\right\} = \underset{\delta\pi_1,\cdots,\delta\pi_m,b_1,\cdots,b_n}{argmin} E_{\text{geo}} + \lambda_1 E_{s1} + \lambda_2 E_{s2}, \tag{4.10}$$

where

$$\begin{aligned}
E_{\text{geo}} &= \sum_{i=1}^{n}\sum_{j=1}^{m}\mathbf{1}_{ij}\sum_{k=0}^{s}\mu\left(\left(\mathbf{x}_{ijk} - \eta(\mathbf{x}_{ijk}, \mathbf{C}^j)\right)^T \cdot g(\eta(\mathbf{x}_{ijk}, \mathbf{C}^j))\right) \\
\mathbf{x}_{ijk} &= f_{\delta\pi_j+\pi_{j,0}}\left(\mathbf{B}(\mathbf{b}_i, t_k)\right) \\
E_{s1} &= \sum_{i=1}^{n}(\|\mathbf{Q}_i - \mathbf{Q}_{i+1}\| - \Delta\mathbf{Q}_{i,0})^2 \\
E_{s2} &= \sum_{i=1}^{n}\left(\measuredangle^2(\mathbf{Q}_{i+1} - \mathbf{Q}_i, \mathbf{v}(\boldsymbol{\theta}_i)) + \measuredangle^2(\mathbf{Q}_{i+1} - \mathbf{Q}_i, \mathbf{v}(\boldsymbol{\theta}_{i+1}))\right).
\end{aligned}$$

$\mathbf{1}_{ij}$ is an indicator function that equals to 1 if the segment i is visible in frame j, and otherwise to 0. Index k runs from 0 to s causing a homogeneous sampling of 3D points along the segment i through the continuous curve parameters t_k. $\mathbf{x}_{ijk}$ results as the k-th 3D point sampled along the spline i and reprojected into the view j. E_{geo} as a result denotes the geometric registration cost given as the sum of disparities between reprojected 3D points $\mathbf{x}_{ijk}$ and their nearest neighbours in $\mathbf{C}^j$. A projection onto the local gradient direction $g(\eta(\mathbf{x}_{ijk}, \mathbf{C}^j))$ is added in order to facilitate optimisation in the *sliding situation*. Note that, as discussed in [57], this step also helps to efficiently overcome the bias discussed in [31] without having to employ the more expensive technique of variable lifting. To conclude, a robust norm $\mu(\cdot)$ such as the Huber norm is added to account for outliers and missing data.

Besides the residuals in the form of curve alignment errors, we additionally have the regularisation costs E_1 and E_2, which are weighted in using the trade-off parameters λ_1 and λ_2. The regularization terms are added to the cost function in order to prevent convergence into wrong local minima. E_1 enforces the length of each segment (i.e. the distance between the first and the last control point) to be consistent with its original value after initialization. The term introduces a penalty in the situation where the algorithm aims at collapsing a segment such that the entire segment would match to a single pixel, and thus return a very low registration cost. Second, because curves are composed of relatively short segments, we want to prevent single segments from presenting too high curvature. E_2 penalises high curvature by making sure that the spatial curve directions in the first and the last control point (indicated by $\mathbf{v}(\boldsymbol{\theta}_i)$ and $\mathbf{v}(\boldsymbol{\theta}_{i+1})$) do not deviate too much from the vector between points $\mathbf{Q}_i$ and $\mathbf{Q}_{i+1}$. E_2 is particularly helpful to prevent uncontrolled curve behaviours in a situation where depth his badly observable.

Initial poses are given from a sparse initialisation, and the initial values for the Bézier splines are obtained using the procedure outlined in Section 4.3.1 (further details about

graph management and initialisation are given in the following section). After initialisation, we solve the bundle adjustment problem (4.10) using an off-the-shelf nonlinear implementation of Levenberg-Marquardt [56]. The latter iteratively performs local linearizations of the residual and regularisation terms in order to gradually update the pose and Bézier spline parameters.

Efficient residual error computation

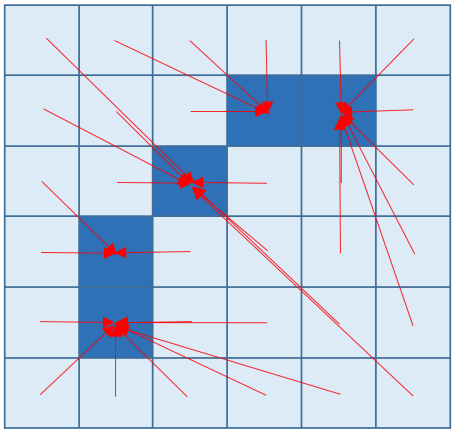


Figure 4.8: Example nearest neighbour field.

One of the more expensive parts of the computation is given by the nearest neighbour look-up $\eta(\mathbf{x}, \mathbf{C}^j)$. Inspired by [55], we employ a simple solution to speed up the optimisation by pre-computing a nearest neighbour look-up field that indicates the nearest pixel on an edge for any pixel in the entire image. An example is given in Figure 4.8, where each pixel in the nearest neighbour field storing a reference to its nearest edge pixel. The extraction of the nearest neighbour field is accelerated by limiting it to pixels which are at most 15 pixels away from an edge.

Overall flow-chart

The input of the system is simply a sequence of RGB images from a calibrated camera. Before we initialize the Bézier Map, we perform ORB SLAM [16] to obtain an initial guess for the camera positions. ORB SLAM is a sparse feature based simultaneous localization and mapping system which can provide an accurate guess of the camera position in real-time. With the initial camera positions in hand, we then incrementally parse our frames and initialise new Bézier splines. Each time a new keyframe is added, we first establish the correspondence with existing segments before adding potentially new segments. The initialisation of splines from a single frame uses the strategy proposed in Section 4.3.1. A flow-chart of the overall system is indicated in Figure 4.9.

The Bézier splines are grouped into two distinct maps, one global map that stores well observed splines and a temporary map that stores new spline initialisations. All Bézier splines are initially put into the temporary map and then moved to the global map once sufficient observations are available. This delayed initialisation scheme helps to robustify the optimisation, as bundle adjustment uses only the well-observed splines in the global map. In order to prevent the addition of redundant representations, the establishment of

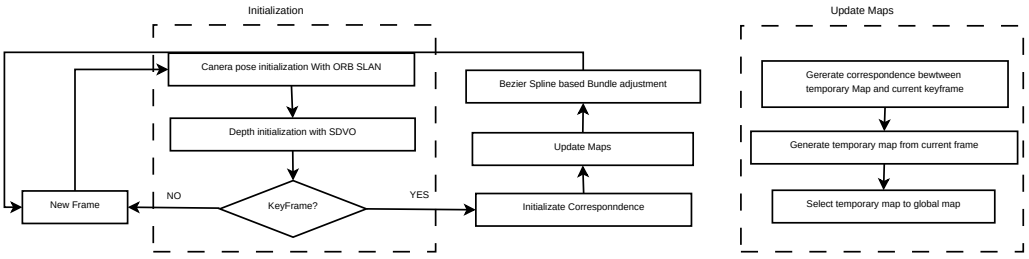


Figure 4.9: Overall flow-chart of our Bézier spline-based structure from motion framework including the initialisation from a sparse point-based method.

correspondences in new keyframes (outlined in Section 4.3.2) needs to first consider the segments in the global map before moving on to the temporary map.

New segments are added to the temporary map whenever a sufficiently large group of connected pixels has not been registered with existing splines, and the semi-dense depth measurement for those pixels succeeded. The segment is added to an existing curve if the seed group of pixels is smoothly connected to the pixels of an already existing curve. To prevent the algorithm from losing too many correspondences in difficult passages, newly initialized Bézier splines may also be added directly into the global map to keep the tracking of subsequent frames alive. The addition of a new keyframe is concluded by local bundle adjustment overall recently observed frames and landmarks in the global map. Segments with less than three observations in keyframes will not be considered for updating the pose of the cameras. We alternately fix the parameters of Bézier splines and camera poses and optimize the other. After all key-frames have been loaded, we perform global bundle adjustment over all frames and splines.

Correspondence establishment

In this section, we are going to explain how we establish and manage the correspondences between segments and key-frames. Correspondences are verified based on four criteria:

- Spatial distance: We require the initial geometric registration cost to be small enough. Points from a spline are required to consistently reproject near a set of connected edge pixels in the image. We evaluate the average reprojection error. Only splines with an average error lower than a given threshold will be considered as an inlier correspondence.

- Appearance-based error: We store the average color of a segment in its original observation. We do not only consider the pixels forming the edge itself, but an isotropically enlarged region around each segment. We again set a threshold on the difference in appearance for determining inliers.

- Viewing direction: Since the appearance in the neighbourhood of edges can depend heavily on the viewing direction (in particular for occlusion boundaries), we add a limitation on the range of possible viewing directions. We assume the viewing direction of an observation to be the vector from the camera center to center of a segment transformed into the world frame. A correspondence is no longer established if the current viewing direction has an angle of more than sixty degrees away from the average viewing direction of all previous observations.

- Depth check: We check the relative depth of each reprojected segment, and discard segments with negative depth.

We further add correspondence pruning based on weak overall curve observations. For curves where more than fifty percent of all segments have no longer been observed in the three most recently added keyframes, the entire curve will be disabled. A disabled curve will no longer be used in local bundle adjustment until it is reactivated by sufficiently new observations in a new frame.

The average color error between a segment in its original image and the closest pixel retrieved via the nearest neighbor field is evaluated as follows. We represent the color in HSV format where, thus returning the average hue H, saturation S, and lightness L values. Operating in the HSV color space can be more robust to illumination changes and difference caused by viewing direction changes. However, due to its definition, the hue value becomes unobservable at zero saturation. Errors in H therefore need to be scaled by the average saturation. The error between two segments is finally given by

$$\mathbf{E} = \lambda * \frac{S_1 + S_2}{2} * \mathbf{min}(|H_1 - H_2|, 2\pi - |H_1 - H_2|) + (L_1 - L_2) \tag{4.11}$$

From experiment, it is found that rather than finding correspondences as much as possible, it is better to make sure all the correspondences are good in quality. We measure the errors of the correspondences in each criteria and only the matches where every error are below a certain threshold are selected as inlier matches.

4.3.3 Experimental evaluation

The algorithm is implemented in C++ and depends on OpenCV and the Google Ceres-solver [56] for solving our curve-based bundle-adjustment. We evaluate the pipeline on several indoor and outdoor images from open-source benchmark sequences [51, 1, 58]. Although some of them are RGB-D datasets, we only use the RGB channel in our pipeline. For initialization, we precompute the camera poses using ORB-SLAM [16], and also compare our solution against its global optimization result. We evaluate our result in terms of both the accuracy of the camera trajectory and the quality of the environment mapping.

Evaluation on Simulation Datasets

Before we test our pipeline on a larger scale dataset, we perform a dedicated experiment where the performance of edge-based registration is analysed and compared against a state-of-the-art point-based solution in different environments. Each test is generated by taking an image of the real-world, assuming it to be a planar environment, and then generating novel views by assuming a circular orbit on top of the plane (new views can easily be generated by homography warping). This allows us to work with features from the real world, but at the same time focuses the experiment on the actual accuracy of the estimation (i.e. issues related to the correspondence establishment are not taken into account). It furthermore allows us to explore a larger variety of environments while each time maintaining information about the ground-truth trajectory.

Three types of images are explored: logos, indoor environments (images are taken from [51]), and outdoor environments (images taken from [58]. We analyze three images for each category while each time adds a varying amount of Gaussian noise to each individual image. Noise is added by adding a random per-pixel intensity disturbance of 0% to 20%.

Noise	0%	5%	10%	15%	20%
Logos (m)	0.016215	X	X	X	X
Indoor imgs (m)	0.000703	0.000738	0.000691	X	X
Outdoor imgs (m)	0.001634	0.000962	X	X	X

Table 4.1: Average Position Error of ORB SLAM on different types of images ('X' means failure)

Noise	0%	5%	10%	15%	20%
Logos (m)	0.000415	0.000312	0.000458	0.0035	0.0111
Indoor imgs (m)	0.000533	0.000533	0.000459	0.0014	0.0046
Outdoor imgs (m)	0.000885	0.0015	0.00153	0.0042	0.0068
Overall (m)	0.000611	0.000798	0.000812	0.003	0.0075

Table 4.2: Average Position Error of our method on different types of images.

A visualization of some patterns plus the distributions of obtained the position errors for the different noise levels are indicated in Figure 4.10.

As can be observed in Tables 4.3.3 and 4.3.3, ORB-SLAM quickly fails as noise is added to the images, especially for the logo patterns which do not contain much texture. In contrast, the proposed curve-based optimization shows that, if a reasonably good initialization for poses is available, a high level of accuracy and robustness can be achieved for all analysed images.

Evaluation on a full 3D dataset

We evaluate the complete algorithm on the living room sequence of the ICML-NUIM benchmark sequence [1]. This synthetic dataset provides realistic images of a camera exploring an indoor environment. Both ground-truth information for poses and 3D model are available, thus permitting the evaluation of both accuracy of motion and quality of the reconstruction. As we use only the RGB channel of the dataset, the recovered scale of the estimation is in fact arbitrary. To properly evaluate the output trajectory, we therefore perform a 7 DoF alignment between the estimated trajectory and ground truth (i.e. a similarity transformation that identifies rotation, translation, and scale for an optimal alignment). The trajectory error is simply the distance between the recovered position and ground truth.

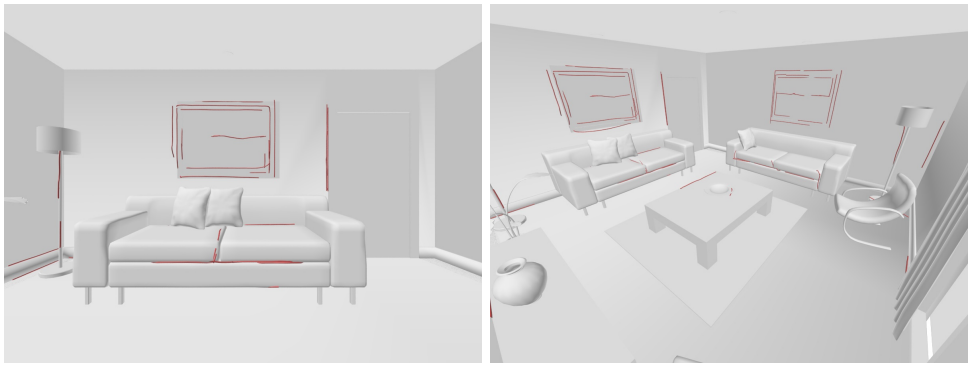
We run the pipeline ten times and compute the average rmse and median of the trajectory error for both ORB SLAM and our Bézier-spline based optimization. The results are indicated in Table 4.3. As can be observed, ORB-SLAM has a lower rmse error while using Bézier splines achieves a smaller median error. While this indicates generally better accuracy, the inferior performance in terms of the rmse error is attributed to a few occasions in the dataset where only few contours are observed, thus leading to no substantial improvement in the optimized pose. Note that, in order to further explore the potential of curve-based optimization, we also analysed the quality if more curves are initialized by also using the available depth channel (note however that depth is only used to initialize the curves, it does not constrain the curves during bundle adjustment anymore). The result is indicated in the last row of Table 4.3. It shows that if sufficient curves can be initialized, curve-based optimization is at last able to competitive with the state-of-the-art point-based approaches. The comparative performance of all alternative algorithms can be found in the paper [59].

Since the ICML-NUIM dataset also provides 3D models of the environment, we can also visualize the quality of the mapping by overlaying some of our estimated curves onto the groundtruth CAD model of the environment. As illustrated in Figures 4.11(a) and 4.11(b), the curves align well with real-world edges, and thus provide a visually appealing, more meaningful representation of the environment than sparse point-based approaches.

Figure 4.10: The first row shows example images for each type of experiment. The second row shows the obtained distributions of position errors between the estimated result and ground truth for varying noise levels. The x-axis shows the error range while y-axis shows the number of errors in that range. For each pattern, we evaluate 5 different noise levels.

Table 4.3: Average errors for ORB-SLAM [16] and our Bézier-spline based optimization.

Algorithm	rmse(cm)	mean(cm)	median(cm)
ORB(Mono)	3.82	3.41	3.02
Bézier	4.45	3.68	2.96
Bézier(RGBD)	4.11	3.07	2.37



(a) (b)

Figure 4.11: Mapping results on the ICML-NUIM living room sequence [1].

4.3.4 Discussion

Our main novelty is a fully automatic structure from motion pipeline where a general, higher-order curve model has been successfully embedded into global bundle adjustment. This result stands in contrast with prior semi-dense visual SLAM pipelines, which alternate between tracking and mapping, and thus are unable to provide a globally consistent, jointly optimised result that explores all correlations between poses and structure. We employ polybéziers, the geometric intuition of which proves great benefits during initialization and regularisation. Our work furthermore illustrates the importance of managing the correspondences between segments and frames, and the resulting graphical form of the optimisation problem. Introducing such correspondences also enables us to prevent the use of the more computationally demanding data-to-model registration paradigm. We present an evaluation of several synthetically generated datasets simulating the appearance of different environments and application scenarios. We demonstrate that it is indeed possible to improve on the accuracy provided by purely sparse methods, and return visually expressive, complete semi-dense models that are jointly optimized over all frames.

Chapter 5

Dense Mapping Application

3D object modeling is a popular sub-branch of computer vision. Important fields of application are given by cultural heritage and 3D content generation for virtual reality applications such as immersive computer gaming or educational software. At least until now, accurate reconstruction of 3D objects is a technology that mostly relies on expensive hardware, depth cameras, or—in the case of visual information—fixed known camera positions. This often impedes the usage in consumer grade applications, where users expect to use simple hand-held hardware such as their private smart phone. Using videos captured by a hand-held device means that the camera poses are unknown and need to be resolved as part of the object reconstruction process, for instance using standard online structure-from-motion or visual SLAM methods. The latter, however, often do not provide sufficiently high accuracy for accurate 3D modeling. In this chapter, we present a full pipeline for 3D object model reconstruction with calibrated hand-held monocular cameras while using Bezier spline based feature for background mapping. Using the result from the previous chapter, we directly fed the result into a space carving based module within which we reconstruct the visual hull of the object. This chapter demonstrates an application of the Bezier spline based structure from motion on a planar scene where the detailed pipeline are shown in section 5.1. The final output from the pipeline is shown in section 5.2

5.1 Framework overview

Let's assume that we have a flat horizontal background with sufficient structured texture on it. The texture does not need to be known in advance, which potentially enables us to apply the method on top of any flat surface that meets the minimum requirements in terms of the available texture. The object to reconstruct is placed on top of the structured background. We then move with a hand-held camera around the object and capture a continuous image stream that is used for solving all tasks: recover camera poses, model the background, and produce our final result, which is a dense 3D model of the object. As an application from the previous chapter, we take the result from the previous chapter and feed directly to the pipeline.

An overview of our end-to-end pipeline is indicated in Figure 5.1. From a high-level perspective, the framework can be divided into two separate processing stages. The first block is an incremental sparse initialization procedure during which we identify initial camera poses with respect to the background plane. During this sparse computation, the background is simply modeled by 2D points lying on the ground plane. The spare time during the sparse initialization is used to prepare each one of the retained keyframes for the

subsequent space carving approach. This notably involves the extraction of the object's silhouette in the image using established segmentation techniques. The second block then contains a sequence of batch optimization algorithms for extracting the dense 3D model. This notably includes the curve-based background modeling step, which is discussed in the previous Chapter.

5.1.1 Silhouette extraction

For each key-frame, we use GrabCut [60] to perform image segmentation and extract the object's silhouette. GrabCut executes graph cut in an MRF to segment out a foreground object from the rest. The corresponding data terms rely on Gaussian Mixture Models (GMMs) for the appearance of both foreground and background pixels. They are initialized from a simple user-defined box drawn around the object. The GrabCut algorithm then alternates between performing the segmentation and refining the foreground and background models using the updated segmentations until convergence.

We chose GrabCut because it permits high quality image segmentations from only very little user input. In our implementation, we require the user to manually draw an initial bounding box around the object in the very first keyframe only. In subsequent keyframes, we find the initial bounding box by pyramidal tracking of the image patch defined by the object's bounding box in a previous keyframe.

5.1.2 Dense 3D modeling

We use space carving to obtain the 3D model of the object. Space carving is a voxel based 3D reconstruction algorithm that aims at approximating the space occupied by the object by cubic voxels. The idea is simple: Each voxel is projected into each keyframe to determine whether it lies inside or outside the silhouettes. If a voxel lies completely outside the silhouette in one of the keyframes, it can be eliminated from the model. Computation is speeded up by a hierarchical, coarse-to-fine voxelization of space: We first aim at culling bigger voxels, and only split them up into 8 smaller voxels if the voxel potentially intersects with the boundary of the object. The approximation error becomes smaller as the resolution of the original voxel grid is growing. The algorithm is furthermore robust against image under-segmentation errors: even if parts of the background end up inside the object's silhouette in one of the keyframes, it will not remain part of the volume as long as there is at least one keyframe where this part is outside the object region.

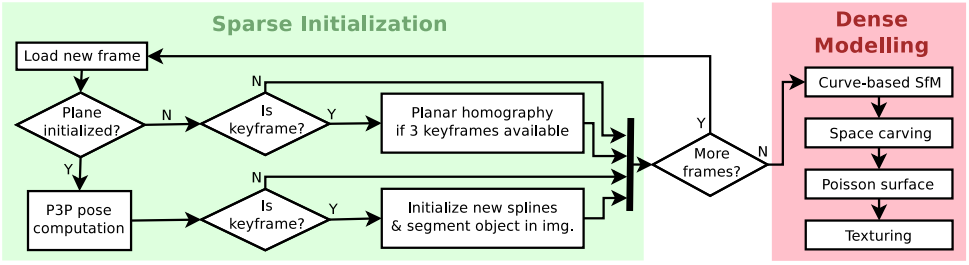
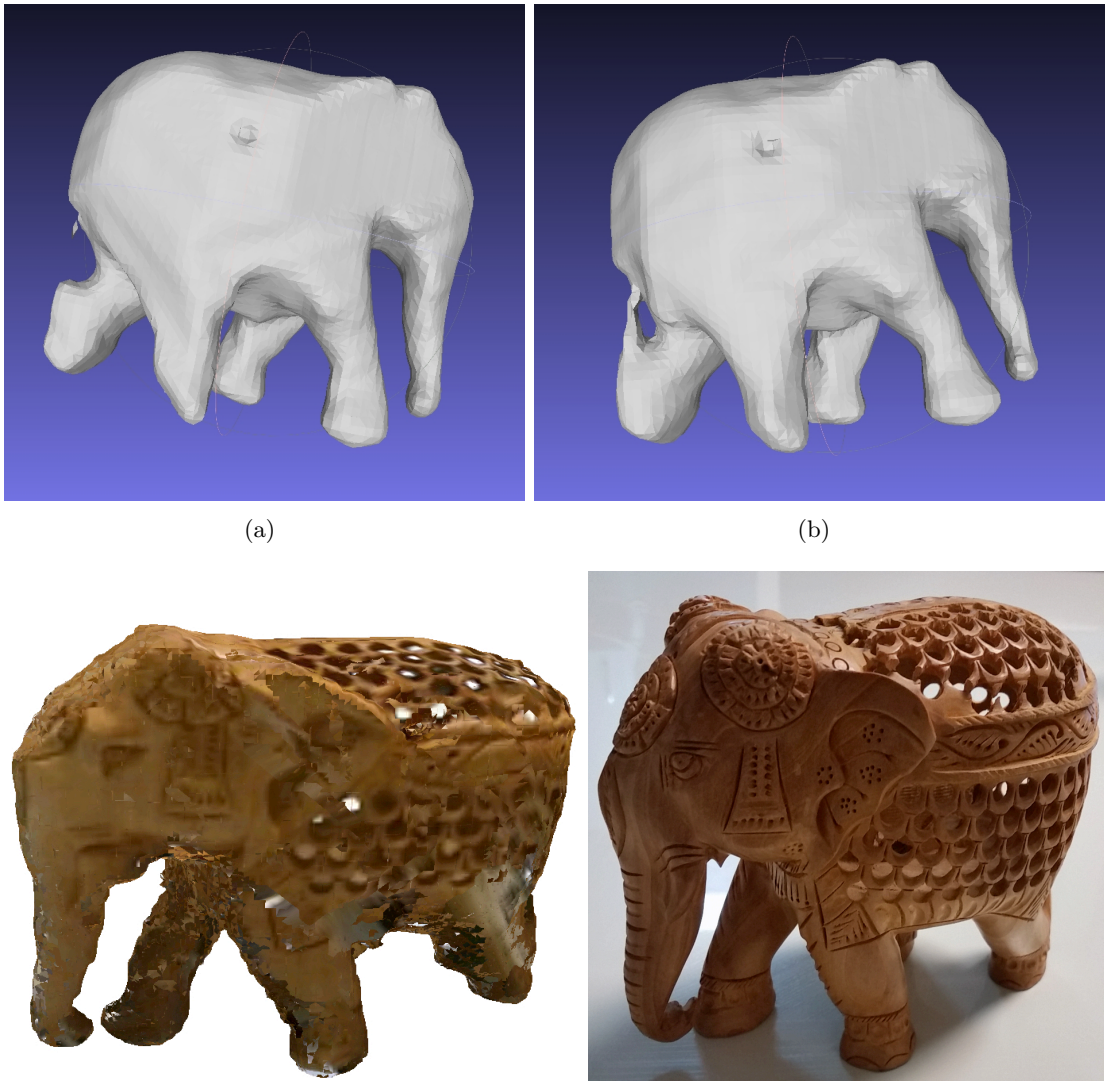


Figure 5.1: System overview. Our pipeline consists of an incremental sparse initialization procedure during which camera poses and the relative orientation of the background plane are identified. The sparse part is followed by a sequence of batch optimization algorithms to recover the final 3D model.



(a) (b)

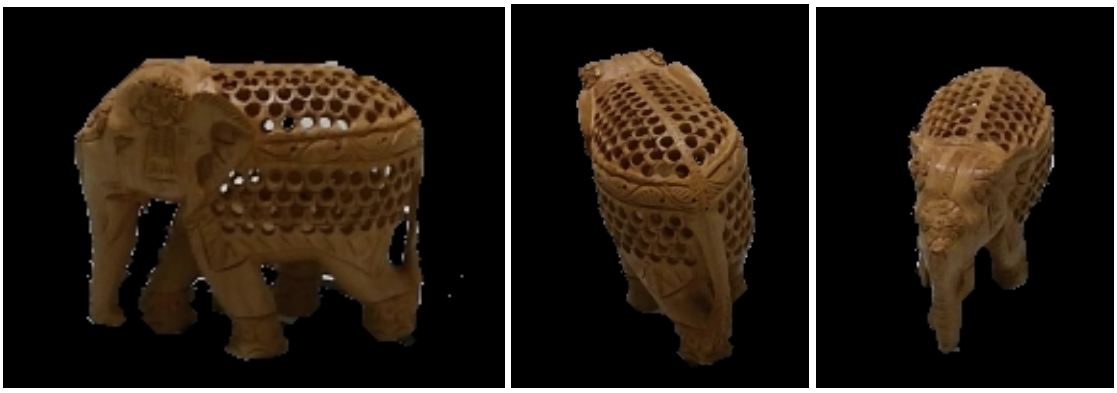
(c) (d)

Figure 5.2: (a): Poisson surface reconstruction from sparse optimization. (b): Poisson Surface reconstruction from curve-based reconstruction. (c): The final textured model. (d): A picture of the original model from a similar view-point.

To conclude, the boundary surface of the object is extracted by identifying all voxel-corner points that are not fully covered by other voxels and deriving a Poisson-surface reconstruction from them. The resulting triangular mesh is finally textured by identifying the most front-parallel keyframe for every triangle and identifying the corresponding piece of texture by reprojecting the vertices of the triangle into that keyframe.

5.2 Evaluation of the dense 3D model

In order to evaluate the importance of the curve-based structure-from-motion part, Figures 5.2(a) and 5.2(b) show some Poisson surface reconstructions obtained from performing space carving with the optimal poses from the sparse and the curve-based optimization, respectively. As can be observed, the reconstruction from sparse poses is losing details such as the tail and part of the foot. This points to bad alignment, as misalignments generally



(a) (b) (c)

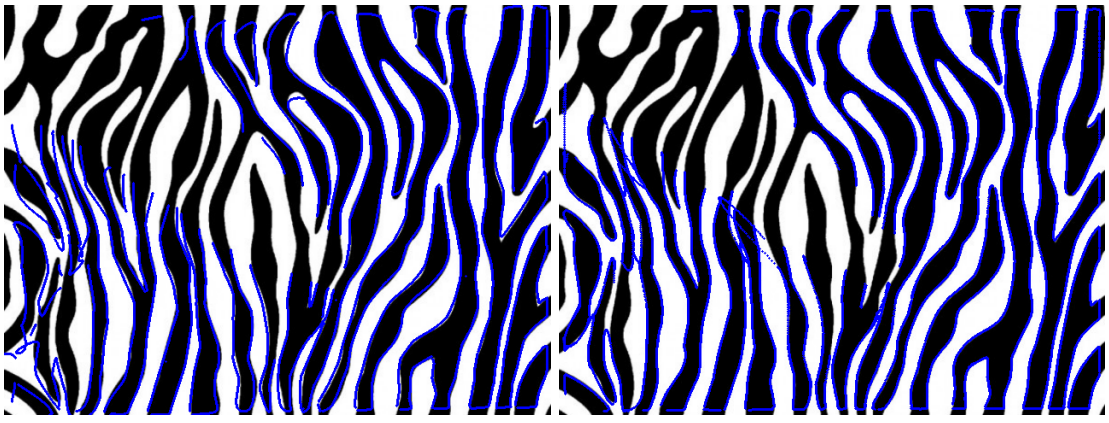
Figure 5.3: Example segmentations from various viewpoints.

cause the intersection volume of space carving to become smaller. As can be observed, the poses from our curve-based optimization module have improved, thus maintaining those details in the reconstruction. Figures 5.2(c) and 5.2(d) show our final textured model and an image of the original object, respectively.

5.2.1 Some segmentation results

The weakly supervised segmentation can of course easily fail if the similarity between the object and the background appearance is too similar. Depending on the shape of the object, the user-defined bounding box also always contains parts of the background as well, which disturbs the initialization of a clean foreground model. Together with specularity, shadows, and reflections, these problems complicate a robust extraction of the object's silhouette and perhaps makes it the weakest part in our current reconstruction pipeline. However, in the example presented here, the segmentation generally returns good results. A few example segmentations are indicated in Figure 5.3.

Evaluation of the spline-based background model



(a) (b)

Figure 5.4: (a): Bezier splines before optimization, registered to the original pattern. (b): Bezier splines after optimization, registered to the original pattern.

The error of our Bezier-spline based background model is computed by registering it directly with the edges extracted from the original pattern image, and evaluating the residual error. The registration is done by a continuous minimization of the sum of Euclidean distances of each point on a Bezier spline to its closest point on an edge extracted from the pattern. The objective is similar to (4.6), except that we have only one image, optimize only for a 2D similarity transformation, and of course—for the sake of evaluating the quality of the model—keep the Bezier spline coefficients fixed.

Figures 5.4(a) and 5.4(b) show the Bezier splines registered inside the original image of the background pattern before and after curve-based global non-linear refinement, respectively. The left-top part of the splines is missing due to some occlusions. The result shows a clear improvement of the background model, although some of the splines fail to converge. This, however, does not compromise the quality of our camera poses, as we use a robust Huber norm that is able to handle a certain number of outlier residuals. The improvement can also be put into numbers, as we observe an error drop from 3.107844e+03 to 2.073013e+03 throughout the curve-based non-linear refinement.

5.3 Discussion

There are many avenues to use this result for further applications. First, we intend to improve several bottlenecks of our dense reconstruction framework and embed it into a real smart-phone application. One important point that would have to be addressed is the compensation of rolling shutter distortions. We furthermore intend to extend the mapping back-end to a hybrid approach that uses a 3D surface representation for the foreground object, and a curve-based representation for the background. The 3D surface will notably be used to enforce joint photometric and silhouette consistency, thus enabling the mapping of cavities with respect to the visual hull, as well as implicit model texturing.

Chapter 6

Conclusion & Future work

This thesis explores the idea of implementing a SLAM system with splines. The idea is inspired by the fact that a tracker based on straight lines typically outperforms those using sparse points in terms of accuracy. Although lines exist widely in a man-made environment, a more flexible feature is preferred to deal with more general scenes. For example, the boundary of modern furniture can be arbitrary curves. To approximate the model with piece-wise lines is neither theoretically elegant nor experimentally accurate. Therefore, this work explores a free-form representation of the environment, more specifically speaking, the method to localize a camera and reconstruct the environment with spline features. Bezier splines are introduced to model the environment, based on which a bundle adjustment pipeline is implemented and achieves more accurate structure from motion with a monocular camera. As another achievement, the accurate pose estimation enables us to perform a space carving method that gives good 3D reconstruction results. In conclusion, this work provides a fully automatic pipeline of structure from motion, which incorporates a general curve model to a global bundle adjustment pipeline. It is demonstrated via experiments that the proposed method outperforms sparse feature based methods, and is able to give visually expressive, semi-dense models.

6.1 Critical View

While this thesis realizes a working system that successfully outperforms a popular point-based method, using a spline based representation still holds a number of disadvantages. Since the power of geometric constraints between frames is highly dependent on the quality of correspondences, it is hard to achieve a level of reliability due to the complexity of curve matching between images. Furthermore, we primarily work in indoor environments, where the majority of the edges are in fact straight lines, and the Bezier-spline based parametrization is in fact too high-order for this type of data, and possibly leads to inferior performance compared to lines. More importantly, the remaining non-straight edges often are so-called apparent contours, which are curves in the image that are created by the planar projection of a smooth, arbitrarily shaped object. In other words, these features do not correspond to real, stable landmarks in 3D that would be viewpoint invariant. Since most stable features are likely to be straight lines, and all other curves are not stable, the Bezier-spline based parametrization simply is not a very effective choice.

Our conclusion is that, unless there is a special case that really requires this kind of parametrization, it is probably preferable to have either a point-based method (if computational efficiency is a concern), or otherwise a photometric, fully dense method (that will be able to handle arbitrary geometries). The desire of a semi-dense solution that finds a very good trade-off between accuracy and computational load has not been realised.

6.2 Future work

The following list some further research and technological directions that can be improve for my works:

- 3D reconstruction is performed in an offline manner. An interesting further work would be implementing an application on a mobile-phone platform which can perform 3D reconstruction online.
- For 3D reconstruction, space carving method is used to reconstruct an object. It is expected to see better reconstruction results by trying with some complex surface reconstruction algorithm.
- A convex surface would lead to a degenerate case for all edge based methods, where the edges of a convex surface are variant with change of viewing perspective. An interesting future work would be to develop novel strategies that can deal with this degenerate case.
- Currently, efficiency is not taken as a core consideration in this work. It is expected to achieve real-time performance by accelerating the computation via using a GPU. (Add some details on which parts can be accelerated by using a GPU).

Bibliography

[1] Handa, A., Whelan, T., McDonald, J., Davison, A.: A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). (2014)

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

[3] Pumarola, A., Vakhitov, A., Agudo, A., Sanfeliu, A., Moreno-Noguer, F.: PL-SLAM: Real-time monocular visual SLAM with points and lines. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Singapore (2017)

[4] Zuo, X., Xie, X., Liu, Y., Huang, G.: Robust visual SLAM with point and line features. Arxiv Computing Research Repository **abs/1711.08654** (2017)

[5] Lu, Y., Song, D.: Robust RGB-D odometry using point and line features. In: Proceedings of the International Conference on Computer Vision (ICCV), Santiago, Chile (2015)

[6] Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-scale direct monocular SLAM. In: European Conference on Computer Vision, Springer (2014) 834–849

[7] Kuse, M.P., Shen, S.: Robust camera motion estimation using direct edge alignment and sub-gradient method. In: IEEE International Conference on Robotics and Automation (ICRA-2016), Stockholm, Sweden. (2016)

[8] Delaunoy, A., Pollefeys, M.: Photometric Bundle Adjustment for Dense Multi-View 3D Modeling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2014)

[9] Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: Dtam: Dense tracking and mapping in real-time. In: 2011 international conference on computer vision, IEEE (2011) 2320–2327

[10] Nistér, D.: An efficient solution to the five-point relative pose problem. IEEE transactions on pattern analysis and machine intelligence **26**(6) (2004) 756–770

[11] Hartley, R.I.: In defense of the eight-point algorithm. IEEE Transactions on pattern analysis and machine intelligence **19**(6) (1997) 580–593

[12] Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: Computer Vision (ICCV), 2011 IEEE international conference on, IEEE (2011) 2564–2571

[13] Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: European conference on computer vision, Springer (2006) 404–417

[14] Leutenegger, S., Chli, M., Siegwart, R.Y.: Brisk: Binary robust invariant scalable keypoints. In: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE (2011) 2548–2555

[15] , R., Zisserman, A.: Multiple View Geometry in Computer Vision. Second edn. Cambridge University Press, New York, NY, USA (2004)

[16] Mur-Artal, R., Montiel, J., Tardós, J.D.: ORB-SLAM: a versatile and accurate monocular slam system. IEEE Transactions on Robotics **31**(5) (2015) 1147–1163

[17] Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: exploring photo collections in 3d. In: ACM transactions on graphics (TOG). Volume 25., ACM (2006) 835–846

[18] Snavely, N., Seitz, S.M., Szeliski, R.: Modeling the world from internet photo collections. International journal of computer vision **80**(2) (2008) 189–210

[19] Porrill, J., Pollard, S.: Curve matching and stereo calibration. Image and Vision Computing (IVC) **9**(1) (1991) 45–50

[20] Feldmar, J., Betting, F., Ayache, N.: 3D-2D projective registration of free-form curves and surfaces. In: Proceedings of the International Conference on Computer Vision (ICCV). (1995)

[21] Kaminski, J.Y., Shashua, A.: Multiple view geometry of general algebraic curves. International Journal of Computer Vision (IJCV) **56**(3) (2004) 195–219

[22] Faugeras, O., Mourrain, B.: On the geometry and algebra of the point and line correspondences between n images. In: Proceedings of the International Conference on Computer Vision (ICCV). (1995)

[23] Kahl, F., Heyden, A.: Using conic correspondences in two images to estimate the epipolar geometry. In: Proceedings of the International Conference on Computer Vision (ICCV). (1998)

[24] Agarwal, S., Snavely, N., Simon, I., Seitz, S., Szeliski, R.: Building rome in a day. In: Proceedings of the International Conference on Computer Vision (ICCV). (2009) 72 –79

[25] Schindler, G., Krishnamurthy, P., Dellaert, F.: Line-based structure from motion for urban environments. In: Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT), Chapel Hill, USA (2006) 846–853

[26] Kaess, M., Zboinski, R., Dellaert, F.: MCMC-based multi-view reconstruction of piecewise smooth subdivision curves with a variable number of control points. In: Proceedings of the European Conference on Computer Vision (ECCV). (2004)

[27] Kahl, F., August, J.: Multiview reconstruction of space curves. In: Proceedings of the International Conference on Computer Vision (ICCV). (2003)

[28] Xiao, Y.J., Li, Y.: Optimized stereo reconstruction of free-form space curves based on a nonuniform rational B-spline model. Journal of the Optical Society of America **22**(9) (2005) 1746–1762

[29] Teney, D., Piater, J.: Sampling-based multiview reconstruction without correspondences for 3D edges. In: Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT). (2012)

[30] Berthilsson, R., Astrom, K., Heyden, A.: Reconstruction of general curves, using factorization and bundle adjustment. International Journal of Computer Vision (IJCV) **41**(3) (2001) 171–182

[31] Nurutdinova, I., Fitzgibbon, A.: Towards pointless structure from motion: 3D reconstruction and camera parameters from general 3d curves. In: Proceedings of the International Conference on Computer Vision (ICCV). (2015)

[32] Fabbri, R., Kimia, B.: 3D curve sketch: Flexible curve-based stereo reconstruction and calibration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2010)

[33] Cashman, T.J., Fitzgibbon, A.W.: What shape are dolphins? building 3d morphable models from 2d images. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) **35**(1) (2013) 232–244

[34] Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Mixed and Augmented Reality, 6th IEEE and ACM International Symposium on. (2007) 225–234

[35] Smith, P., Reid, I.D., Davison, A.J.: Real-time monocular slam with straight lines. (2006)

[36] Engel, J., Sturm, J., Cremers, D.: Semi-dense visual odometry for a monocular camera. In: Proceedings of the IEEE International Conference on Computer Vision. (2013) 1449–1456

[37] Tarrio, J.J., Pedre, S.: Realtime edge-based visual odometry for a monocular camera. In: IEEE International Conference on Computer Vision (ICCV). (2015) 702–710

[38] Li, H., Yao, J., Bazin, J.C., Lu, X., Xing, Y., Liu, K.: A monocular slam system leveraging structural regularity in manhattan world. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2018) 2518–2525

[39] Li, H., Yao, J., Lu, X., Wu, J.: Combining points and lines for camera pose estimation and optimization in monocular visual odometry. In: Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, IEEE (2017) 1289–1296

[40] Canny, J.: A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence (6) (1986) 679–698

[41] Hartley, R.: Projective reconstruction from line correspondences. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA (1994) 903–907

[42] Mei, C., Malis, E.: Fast central catadioptric line extraction, estimation, tracking and structure from motion. In: Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS), Beijing, China (2006)

[43] Cayley, A.: About the algebraic structure of the orthogonal group and the other classical groups in a field of characteristic zero or a prime characteristic. In: Reine Angewandte Mathematik. (1846)

[44] Gomez-Ojeda, R., Moreno, F., Scaramuzza, D., Jiménez, J.G.: PL-SLAM: a stereo SLAM system through the combination of points and line segments. Arxiv Computing Research Repository **abs/1705.09479** (2017)

[45] Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: Computer vision and pattern recognition, 2006 IEEE computer society conference on. Volume 2., Ieee (2006) 2161–2168

[46] Whelan, T., Leutenegger, S., Salas-Moreno, R., Glocker, B., Davison, A.: Elasticfusion: Dense slam without a pose graph, Robotics: Science and Systems (2015)

[47] Kerl, C., Sturm, J., Cremers, D.: Dense visual slam for RGB-D cameras. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, IEEE (2013) 2100–2106

[48] Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., Burgard, W.: An evaluation of the RGB-D slam system. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), St Paul, USA (2012)

[49] Stückler, J., Behnke, S.: Multi-resolution surfel maps for efficient dense 3d modeling and tracking. Journal of Visual Communication and Image Representation **25**(1) (2014) 137–147

[50] Whelan, T., Kaess, M., Fallon, M., Johannsson, H., Leonard, J., McDonald, J.: Kintinuous: Spatially extended kinectfusion. In: 3rd RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras, Sydney, Australia (2012)

[51] Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS). (2012) 573–580

[52] Malis, E., Vargas, M.: Deeper understanding of the homography decomposition for vision-based control. PhD thesis, INRIA (2007)

[53] Kneip, L., Scaramuzza, D., Siegwart, R.: A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Colorado Springs, USA (2011)

[54] Kneip, L., Furgale, P.: OpenGV: A Unified and Generalized Approach to Real-Time Calibrated Geometric Vision. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Hongkong (2014)

[55] Zhou, Y., Kneip, L., Li, H.: Semi-dense visual odometry for RGB-D cameras using approximate nearest neighbour fields. Accepted by ICRA 2017 **abs/1702.02512** (2017)

[56] Agarwal, S., Mierle, K., Others: Ceres solver. `http://ceres-solver.org`

[57] Zhou, Y., Li, H., Kneip, L.: Canny-vo: Visual odometry with rgb-d cameras based on geometric 3-d–2-d edge alignment. IEEE Transactions on Robotics **35**(1) (2019) 184–199

[58] Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2012)

[59] Zhou, Y., Li, H., Kneip, L.: Canny-vo: Visual odometry with rgb-d cameras based on geometric 3-d–2-d edge alignment. IEEE Transactions on Robotics (2018)

[60] Rother, C., Kolmogorov, V., Blake, A.: Grabcut: Interactive foreground extraction using iterated graph cuts. In: ACM transactions on graphics (TOG). Volume 23., ACM (2004) 309–314