Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

SLAM Localization Using Stereo Camera

Lukáš Majer

Supervisor: Ing. Jan Chudoba
May 2019

Acknowledgements

I would like to thank my supervisor Ing. Jan Chudoba for his patience and guidance, as well as the whole Intelligent and Mobile Robotics Group for providing necessary equipment and support. Last, but not least I am grateful for all the support provided by my girlfriend.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, May 22, 2019

.....

signature

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 22. května 2019

.....

podpis autora práce

Abstract

This work deals with an application of simultaneous localization and mapping (SLAM) methods for a stereo camera for the purpose of a mobile robot or human localization. First, research of available method implementation is conducted and selected methods are evaluated with the comparison. Based on results from previous steps a system for a mobile robot localization is proposed and tested.

Keywords: RGB-D SLAM, stereo camera, localization

Supervisor: Ing. Jan Chudoba
CIIRC B-323,
Jugoslávských partyzánů 3,
16000 Praha 6

Abstrakt

Tato práce se zabývá aplikací současného mapování a localizace (SLAM) pro stereo kameru s cílem lokalizace mobilního robotu nebo člověka. Nejprve je provedena rešerše dostupných metod, ze kterých je posléze několik otestováno a porovnáno. Na základě těchto výsledků je následně navržen a otestován lokalizační systém pro mobilního robota.

Klíčová slova: RGB-D SLAM, stereo kamera, lokalizace

Překlad názvu: Lokalizace SLAM pro stereokameru

Contents

Figures

Tables

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Majer Lukáš** Personal ID number: **457193**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

Branch of study: **Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

SLAM Localization Using Stereo Camera

Bachelor's thesis title in Czech:

Lokalizace SLAM pro stereokameru

Guidelines:

The topic of the thesis is an application of simultaneous localization and mapping (SLAM) methods for stereo camera for the purpose of mobile robot or human localization. Do a research of 3D SLAM methods able to use a depth map from the stereo camera as an input. Select a convenient available method implementation (or more methods allowing comparison) and apply data from the camera provided by the thesis supervisor. Suppose a general type of the camera motion. Perform the practical experiments and make an evaluation of an achievable localization precision and limits of the method functionality.

Bibliography / sources:

[1] Šonka, M., Hlaváč, V.: Počítačové vidění. Grada, Praha 1992
[2] Besl, Paul J.; N.D. McKay (1992). "A Method for Registration of 3-D Shapes". IEEE Trans. on Pattern
[3] R. Mur-Artal, J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular stereo and RGB-D cameras", IEEE Trans. Robot., vol. 33, no. 5, pp. 1255-1262, Oct. 2017.

Name and workplace of bachelor's thesis supervisor:

Ing. Jan Chudoba, Intelligent and Mobile Robotics, CIIRC

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **09.01.2019** Deadline for bachelor thesis submission: **24.05.2019**

Assignment valid until: **30.09.2020**

Ing. Jan Chudoba	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Ing. Pavel Ripka, CSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____ Date of assignment receipt

_____ Student's signature

Chapter 1

Introduction

1.1 Context

SLAM, which is an abbreviation for **S**imultaneous **L**ocalization **A**nd **M**apping, has been the main research interest in robotics for more than thirty years and to this day is considered one of the most fundamental problems in mobile robotics. As the name suggests, the goal is to build a map of the unknown environment while also keeping track of a position in said map. In other words, an input to SLAM algorithms are sensor measurements and optionally control inputs, the output of SLAM is a location inside the map and optionally also the map itself.

This work deals with a subset of SLAM problems, called **RGB-D** SLAM, where the only or one of the sensors is a stereo camera. Unlike a regular camera, which provides a two-dimensional representation of three-dimensional scene without depth information, the stereo camera is able to capture a three-dimensional image, sometimes called scans. This approach to SLAM has been on the rise in recent years because it simulates human binocular vision and the way humans use eyes to navigate.

The goal of this thesis is to find one or more suitable methods for RGB-D SLAM and conduct practical experiments. In order to do so it is also needed to get familiar with stereo cameras in general and understand how the 3D scene is captured.

1.2 Structure

My thesis can be divided into three main parts. The first part provides a brief overview of mathematical constructs used in this work while also getting the reader familiar with a basic camera model. This knowledge is then expanded to build a simple yet accurate model of a stereo camera. Also, hardware and software parameters of the device used for practical experiments, ZED Stereo Camera, are discussed.

Next part deals with RGB-D SLAM, first it defines the problem itself while also stating the most prominent challenges. It gives a theoretical overview of used algorithms, while also presenting details about respective implementation. Other approaches to the problem are also discussed and referenced.

Last part is the center of this work. It describes practical experiments and gives an overview of obtained results. Those results are then employed to a more practical problem of SLAM with a mobile robot.

Chapter 2

Mathematical apparatus

In this chapter, I would like to give a short overview of the mathematics used throughout the thesis. Also, important conventions are set here, which are used and respected throughout the whole work.

First, I outline ways of representing rotation in $\mathbf{R}^3$ space and discuss metrics of rotation, then I describe homogeneous coordinates and how to use this mathematical construct to transform between different coordinate systems. Also, a topic of determining the difference between two rotations (metric) is discussed.

Let me first make a few remarks. Throughout the thesis, unless stated otherwise, I will use a right-handed Cartesian coordinate system. The default unit of distance will be meter and radians will be used for angular distance.

2.1 Rotation matrix

Perhaps the most straightforward way to represent the rotation of an object is to use a rotational matrix. Rotation matrix is 3x3 matrix in general form as follows

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \tag{2.1}$$

One way to construct such a matrix is to use a formula, which represents rotation around one of the principal axes as defined in [2]. Rotation matrix representing rotation of angle γ around z axis

$$\mathbf{R}_z(\gamma) = \begin{bmatrix} cos(\gamma) & -sin(\gamma) & 0 \\ sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

then rotation matrix representing a rotation of the angle β around new y axis

$$\mathbf{R}_y(\beta) = \begin{bmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{bmatrix} \tag{2.3}$$

and rotation matrix representing a rotation of the angle α around new x axis

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix}. \tag{2.4}$$

To represent full rotation, which consists of 3 DOF, three consecutive rotations must be used. This is achieved by simple matrix multiplication

$$\mathbf{R} = \mathbf{R}_z(\gamma)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha). \tag{2.5}$$

2.2 Euler angles

Another way to represent rotation is to use Euler angles. Euler angles are a set of three numbers, each representing rotation around the previously defined axis.

In my work I use yaw (γ), pitch (β) and roll (α) convention (in order), this means rotation around z axis, new y axis and new x axis, as can be seen in figure 2.1.

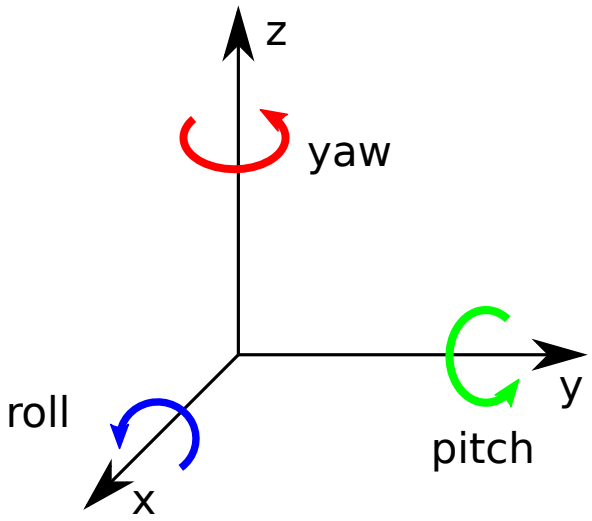


Figure 2.1: Definition of Euler angles.

Euler angles are not redundant, are relatively easily interpreted, but suffer from gimbal lock problem. Also, there is no direct formula for rotating a vector using Euler angles. For that we can use, among others, corresponding combination of equations 2.4,2.3 and 2.2 or convert them to quaternions.

2.3 Quaternion

Quaternion, as defined in [2], has following equation

$$\mathbf{q} = w + xi + yj + zk \tag{2.6}$$

where w is real number and x, y, z represent imaginary parts. In this work I will use unit quaternion, which satisfies a condition $|\mathbf{q}| = 1$. To rotate vector $\mathbf{r}$ by quaternion $\mathbf{q}$ the following formula can be used

$$\mathbf{r}' = \mathbf{q}\mathbf{r}\mathbf{q}^* \tag{2.7}$$

where $\mathbf{q}^*$ stands for a complex conjugate defined as

$$\mathbf{q}^* = w - xi - yj - zk. \tag{2.8}$$

Moreover, a rotation matrix as defined in 2.1 can be constructed directly from quaternion using a following formula

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} 1-2(y^2+z^2) & 2(xy-zw) & 2(xz+yw) \\ 2(xy+zw) & 1-2(x^2+z^2) & 2(yz-xw) \\ 2(xz+yw) & 2(yz+xw) & 1-2(x^2+y^2) \end{bmatrix}. \tag{2.9}$$

Using quaternion it is possible and also convenient, according to [3], to represent the difference between two rotations. This mapping is also called metric and can be obtained using following equation

$$\alpha = 2\arccos(|\mathbf{q}_1 \cdot \mathbf{q}_2|), \tag{2.10}$$

which will be my default metric to evaluate error between two rotations.

2.4 Homogenous coordinates

Most of my thesis deals with transitions between different coordinate systems, to achieve this task I use homogeneous coordinates. If x,y and z are regular Cartesian coordinates in $\mathbf{R}^3$ space, then we can define homogeneous coordinates as follows

$$\mathbf{r} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \tag{2.11}$$

where w stands for scale. To transform between non-homogeneous and homogeneous coordinates we can use following relations

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.12}$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow \begin{bmatrix} x/w \\ y/w \\ z/w \end{bmatrix}. \tag{2.13}$$

One advantage of using homogeneous coordinates is possibility of constructing a transformation matrix, representing rotation and translation, defined as follows

$$\mathbf{A} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.14}$$

Transformation matrix $\mathbf{A}$ represents translation and orientation of an object and is sometimes called *pose*. To transform a vector in homogeneous, coordinates we multiply the vector by the transformation matrix on the right side

$$\mathbf{r}' = \mathbf{A}\mathbf{r}. \tag{2.15}$$

In some cases it is required to apply multiple transformations in a row, using homogeneous coordinates this can be achieved by matrix multiplication of consecutive transformations between coordinate systems *0,1,...,n* as

$$\mathbf{A}_n^0 = \mathbf{A}_1^0 \mathbf{A}_2^1 ... \mathbf{A}_n^{n-1}. \tag{2.16}$$

Throughout the thesis $\mathbf{A}(\mathbf{q}, \mathbf{t})$ notation is used, which represents transformation matrix as defined in 2.14, but constructed using quaternion $\mathbf{q}$ and translation vector $\mathbf{t}$ as

$$\mathbf{A} = \begin{bmatrix} \mathbf{R}(\mathbf{q}) & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.17}$$

Chapter 3

Camera model

The goal of this chapter is to propose a suitable mathematical model of passive stereo camera and outline ways to use said model for 3D reconstruction. First, a standard pinhole camera model is introduced, which is later expanded by accounting for various distortions. Then I highlight a way to obtain depth from a pair of regular cameras.

3.1 Pinhole camera model

3.1.1 Projection equation

Figure 3.1 shows four important coordinate systems with all four of them using homogeneous coordinates. I will adopt notation used in [4]. The first coordinate system with origin O_w represents the world coordinate system. Then there is O_c, which stands for camera coordinate system, which is positioned in a way that Z_c axis is aligned with camera optical center with Y_c axis facing down. The third coordinate system represents a normalized image plane with origin o_i and last coordinate system, which represents an image influenced by intrinsic camera parameters, with origin o_a also located on the image plane.

Transformation from O_w to O_c, also called extrinsic camera matrix, can be expressed using homogeneous coordinates as

$$\mathbf{E} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.1}$$

Matrix $\mathbf{E}$ is transformation matrix as defined in equation 2.14,thus representing rotation and translation. Now matrix representing projection from camera coordinate frame O_c to normalized image plane o_i can be written as

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.2}$$

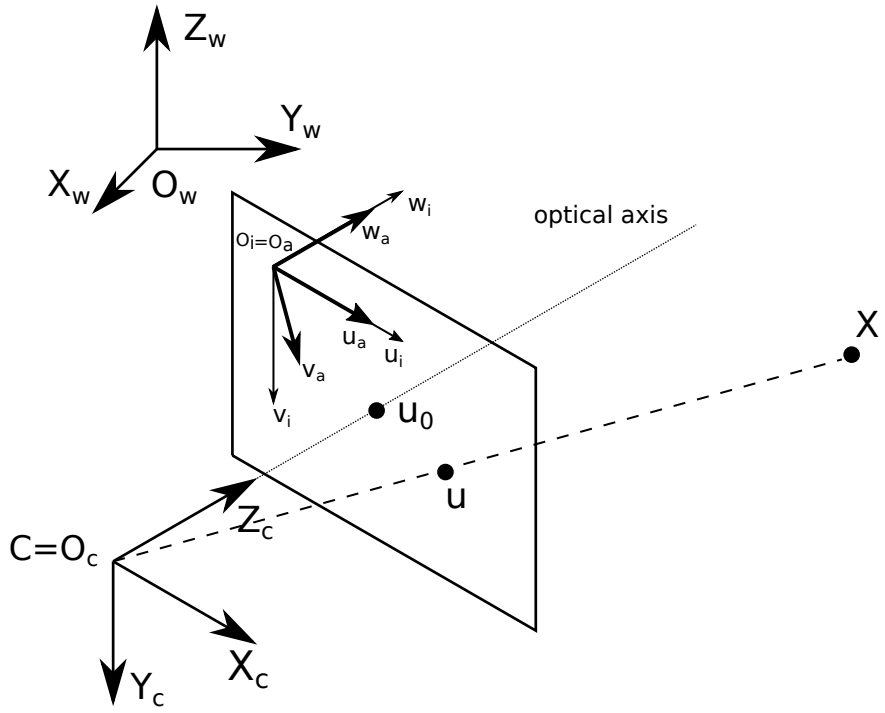


Figure 3.1: Coordinate system definition for pinhole camera model.

Matrix representing camera intrinsic parameters and transformation to coordinate system o_a is defined as

$$\mathbf{K} = \begin{bmatrix} f_x & s & -u_0 \\ 0 & f_y & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

where f_x and f_y are focal lengths in respective axis in meters, u_0 and v_0 are coordinates of the principal point $\mathbf{u}_0$ also in meters and s represents shear parameter, which is sometimes omitted. Equations 3.1,3.2 and 3.3 together form a camera matrix defined as

$$\mathbf{M} = \mathbf{KHE}. \tag{3.4}$$

It is important to note that in most of the software used in this thesis, I represent the focal length and principal point in pixels. To convert from distance unit to pixel unit, a simple formula can be used

$$f_p = m f_d. \tag{3.5}$$

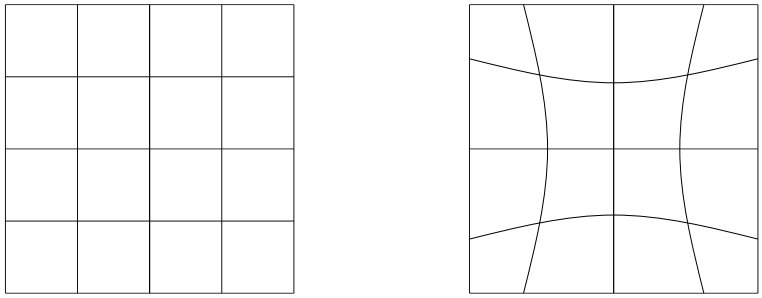
Value m has units [pixel/meter] and represents pixel density, value f_d represents focal length/principal point in [meters] and f_p represents the same, but in [pixel] units.

The complete process of projection from the 3D world coordinate system to 2D image coordinate system is then defined as follows in equation

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x & s & -u_0 \\ 0 & f_y & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}. \tag{3.6}$$

3.1.2 Distortion model

Radial distortion



(a) : undistorted image **(b) :** distorted image

Figure 3.2: Radial distortion.

The ideal pinhole camera model does not account for lens distortion, which especially in cheap cameras, can be quite substantial. It is mostly visible when taking pictures with visible straight lines. Lines passing image center will appear undistorted, but lines farther away from the center will appear curved as seen in figure 3.2. One obvious solution to solve such a problem is to use lens with minimal optical deficiency. That might not always be an option considering price and desired accuracy. One way to address this issue is to use a software correction. As proposed by [5], radial distortion can be represented as a polynomial of even degree, with second degree term accounting for most of the distortion. With this knowledge, radial distortion can be modeled as

$$x_d = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.7}$$

$$y_d = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.8}$$

$$r^2 = x^2 + y^2 \tag{3.9}$$

where x, y stand for undistorted pixel coordinates x_d, y_d for distorted pixel coordinates and parameters k_1, k_2, k_3 are called radial distortion coefficients. These parameters are either provided by lens manufacturer or have to be estimated.

Tangential distortion

Tangential distortion occurs when the lens and a camera sensor are not in parallel. Result of this distortion is an image tilt around one or both of the image axis. Although this kind of defect is usually not as substantial as radial distortion, it can be corrected by the following equations

$$x_d = x + (2p_1xy + p_2(r^2 + 2x^2)) \tag{3.10}$$

$$y_d = y + (p_1(r^2 + 2y^2) + 2p_2xy) \tag{3.11}$$

where x, y stand for undistorted pixel coordinates x_d, y_d for distorted pixel coordinates and parameters p_1 and p_2 are tangential parameters and r is defined by equation 3.9.

3.2 Stereo camera rig

The motivation behind this section is to devise a way to estimate depth from two images using two cameras in a special configuration. Depth estimation is typically done in three steps:

1. Establishing constraints based on known information about both cameras, e.g. extrinsic and intrinsic camera parameters.
2. Finding correspondences between pixels in the left and the right image.
3. Using found correspondences to estimate depth.

The task of reconstructing 3D scene is non-trivial and in some cases, also impossible as discussed later on.

3.2.1 Epipolar geometry

In order to make finding correspondences easier, it is desirable to apply as many constraints as possible. One of those constraints is an epipolar constraint, as mentioned in [6]. In figure 3.3 a basic stereo setup can be seen.

Points e and e' are called epipoles. It is obvious that epipole e' is projection of the optical center C in the image plane of the left camera and vice versa. From vectors $\mathbf{u}$ and $\mathbf{e}$ a line can be constructed representing all possible locations of $\mathbf{u}'$ in the left image plane. Thus epipolar constraint restricts search for pixel correspondences from 2D space (whole image) to 1D space (line).

Searching for correspondences can be further simplified with rectified camera configuration. To define such configuration of cameras k and k' with their respective optical centers $\mathbf{C}$ and $\mathbf{C}'$ and image planes p and p'. Then we call k and k' a rectified configuration iff p_1 and p_2 coincide and line $\mathbf{CC}'$ is parallel to p and p'.

In rectified camera configuration epipoles e and $e^{'}$ lie in infinity. This makes lines **ue** and $\mathbf{u}^{'}\mathbf{e}^{'}$ horizontal and thus matching correspondences becomes more convenient.

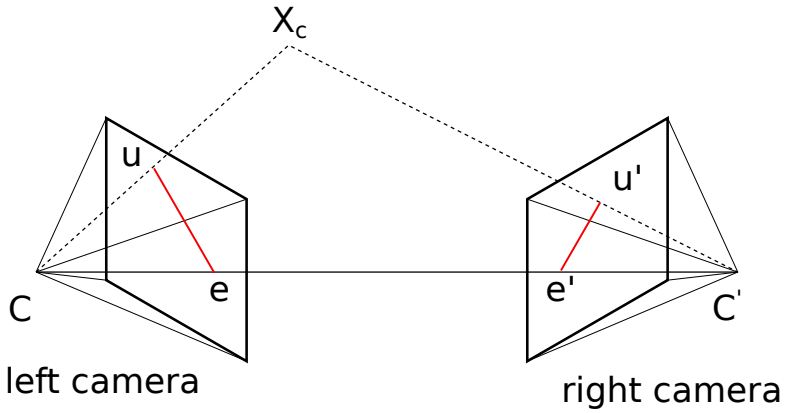


Figure 3.3: Epipolar geometry.

3.2.2 Finding correspondences

The most difficult part of a 3D scene reconstruction process is to find pixel correspondences between a pair of images.This can either be done manually or more preferably by correspondence matching algorithm. Many such algorithms were proposed over the years, and more approaches are being researched even now. Two most prominent categories of algorithms are:

- Correlation-based block matching.
- Feature-based correspondence matching.

Correlation-based block matching algorithms use a premise that corresponding pixels have similar intensity levels. Although intensity alone is not enough to find correspondences, neighbouring pixels can also be considered and thus forming a block of pixels. This block is then searched for in the other image based on a metric.

Feature-based correspondence matching algorithms usually work in two steps. First "features" features are extracted from images and then matched between images. Some examples are SURF features [7], SIFT [8] and ORB [9]. An in-depth explanation of above features is beyond a scope of this thesis. However, general idea common to these algorithms is to use distinctive points in an image to generate a feature descriptors. This description can then be used to identify and register key points. It is desirable for these descriptor generators to offer some "resistance" to changes in illumination, scale or noise.

3.2.3 Estimating depth

Based on the knowledge presented in previous chapters I am now able to estimate depth from stereo camera rig, but first a few assumptions have to be made:

- Intrinsic parameters of both cameras are known and are presumed to be the same.
- Cameras are in rectified configuration.
- Baseline, the distance between optical centers of both cameras, is known.

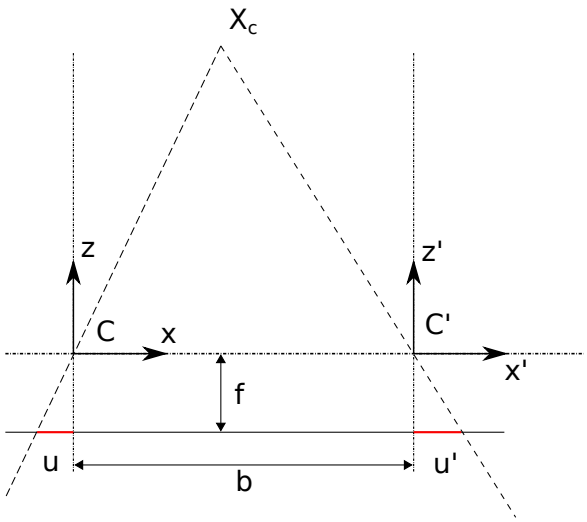


Figure 3.4: Estimating depth from rectified stereo setup.

First, a term called disparity should be defined. It is the difference in horizontal coordinates of corresponding image points, therefore can be represented by the following equation

$$d = u^{'} - u. \tag{3.12}$$

Looking at figure 3.4 there are two coordinate systems with origins $\mathbf{C}$ and $\mathbf{C}^{'}$ representing right and left camera. All points will be in the coordinate system of the left camera with origin $\mathbf{C}$. The right camera coordinate system is for illustration. To reconstruct point in the 3D scene, I start with the use of triangle similarity

$$\frac{u}{f} = -\frac{x}{z} \tag{3.13}$$

$$\frac{u^{'}}{f} = -\frac{b-x}{z}. \tag{3.14}$$

Combining equation 3.13, 3.14 and 3.12 while eliminating x, depth information z can be recovered as

$$z = \frac{bf}{d}. \tag{3.15}$$

Similarly from equations 3.13, 3.14, 3.12 and eliminating z, it is possible to calculate x coordinate as

$$x = \frac{-b(u + u^{'})}{2d}. \tag{3.16}$$

Using assumption about rectified stereo camera configuration as stated before:

$$v = v^{'} \tag{3.17}$$

To obtain the last coordinate y, once again similar triangles can be used as

$$\frac{v}{f} = \frac{y}{z}. \tag{3.18}$$

Plugging equation 3.15 into 3.18 last 3D coordinate y can be obtained as follows

$$y = \frac{bv}{d}. \tag{3.19}$$

3.2.4 Error modeling

As suggested in [10] depth error can be obtained by differentiating equation 3.15 with respect to disparity as

$$\frac{\delta z}{\delta d} = -\frac{bf}{d^2}. \tag{3.20}$$

Substituting eq. 3.15 to 3.20 and rearranging the equation following relation can be obtained

$$|\delta z| = \frac{z^2 \delta d}{bf} \tag{3.21}$$

where f is focal length, b denotes baseline and δd stands for disparity error. It is important to note the interpretation, error in depth estimation grows quadratic-ally with distance.

Chapter 4

ZED Stereo camera

This chapter contains a description of used stereo camera sensor used in my thesis. ZED Stereo Camera was provided by my supervisor Ing. Jan Chudoba. First important hardware specifications are outlined, and then I give a brief overview of ZED native SDK.

4.1 Hardware description

ZED Stereo Camera is a passive stereo device developed by Stereolabs Inc. and as such knowledge about under the hood hardware is limited to the public. For illustration, I include a picture of ZED Stereo Camera along with product dimensions in figure 4.1.

(a) : photo of ZED camera [11] **(b) :** schematics with dimensions

Figure 4.1: ZED Stereo Camera illustration.

In table 4.1 general hardware specifications are presented as listed in [11]. For experiments, I decided the lowest resolution because the goal of this thesis is to design SLAM solution capable of real-time operation and for this resolution, ZED Stereo Camera provides reasonable 100 frames per second.

Each camera comes with a unique factory calibration file, which is downloaded automatically. Camera calibration parameters unique for ZED Stereo Camera used in this thesis are in table 4.2. The notation is the same as in chapter 3. It is important to note that calibration file from Stereolabs Inc. completely omits tangential distortion and shear parameter s and uses only the first two terms of radial distortion. This approximation should not have a huge influence on accuracy, because as mentioned earlier, their magnitude in modern digital cameras is minimal.

Parameter	Value
Resolution (side by side)	4416 x 1242 pixels with 15 FPS
with respective max. FPS	3840 x 1080 pixels with 30 FPS
	2560 x 720 pixels with 60 FPS
	1344 x 376 pixels with 100 FPS
Baseline	200 mm
Depth range	0.5 - 20 m

Table 4.1: ZED Stereo Camera hardware specifications.

Parameter	Left camera	Right camera
f_x [pixels]	350.034	349.87
f_y [pixels]	350.034	349.87
u_0 [pixels]	358.473	346.888
v_0 [pixels]	206.673	201.366
k_1 [-]	-0.173	-0.170
k_2 [-]	0.027	0.026
baseline [mm]	119.907	

Table 4.2: ZED Stereo Camera calibration parameters for WVGA resolution.

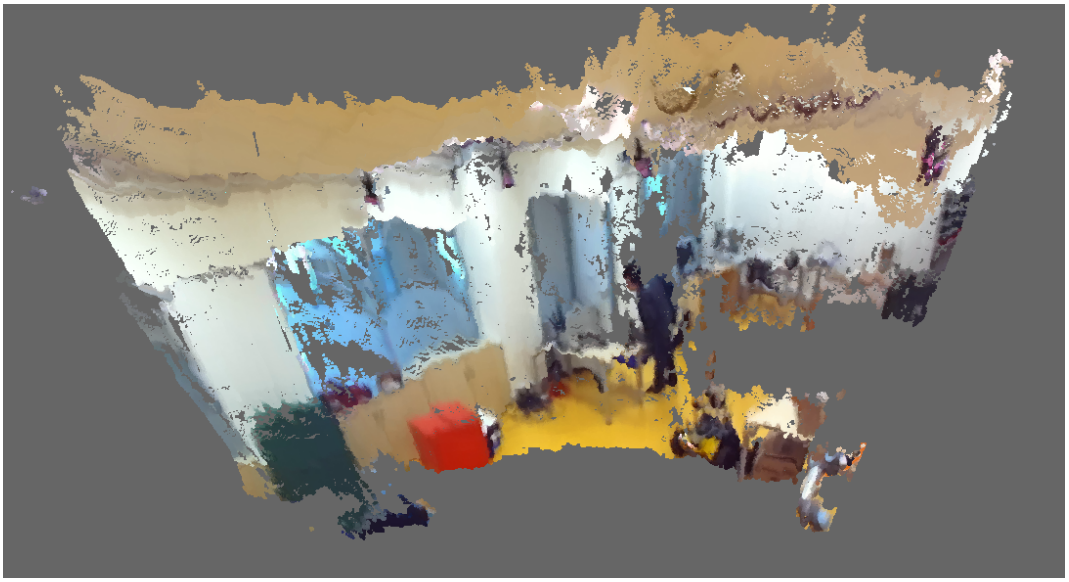
4.2 Software interface

The exact process how camera estimates depth was not at the time of writing of this thesis has not been made public, however due hardware nature of this sensor it is safe to say it follows the process outlined in section 3.2 with some unknown modifications.

The camera itself is controlled by C++ ZED SDK with online documentation which can be found in [12]. Mentioned API provides means to capture regular images, depth maps, video and also incorporates its own SLAM method.

Typical usage of said SDK resolves around **Camera** class. First, **Camera** object is created which is then provided with **InitParameters** which defines resolution, FPS and also defines the coordinate system in which all data is received. Data can then be obtained using **grab()** method. Output of the stereo camera is 2D RGB image and corresponding depth map, which is then re-projected into a 3D colored point cloud. Result can be seen in figure 4.2.

(a) : Image of the scene.



(b) : Reconstructed scene as 3D colored point cloud.

Figure 4.2: Illustration of ZED Stereo Camera 3D reconstruction.

Chapter 5
RGB-D SLAM

This chapter should give a potential reader an overview of RGB-D SLAM problem, outline used solutions and present other approaches. Furthermore I discuss implementation details of SLAM algorithms.

5.1 Problem definition

RGB-D SLAM is a specific case of Simultaneous Localization And Mapping where a 3D colored point cloud serves as an input. The output is a camera pose, which is represented by a transformation matrix as defined in equation 2.14. To get a more formal definition, let me denote Z_i a 2-tuple of stereo camera measurement X_i and associated time t_i. RGB-D SLAM is then a mapping which assigns a pose $\mathbf{A}_i(\mathbf{q}, \mathbf{t})$, consisting of quaternion $\mathbf{q}$ and translation vector $\mathbf{t}$, to every Z_i for i = *1,2,...,N* , where N is a number of measurements.

The stereo camera measurement X_i is represented as a 3D colored point cloud. To define such a term let me first consider a set of vectors $P = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ where $\mathbf{x}_i \in \mathbf{R}^3$. The set P is called a 3D point cloud. Now consider a mapping c: $x_i \longrightarrow V$ where V is an RGB color space and $\mathbf{x}_i \in \mathbf{R}^3$. Then the 3D colored point cloud is a 2-tuple of $P = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ and $C = \{c(\mathbf{x}_1), c(\mathbf{x}_2), ..., c(\mathbf{x}_n)\}$.

The uniqueness of this problem comes from providing two kinds of information at the same time. It provides pure geometric information as a regular 3D point cloud, but also providing a corresponding 2D colored image aligned with the said point cloud. Therefore it is obvious that most of the approaches trying to solve RGB-D SLAM problem will either try pure geometric approach, image feature based approach or their combination [13].

There are many difficulties when solving RGB-D SLAM. The most prominent among them is time and space algorithmic complexity. Typical 3D colored point cloud needs seven numbers to represent a point, considering x, y, z components of position and R, G, B, alpha color channels. If a stereo camera with n x n resolution is considered, for each scan $7n^2$ numbers are generated. Although there are more compact representations of 3D scans,

curse of dimensionality can not be eliminated. This proves especially problematic if a long term navigation over a large distance is required.

Also dealing dealing with outliers, caused for example by improper 3D scene measurement, can prove to be quite problematic. Another common problem is caused by uniform environment. Imagine now a simple scene consisting of a perfectly flat wall. For a human observer, the scene is understandable, but for RGB-D SLAM algorithm there are no geometric nor visual features to navigate by.

Problem constraints

- The camera is able to move and rotate in all three axes. Thus every pose of the camera has 6 DOF.
- Considering the purpose of mobile robot and human localization, I will consider the maximum translation velocity of $\approx 1.5m/s$ and angular velocity of $\approx 25°/s$. Such speeds are more than sufficient for a human or a mobile robot, while also allowing me to neglect the effect of rolling shutter.
- As stated before uniform environments (empty halls) are a difficult task, for that reason, I will limit the experiments to places with visual landmarks (office, forest, etc).
- Because I am using a passive stereo camera, there must be enough illumination. Only sufficiently lit ($>$ 80 lux) environments (common classroom with lights on) will be considered, thus excluding poorly lit environments (dark rooms, nighttime, etc).
- Considering the range of ZED Stero Camera, it is required for robot or human to be within the sensor distance of the scene. In other words, I will consider a ground robot or a low flying UAV ($<$10 m of height).
- Algorithm should be able to work in real-time. However, it is difficult to give a general definition of real time, so for my purposes, I will consider 0.5 FPS as a boundary.

5.2 Other approaches

5.2.1 RTAB-Map

RTAB-Map as described in [14] is a graph-based SLAM approach. Unlike previously mentioned SLAM algorithms it has native support for odometry as a direct input. It relies on feature extractor GFTT [15] and BRIEF feature descriptor [16] for pose estimation. SLAM problem is represented as a graph where every node corresponds to a pose of the robot, and every edge marks a

constraint between them. This graph is then continuously optimized based on new measurements and loop detection.

5.2.2 Direct RGB-D SLAM

The approach developed in [17] is unique a solution to RGB-D SLAM, because instead of relying on feature extraction, it operates directly on pixel intensities. It uses an assumption that if 3D point $\mathbf{x}$ is a pixel $\mathbf{u}$ in the first image and pixel $\mathbf{u}^{'}$ in the second image then

$$I(\mathbf{u}) = I(\mathbf{u}^{'}) \tag{5.1}$$

where I is a pixel intensity. This assumption is also called photo-consistency. Direct RGB-D SLAM then tries to find an optimal transformation $\mathbf{A}$ by maximizing photo-consistency while at the same time minimizing the difference between the predicted and the actual depth measurement, thus obtaining camera motion.

5.3 Used solutions

In this section, I will provide a more in-depth overview of RGBD-SLAM algorithms used in this thesis. First, the general idea is presented and their respective implementations are discussed. In total, I have tested three RGB-D SLAM approaches. ICP algorithm was chosen as an example of shape based approach and ORB-2 SLAM represents feature extraction group of RGB-D SLAM algorithms. ZED SLAM is also tested as it is conveniently included in the ZED SDK.

5.3.1 ICP SLAM

Principle

Iterative closest point algorithm as proposed by [18] works generally in three steps. An input is a point set of a new unregistered scan $P = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$ $\mathbf{p}_i \in \mathbf{R}^3$ and $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ $\mathbf{x}_i \in \mathbf{R}^3$ which denotes our base scan to match P against. First step is to find corresponding point $\mathbf{x}_i$ in X for every $\mathbf{p}_i$ in P . In other words it is required for every $\mathbf{p}_i$ to find minimal distance from point set X. This process can be described by equation

$$d(\mathbf{p}, A) = min \quad d(\mathbf{p}, \mathbf{a}_i) \quad i \in \{1, 2, .., N\}. \tag{5.2}$$

With set correspondences, next step is to find a transformation $\mathbf{A}(\mathbf{q}, \mathbf{t})$, consisting of quaternion $\mathbf{q}$ and translation vector $\mathbf{t}$, for which the distances between corresponding points will be minimal. This can be done with mean square error function

$$e(\mathbf{q}, \mathbf{t}) = \frac{1}{N_p} \sum_{i=1}^{N_p} |\mathbf{x}_i - \mathbf{R}(\mathbf{q})\mathbf{p}_i - \mathbf{t}|^2 \tag{5.3}$$

where N_p is number of correspondences. Then optimal transformation between point sets can be acquired as

$$\mathbf{A}(\mathbf{q}, \mathbf{t}) = \arg\min_{\mathbf{q},\mathbf{t}} e(\mathbf{q}, \mathbf{t}). \tag{5.4}$$

Now to put this all together into a complete ICP algorithm:

1. Considering a new unregistered scan P and a base scan X compute $d(\mathbf{p}_i, X)\ \forall \mathbf{p}_i \in P$.
2. Minimize error function $e(\mathbf{q}, \mathbf{t})$ and retrieve optimal transformation $\mathbf{A}(\mathbf{q}, \mathbf{t})$.
3. Apply resulting transformation $\mathbf{A}(\mathbf{q}, \mathbf{t})$ on P.
4. Repeat steps 1-3 until terminating condition (required precision, number of iterations) is met.

Implementation

The implementation included in **MRPT** [19] has in total, five settings and can be described with pseudo-code as follows:

Algorithm 1 Iterative closest point

```
1: i ← 0
2: Pose(i) ← Pose(0)
3: while i < maxIterations  or  thresDist > thresDistMin do
4:     matchings ← matchPoints(m1, m2)
5:     Pose(i + 1) ← leastSquare(matchings))
6:     if |Pose(i + 1) − Pose(i)| < err then
7:         thresDist ← thresDist ∗ alpha
8:         thresAngl ← thresAngl ∗ alpha
9:     i ← i + 1
```

With following parameters:

1. ***maxIterations*** defines a maximum number of iterations before the algorithms terminates
2. ***alpha*** scaling constant for pose transformation correction
3. ***theresDist*** maximal translational distance for which points are considered to correspond with each other
4. ***theresAngl*** maximal angular distance for which points are considered to correspond with each other
5. ***theresDistMin*** distance for which algorithm terminates

My program then takes a new scan and tries to align it relatively to the previous scan. This is done in an endless loop and thus it keeps building the map from registered scans until terminated. Overview of the process is outlined in figure 5.1. This approach is implemented in *icp_slam*.

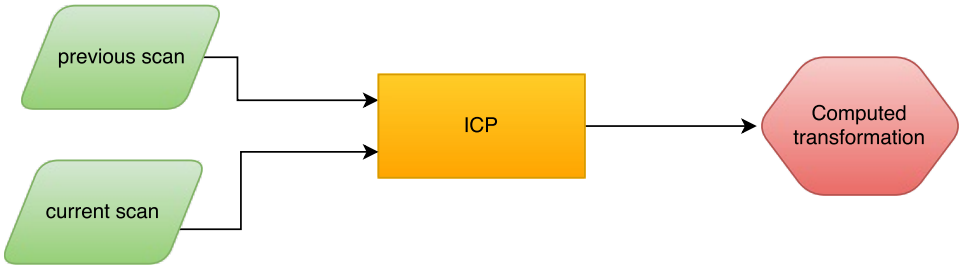


Figure 5.1: Flowchart of ICP scan alignment.

5.3.2 ORB2 SLAM

ORB features

ORB-2 SLAM uses **ORB** features [9], which as the full name **O**riented FAST and **R**otated **B**RIEF suggests have two main components FAST [20] key feature detector and BRIEF[16] feature descriptor.

FAST is a corner detection method. Considering pixel p which is to be tested for being FAST feature and Bresenham circle around the pixel p with a radius of three. Then all pixels inside the circle are sorted into three categories based on their intensity: darker than p, lighter than p and similar intensity to p. If 8 consecutive pixels are darker or lighter than p, then p is considered as FAST feature.

There is an obvious problem with scale variance, which is handled by constructing a pyramid with original image on base level. Then on every other level is a down sampled (zoomed) version of the image on previous level. FAST features are then extracted from the whole pyramid instead of just the original image. This approach offers partial scale in-variance at cost of increased computational time.

To determine corner orientation pixel intensity moment of $p+q$ order,

$$m_{pq} = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} u^p v^q I(u,v) \tag{5.5}$$

where u,v are pixel coordinates and $I(u,v)$ denotes pixel intensity, can be used. Orientation of corner is then easily extracted as

$$\theta = atan2(m_{01}, m_{10}). \tag{5.6}$$

BRIEF takes FAST features found by the FAST algorithm and converts it into a binary feature vector representing N bit binary string. To construct such a string BRIEF chooses N random pixel pairs p_x and p_y in the pixel neighbourhood and performs a binary test defined as

$$\tau(p_x, p_y) = \begin{cases} 1 & \mathrm{I}(p_x) < \mathrm{I}(p_y) \\ 0 & \mathrm{I}(p_x) \geq \mathrm{I}(p_y) \end{cases} \tag{5.7}$$

where $I(p)$ represents pixel intensity. The whole binary string is then iteratively constructed as

$$f(n) = \sum_{i=1}^{N} 2^{i-1} \tau(p_x, p_y). \tag{5.8}$$

The problem is that BRIEF descriptor can not deal with the rotation of key-points. ORB proposes a method to steer BRIEF according to the orientation of key-points.

Principle

Next SLAM algorithm is ORB2 SLAM as devised by [1]. Overview of the whole system can be seen in fig. 5.2.

ORB2 SLAM utilizes three parallel threads. First thread performs tracking

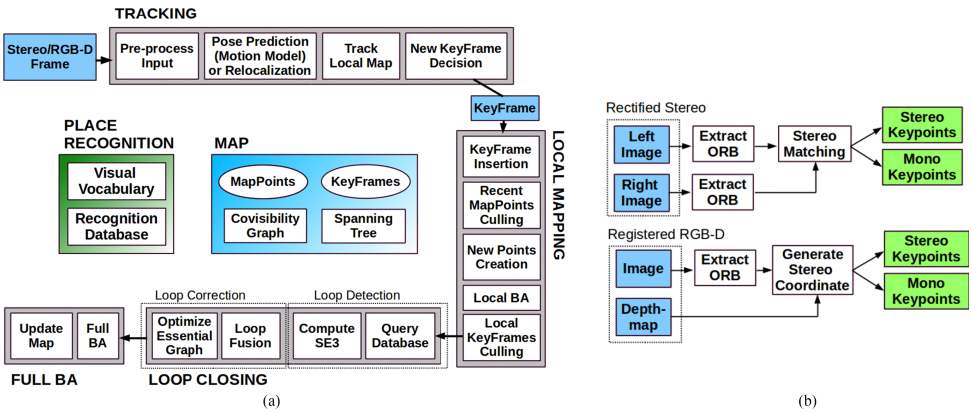


Figure 5.2: ORB-2 SLAM architecture overview.[1]

with every new frame by finding key feature matches in the local map with the help of motion-only Bundle Adjustment. Second thread maintains a local map using local Bundle Adjustment. Third thread performs detection of loops in the map and in case a loop is found starts another thread which performs global Bundle Adjustment.

Considering m points in a 3D scene which are denoted $\mathbf{x}_i$ and n camera views which are represented by camera matrix $\mathbf{M}_j$ Bundle Adjustment as defined in [4] can then be formulated as an optimization problem as

$$\min_{\mathbf{x}_i, \mathbf{M}_j} \sum_{i=1}^{m} \sum_{j=1}^{n} \left[\left(\frac{\mathbf{m}_1^j \mathbf{x}_i}{\mathbf{m}_3^j \mathbf{x}_i} - \mathbf{u}_j^j \right)^2 + \left(\frac{\mathbf{m}_2^j \mathbf{x}_i}{\mathbf{m}_3^j \mathbf{x}_i} - \mathbf{u}_j^j \right)^2 \right] \tag{5.9}$$

where $\mathbf{m}_i^j$ denotes i-th row of camera matrix $\mathbf{M}_j$ as defined in equation 3.4.

Implementation

ORB-2 SLAM implementation can be found on Raul Mur-Artal Github repository[21]. In my thesis I use version based on commit with the specified SHA-1 hash *f2e6f51cdc8d067655d90a78c06261378e07e8f3*.

Implementation takes the following arguments tied to feature extraction:

1. ***nFeatures*** defines number of ORB features extracted per frame
2. ***nLevels*** determines number of levels in the scale pyramid
3. ***scaleFactor*** scaling factor between levels in the scale pyramid
4. ***iniThFAST*** thereshold for FAST extraction
5. ***minThFAST*** thereshold for FAST extraction in case of no extracted corners

Also to reconstruct 3D points, ORB-2 SLAM needs camera calibration parameters from section 4.1 excluding distortion parameters, because ZED Stereo Camera provides already corrected images and bag of binary words used in BRIEF stage. This approach is implemented in *orb2_slam*.

5.3.3 ZED SLAM

Principle

ZED SLAM is a commercial SLAM implementation by Stereolabs Inc. and the time of writing this thesis said company have not made their algorithm public, and their description of the used algorithm is relatively vague.

> *"The ZED uses visual tracking of its surroundings to understand the movement of the user or system holding it. As the camera moves in the real-world, it reports its new position and orientation. This information is called the camera 6DoF pose. Pose information is output at the frame rate of the camera, up to 100 times per second in WVGA mode."*
> [22, **Stereolabs Inc.**]

Due to the nature of the ZED Stereo Camera, I will try to make an educated guess, but first I would like to list a few observations:

- ZED SLAM implementation runs on the same frequency as image acquisition.
- Some key features of unknown nature are already extracted in order to reconstruct 3D scene.
- ZED SLAM can keep a local map of the camera surroundings.

Based on these observations my hypothesis is that their software keeps track of previously extracted features from correspondence determination step of 3D reconstruction. ZED SLAM then reuses those features to compute optimal transformation between scans. This pose can later be refined based on some kind of local map, possibly with a loop closure detection algorithm.

The fact that ZED SLAM implementation is not known does not bring any new value from the scientific point of view but serves as a comparison between a finished commercial product and public open source implementation of published approaches from scientific community.

Implementation

ZED SLAM implementation has the following parameters:

1. ***initial_world_transform*** defines an initial position of the camera in the world coordinate frame
2. ***enable_spatial_memory*** enables ZED to keep a local map
3. ***enable_pose_smoothing*** provides pose refinement based on interpolation between poses
4. ***set_floor_as_origin*** initializes the tracking aligned with the extracted floor plane

Due to nature of use (via ZED SDK), it is not implemented as standalone software, but is a part of *data_gatherer* program.

Chapter 6

Practical experiments

In this chapter, I describe the testing environment in which all of the practical experiments were conducted. In addition obtained results are presented and evaluated.

6.1 Lab environment

For capturing of datasets and processing point clouds I used laptop **ASUS ROG GL702VT** with CPU **IntelCore i76700HQ** and dedicated graphic card **NVIDIA GeForce GTX 970M**. As an operating system I used Linux **Ubuntu 18.04.2 LTS**. Experiments were conducted with **NVIDIA CUDA V9.0.176** and graphic card driver version **390.116**.

For comparison of SLAM algorithms, computational time and accuracy are generally used as those are two most defining factors. Time is easily measured by computer, that is performing SLAM, accuracy, on the other hand, is much more complicated problem. There are multiple approaches to evaluating accuracy of SLAM algorithms. The one I use in this project works by capturing the motion of the stereo-camera with an external device and pairing each stereo camera measurement with the corresponding location and thus creating the "ground-truth".

For capturing "ground-truth", I used Vicon [23] motion capture system. Illustration of a laboratory environment used to capture "ground-truth" can be seen in fig. 6.1. ZED Stereo Camera is provided with special markers which are then tracked by multiple cameras in Vicon system.

Vicon provides 6 DOF position, which is relative to Vicon coordinate system. For experiment evaluation, I needed "ground-truth" relative to first measurement of ZED. This is easily accomplished by measuring transformation $\mathbf{A}(\mathbf{q}, \mathbf{t})$ from origin of Vicon $\mathbf{o}$ to origin of coordinate system of the the first measurement $\mathbf{o}^{'}$ and constructing inverse of said transformation. The new transformation $\mathbf{A}_i^{'}$ relative to the first measurement is then obtained as

$$\mathbf{A}_i^{'} = \mathbf{A}^{-1}(\mathbf{q}, \mathbf{t})\mathbf{A}_i \tag{6.1}$$

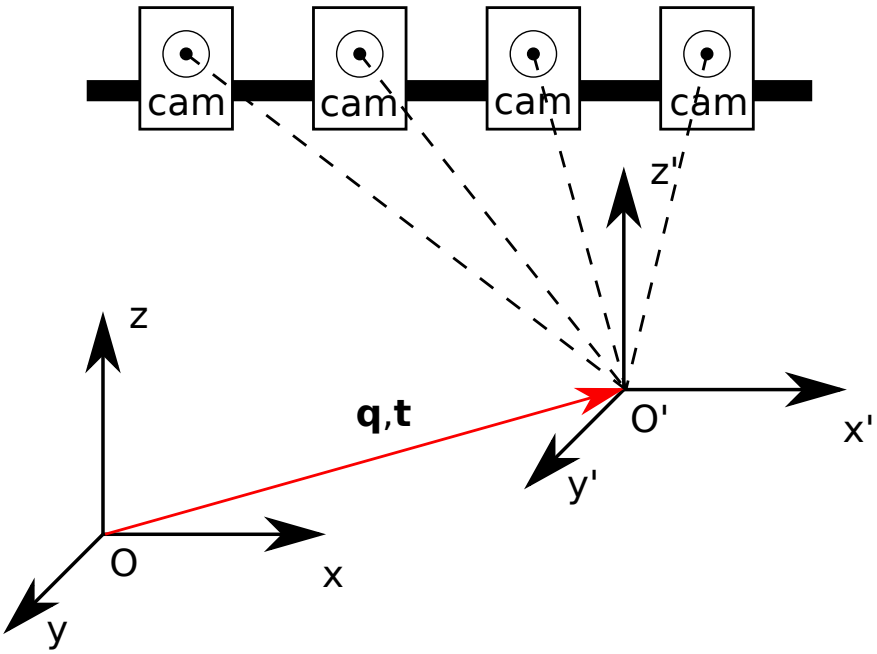


Figure 6.1: Vicon setup for measuring data-sets with ground truth.

where $\mathbf{A}_i$ is transformation relative to Vicon coordinate system and $\mathbf{q}$ is quaternion representing orientation and $\mathbf{t}$ is translational vector.

For capturing datasets with corresponding ground truth software called *data_gatherer* was developed. There are two parallel threads, first one gathers measurements from the stereo camera using ZED SDK and every time a new measurement is captured, it starts three new threads, to save an image, a binary depth map and a depth map as a picture. Second thread continuously receives ground truth measurements from Vicon using TCP/IP protocol.

6.2 Setting up algorithms

It is needed to find out the proper settings of previously stated algorithms. In order to do so, I captured two training datasets. These will be used to empirically find out optimal settings. Totally two datasets were collected with ZED Stereo Camera mounted on a wheeled vehicle to restrict the motion to two translational DOF and one rotational DOF. For ground truth collection, I used Vicon motion capture system. The logic behind this approach is that using restricted motion, it is easier to empirically estimate optimal settings. Training datasets are not attached because of the large file size. Results, obtained on training datasets, are included for reference in tables 6.2 and 6.4. The error was computed between the last output pose of SLAM algorithm and corresponding ground truth.

6.2.1 Training dataset 1

Dataset info	Value
Trajectory	15.1353 m
Number of scans	864 [-]
Camera FPS	15 FPS
Scan resolution	672 x 376

Table 6.1: Information about first training dataset.

Algorithm	Translational error [m]	Angular error [◦]	Processing time [s]
ICP	1.14	35.886	797.170
ORB-2	0.118	0.834	15.142
ZED	0.480	11.442	57.600

Table 6.2: Evaluation of training dataset 1.

6.2.2 Training dataset 2

Dataset info	Value
Trajectory	6.321 m
Number of scans	416 [-]
Camera FPS	15 FPS
Scan resolution	672 x 376

Table 6.3: Information about second training dataset.

Algorithm	Translational error [m]	Angular error [◦]	Processing time [s]
ICP	1.011	3.166	397.373
ORB-2	0.450	1.902	5.206
ZED	0.336	4.284	27.733

Table 6.4: Evaluation of training dataset 2.

6.2.3 Optimal settings

ICP

Finding optimal settings for Iterative Closest Point algorithm proved to be the most difficult out of all three used algorithms. I was not able to find any setting for which algorithm runs at the ZED Stereo Camera scanning rate of 15 Hz. Imposing imaginary boundary, for a SLAM algorithm to be considered real time capable, to be at-least 0.5 FPS, anything more than 100 iterations of ICP proved to be unacceptable. On the other hand, lowering $maxIterations$

directly impacted resulting accuracy. Parameter *theresDistMin* was set in a similar fashion.

Parameters *theresDist* and *theresAngl* distance serve as correspondence rejection attributes. Higher values increase computational time while sometimes increasing accuracy. Lowering said attributes can cause too few correspondences to be found, thus making it impossible to find the correct transformation between scans. Settings for which ICP performed best on training datasets can be seen in table 6.5.

Parameter	Value
maxIterations	100
alpha	0.5
theresDist	0.05
theresAngl	0.01
theresDistMin	0.1

Table 6.5: Empirically determined ICP SLAM settings.

ORB-2

Attribute *nFeatures* sets an upper boundary to the number of extracted ORB features. Raising above 1000 features did not improve accuracy while only increasing run time and around 500 I noticed a decrease in precision, thus optimal value I find to be 800. Next pair of parameters is *nLevels* and *scaleFactor* both are tied to FAST feature extraction scale pyramid explained in section 5.3.2. Setting *nLevels* to higher values improves partial scale variance, but increases computational time.

For example, for close quarters, it makes sense to set this parameter to lower values because the change in scale of the features is minimal. On the other hand consider a stereo camera mounted on the front of a moving car with optical axes collinear to the direction of motion, in this case, there is a considerable change in scale between each frame and number of detected features can be greatly improved. Parameter *scaleFactor* should then be adjusted according to number of levels and camera resolution. For general purpose best settings proved to be ten levels of the pyramid with the scale factor of 1.2. For an outdoor environment with I would advise to increase the number of levels and to lower scale factor and when deployed to close quarters or for the purpose of mapping historical monuments to do the opposite.

Last two parameters *iniThFAST* and *minThFAST* influences the threshold for sorting pixels into bins during FAST extraction. Adjusting of these parameters is done according to expected illumination and thus resulting contrast of images. For darker scenes, they should have a higher value, for an environment with more illumination lower. Table 6.6 presents settings for which ORB-2 SLAM performed the best on the training datasets. It should be noted that slight changes in those parameters did not have a drastic influence on accuracy.

Parameter	Value
nFeatures	800
nLevels	10
scaleFactor	1.2
iniThFAST	20
minThFAST	7

Table 6.6: Empirically determined ORB-2 SLAM settings.

ZED

Only parameters which directly affect accuracy are *enable_spatial_memory* and *enable_pose_smoothing*. Setting first parameter to *true* considerably improved accuracy on testing datasets. Parameter *enable_pose_smoothing* on the other hand lead to false corrections. Full settings can be seen in table 6.7.

Parameter	Value
initial_world_transform	-
enable_spatial_memory	true
enable_pose_smoothing	false
set_floor_as_origin	false

Table 6.7: Empirically determined ZED SLAM settings.

6.3 Results

This section presents a performance comparison on captured testing datasets. Testing datasets were captured in the same environment described in section 6.1, but with different trajectories. All algorithms were run with settings described in section 6.2.3 and were not manipulated or optimized to improve results on testing datasets. These datasets are complex and non trivial, in other words, combined translation and rotation is present in all three axes. Experiments were conducted walking around the laboratory while holding ZED Stereo Camera in hand.

6.3.1 Testing dataset 1

Results

Information about the first testing dataset can be seen in table 6.8, trajectory value is an absolute distance the camera travelled during the whole dataset. Results, which seen in table 6.9, are distances between the last output pose of SLAM algorithm and corresponding ground truth. Processing time for ZED SLAM is devised from camera FPS, the real computational time can not be measured directly. It is important to note, that ZED SLAM can run on 100 FPS for this resolution. Figures 6.2 and 6.3 show instantaneous error for each pose computed by SLAM algorithm.

Dataset info	Value
Trajectory	10.830 m
Number of scans	121 [-]
Camera FPS	15 FPS
Scan resolution	672 x 376

Table 6.8: Information about first testing dataset.

Algorithm	Translational error [m]	Angular error [◦]	Processing time [s]
ICP	2.228	22.686	228.005
ORB-2	0.344	6.795	1.928
ZED	0.578	4.436	8.000

Table 6.9: Evaluation of testing dataset 1.

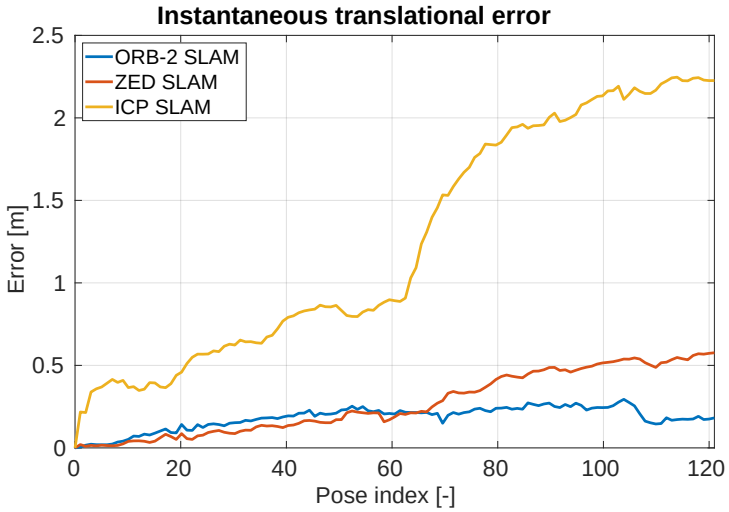


Figure 6.2: Translational error for dataset 1.

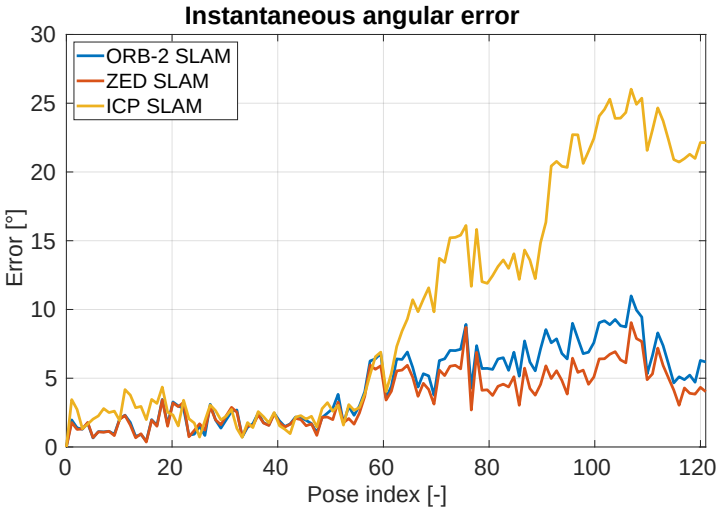


Figure 6.3: Angular error for dataset 1.

6.3.2 Testing dataset 2

Results

Information about the second testing dataset can be seen in table 6.10 and results can be seen in table 6.11. Figures 6.4 and 6.5 show error for each computed pose.

Dataset info	Value
Trajectory	16.613 m
Number of scans	331 [-]
Camera FPS	15 FPS
Scan resolution	672 x 376

Table 6.10: Information about second testing dataset.

Algorithm	Translational error [m]	Angular error [◦]	Processing time [s]
ICP	3.036	8.488	342.607
ORB-2	0.095	6.452	3.662
ZED	0.044	3.064	22.066

Table 6.11: Evaluation of testing dataset 2.

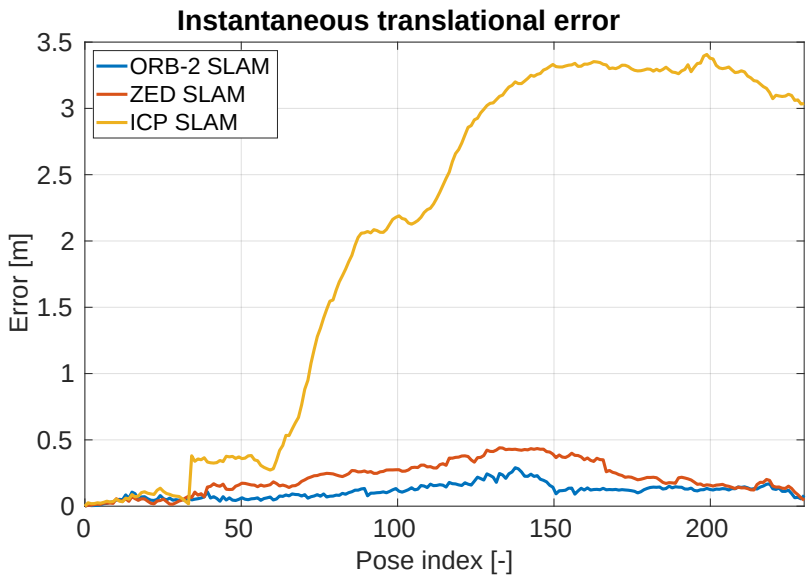


Figure 6.4: Translational error for dataset 2.

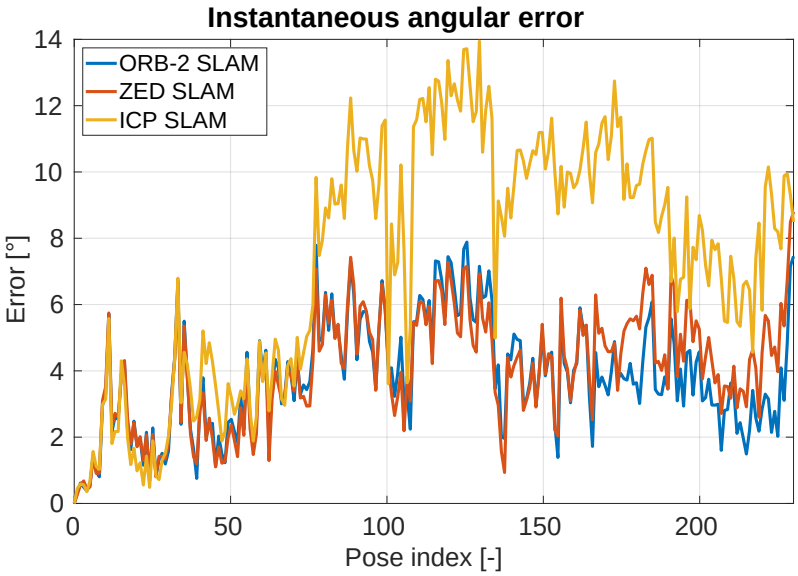


Figure 6.5: Angular error for dataset 2.

6.3.3 Testing dataset 3

Results

Information about third testing dataset can be seen in table 6.12 and results can be seen in table 6.13. Figures 6.6 and 6.7 show error for each computed pose. In this dataset camera optical axis is mostly co-linear to motion.

Dataset info	Value
Trajectory	24.221 m
Number of scans	727 [-]
Camera FPS	15 FPS
Scan resolution	672 x 376

Table 6.12: Information about third testing dataset.

Algorithm	Translational error [m]	Angular error [◦]	Processing time [s]
ICP	7.418	22.618	970.307
ORB-2	0.991	10.527	13.7595
ZED	1.043	12.836	48.466

Table 6.13: Evaluation of testing dataset 3.

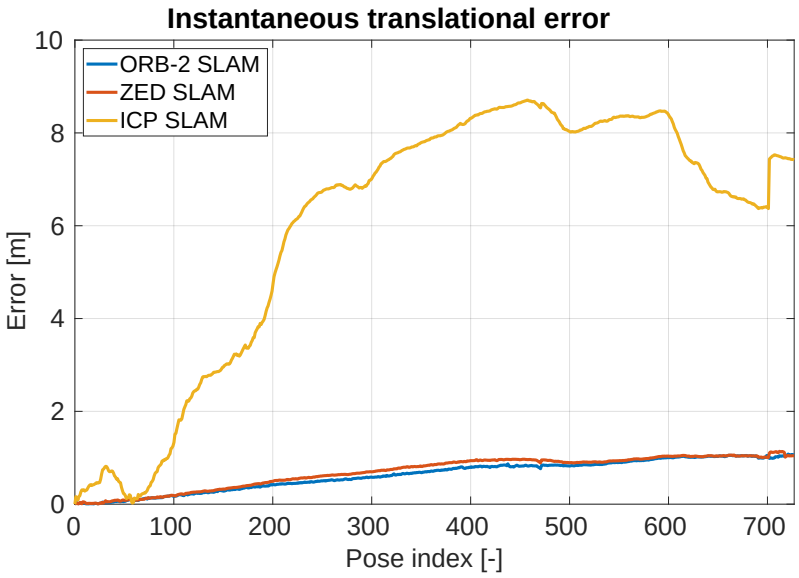


Figure 6.6: Translational error for dataset 3.

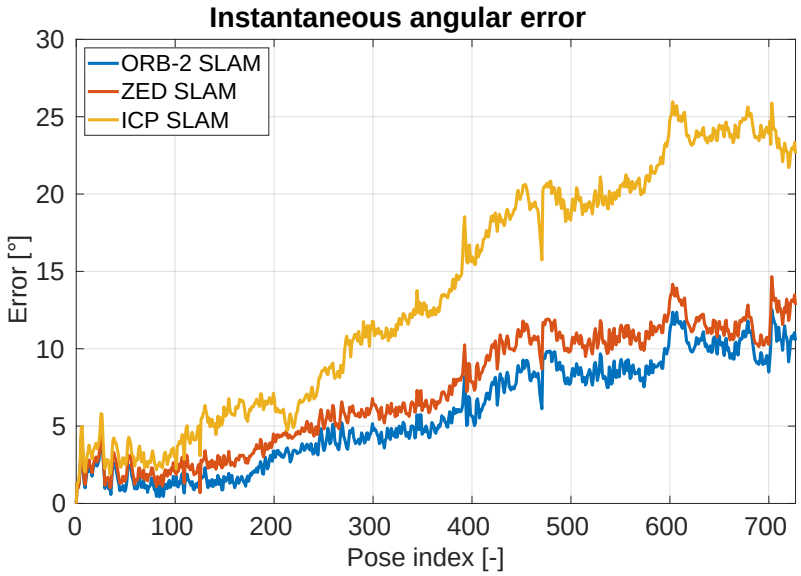


Figure 6.7: Angular error for dataset 3.

6.3.4 Discussion

It is obvious that ICP SLAM is an inferior approach in both accuracy and processing time. There is an intuitive explanation for this. Looking at the captured 3D scan in figure 4.2, one can clearly see there is a lot of "noise" in the measurement, walls are not perfectly flat, and objects suffer from obvious deformation. Because in ICP all points are considered equal, even those noisy points are processed. A possible solution would be to either try correcting 3D scans based on for example plane detection. This could, in theory work, but would increase the processing time even further.

The other question is why ORB-2 SLAM and ZED SLAM perform much better. For this, I can also offer an intuitive explanation. When ZED Stereo Camera reconstructs the 3D scene, it extracts features in both images. It is obvious that distinctive points (corners, sudden color change, etc.) are easier to match, than non-distinctive points. This, in turn, means distinctive points are less likely to be mismatched. ORB-2 SLAM and ZED SLAM extract features from images which are likely to be a subset of features from a reconstruction of the 3D scene. This means that both of these algorithms compute the transformation based on points which have a high chance of being reconstructed properly. Of course, my statement can not be formally proven, because the precise process of reconstruction, performed by ZED Stereo Camera, is not known.

Conducted experiments clearly show, that ICP approach is sub-par compared to other algorithms and that ORB-2 SLAM and ZED SLAM performed similarly in terms of accuracy.

Chapter 7

SLAM with a mobile robot

This chapter provides a description of how to incorporate odometry, which is information provided by wheel odometers of a mobile robot, into previously proposed SLAM algorithms. Because used robot moves on a plane, constraints outlined in section 5.1 will be further restricted to 2 DOF of translation and 1 DOF of rotation.

7.1 Theory

7.1.1 Mobile robot

Robot model

To describe robot motion I use a pure kinematic model proposed by [24]. The idea behind this representation is to use state space approach to model robot kinematics. This has obvious advantage when working with Extended Kalman Filter as discussed later. If x_k, y_k and θ_k denote current states with x_k being x coordinate, y_k being y coordinate and θ_k an angle representing orientation of the robot, then state space representation can be described as

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \Delta s_k \cos(\theta_{k-1} + \Delta\theta_k/2) \\ \Delta s_k \sin(\theta_{k-1} + \Delta\theta_k/2) \\ \Delta\theta_k \end{bmatrix} \tag{7.1}$$

$$\Delta s_k = (\Delta r_k + \Delta l_k)/2 \tag{7.2}$$

$$\Delta\theta_k = (\Delta r_k - \Delta l_k)/b \tag{7.3}$$

where Δr_k and Δl_k are translational distances of the right/left wheel, which are treated as control inputs, and b is distance between wheels. It is important to note that proposed kinematic model is non-linear.

Experimental platform

As experimental platform HUSKY [25] was used. Although this robot has four wheels it can still be modelled with equation 7.1 provided wheels at the same side turn at approximately the same speed. Overview of the experimental

platform can be seen in figure 7.1. ZED Stereo Camera is mounted on the HUSKY robot with its optical axis being perpendicular to robot motion.

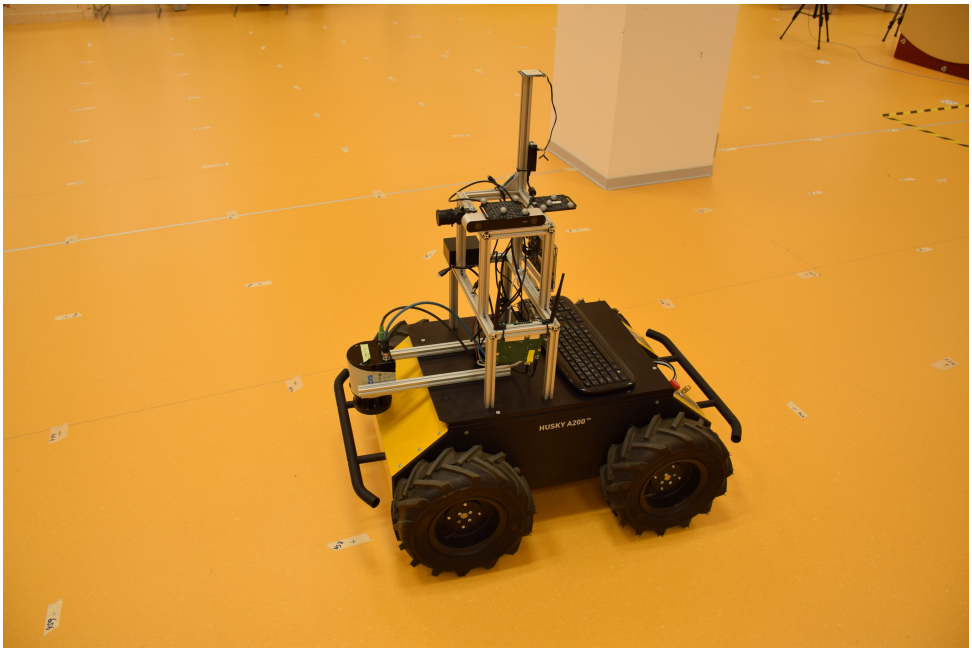


Figure 7.1: Experimental platform consisting of HUSKY robot and ZED Stereo Camera.

7.1.2 Extended Kalman filter

Principle

Because the robot model presented in section 7.1.1 is not linear, I can not use regular Kalman Filter [27] for sensor fusion. **EKF** or **E**xtended **K**alman **F**ilter [26] is an extension of regular Kalman filter for non-linear systems. Regular KF is an optimal linear quadratic estimator, but for EKF optimality is generally not guaranteed. Nevertheless, it is still one of the most widely used algorithms for estimation.

For nonlinear systems discrete time state space model can be represented as

$$\mathbf{x}_k = \mathbf{f}(x_{k-1}, u_k) + \mathbf{w}_k \tag{7.4}$$

$$\mathbf{y}_k = \mathbf{h}(x_k, u_k) + \mathbf{v}_k \tag{7.5}$$

where functions $\mathbf{f}(x_k, u_k)$ and $\mathbf{h}(x_k, u_k)$ are non-linear vector functions of state transitions. Vector $\mathbf{w}_k$ is called process noise and $\mathbf{v}_k$ is called measurement noise, both are assumed to be zero mean Gaussian noises.

Extended Kalman Filter works in two steps. First is prediction step described by equations 7.6 and 7.7 and update step defined with equations from

7.9 to 7.1.2.

Predict step is performed when the system receives a control input. First, current estimate of state is predicted using control input and state transition function

$$\hat{\mathbf{x}}_k = \mathbf{f}(x_{k-1}, u_k) \tag{7.6}$$

and then state co-variance matrix is adjusted as

$$\mathbf{P}_k = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^T \tag{7.7}$$

where $\mathbf{Q}_k$ is control input co-variance matrix and $\mathbf{F}_k$ is Jacobi matrix of function $\mathbf{f}(x_k, u_k)$ defined as

$$\mathbf{F}_k = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}_{|\mathbf{x}_{k-1}, \mathbf{u}_k} \tag{7.8}$$

f_i marks i-th row of state transition function from equation 7.6 and x_j stands for j-th state variable. Matrix $\mathbf{G}_k$ is also Jacobi matrix, but differentiated with respect to control input variables.

Update step is performed every time a new observation of states $\mathbf{z}_k$ is obtained. First innovation $\mathbf{e}_k$ is computed as

$$\mathbf{e}_k = \mathbf{z}_k - h(\mathbf{x}_k, \mathbf{u}_k). \tag{7.9}$$

Then Kalman gain $\mathbf{K}_k$ is computed as

$$\mathbf{K}_k = \mathbf{P}_{k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \tag{7.10}$$

where $\mathbf{H}_k$ is Jacobian matrix of function $\mathbf{h}(x_k, u_k)$ differentiated with respect to state variables and $\mathbf{S}_k$ is defined as

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k-1} \mathbf{H}_k + \mathbf{R}_k \tag{7.11}$$

Using equations 7.9 and 7.10 states can be updated as

$$\hat{\mathbf{x}}_k = \mathbf{x}_k + \mathbf{K}_k \mathbf{e}_k \tag{7.12}$$

with respective state co-variance matrix updated as well

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\, \mathbf{P}_{k-1}. \tag{7.13}$$

From equations above it is obvious that EKF works similar to KF, but uses linearization of the state transition and output function.

Implementation

EKF was implemented using Matlab in *ekf*. Flowchart of the implementation can be seen in figure 7.2. The output of the RGB-D SLAM algorithm serves as state measurement, and odometry is a control input. As state transition function as described in equation 7.6 I used the kinematic robot model from equation 7.1, which gives following state space form

$$\mathbf{x}_k = \begin{bmatrix} x_{k-1} + \Delta s_k \cos(\theta_{k-1} + \Delta\theta_k/2) \\ y_{k-1} + \Delta s_k \sin(\theta_{k-1} + \Delta\theta_k/2) \\ \theta_{k-1} + \Delta\theta_k \end{bmatrix} \tag{7.14}$$

$$\mathbf{y}_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}. \tag{7.15}$$

Constructing Jacobi matrix $\mathbf{F}_k$ from 7.14 as outlined in equation 7.8 yields

$$\mathbf{F}_k = \begin{bmatrix} 1 & 0 & -\Delta s_k \sin(\theta_{k-1} + \Delta\theta_k/2) \\ 0 & 1 & \Delta s_k \cos(\theta_{k-1} + \Delta\theta_k/2) \\ 0 & 0 & 1 \end{bmatrix}_{|\mathbf{x}_{k-1}, \mathbf{u}_k}. \tag{7.16}$$

Matrix $\mathbf{G}_k$ is obtained in similar fashion as

$$\mathbf{G}_k = \begin{bmatrix} \frac{\cos(\theta_{k-1}+c)}{2} + \frac{\sin(\theta_{k-1}+c)\Delta s_k}{8} & \frac{\cos(\theta_{k-1}+c)}{2} - \frac{\sin(\theta_{k-1}+c)\Delta s_k}{8} \\ \frac{\sin(\theta_{k-1}+c)}{2} - \frac{\cos(\theta_{k-1}+c)\Delta s_k}{8} & \frac{\sin(\theta_{k-1}+c)}{2} + \frac{\cos(\theta_{k-1}+c)\Delta s_k}{8} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix}_{|\mathbf{x}_{k-1}, \mathbf{u}_k} \tag{7.17}$$

with substitution term c defined as

$$c = \frac{(\Delta l_k - \Delta r_k)}{4}. \tag{7.18}$$

Output matrix $\mathbf{H}_k$ is computed from equation 7.15 as

$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{7.19}$$

Matrices $\mathbf{Q}_k$, $\mathbf{R}_k$ are estimated empirically, however their exact values is not as important as their ratio. Also looking at equation 7.7 one can see that $\mathbf{P}_k$ must have a non-zero initial value. $\mathbf{P}_1$ represents uncertainty of initial state and a rule of thumb is to set $\mathbf{P}_k$ as $\mathbf{P}_k := \mathbf{R}_k$.

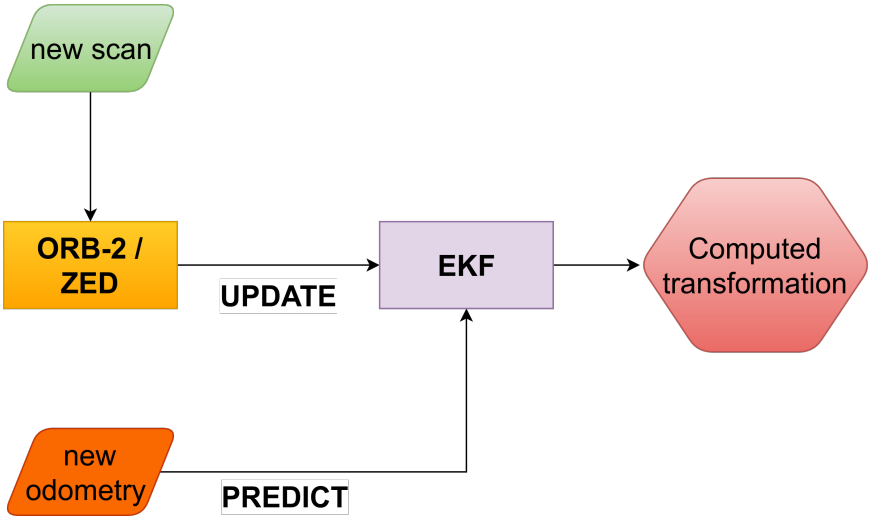


Figure 7.2: Flowchart of sensor fusion using EKF.

7.1.3 ICP with odometry

Extending Iterative Closest Point with odometry can be done by using the output of the equation 7.1 as an initial guess. This should, in theory, decrease ICP run time and also increase accuracy, if the odometry is precise enough. Reason for this is, that if scans are already nearly aligned ICP will require fewer iterations, thus decreasing run time and also there is a lesser chance to get stranded in local optimum, thus increasing accuracy. Also, the output of ICP can be improved by merging it with odometry using EKF. Overview of the whole process can be seen in figure 7.3.

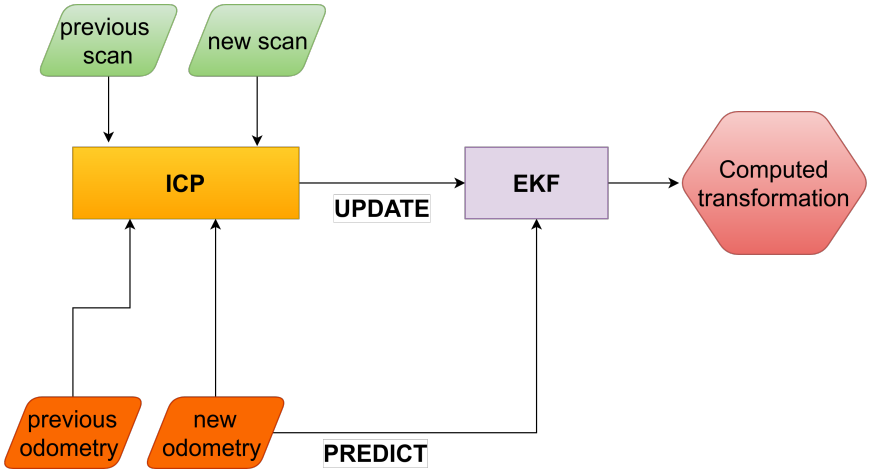


Figure 7.3: Flowchart of ICP scan alignment with additional odometry input.

7.2 Results

Dataset was captured with *data_gatherer* program which was extended to receive odometry via TCP/IP connection from HUSKY robot. Information can be seen in table 7.1 and results in table 7.2. Figures 7.4, 7.5 and 7.6 represent 2D trajectory of the mobile robot. In each plot, there are four trajectories: ground truth captured by Vicon, unmodified odometry output and the comparison of SLAM algorithms versus their respective outputs of EKF.

Dataset info	Value
Trajectory	24.292 m
Number of scans	1245 [-]
Camera FPS	15 FPS
Scan resolution	672 x 376

Table 7.1: Information about first testing dataset.

Algorithm	Trans error [m]	Angular error [○]	Processing time [s]
ICP + odometry	0.544	7.278	2244.006
ICP + odometry + EKF	0.596	7.394	2244.006 + 0.101
ORB-2	0.118	1.185	24.193
ORB-2 + EKF	0.111	1.171	24.193 + 0.0847
ZED	0.983	9.620	83.000
ZED + EKF	0.940	9.452	83.000 + 0.0803

Table 7.2: Evaluation of testing dataset 1.

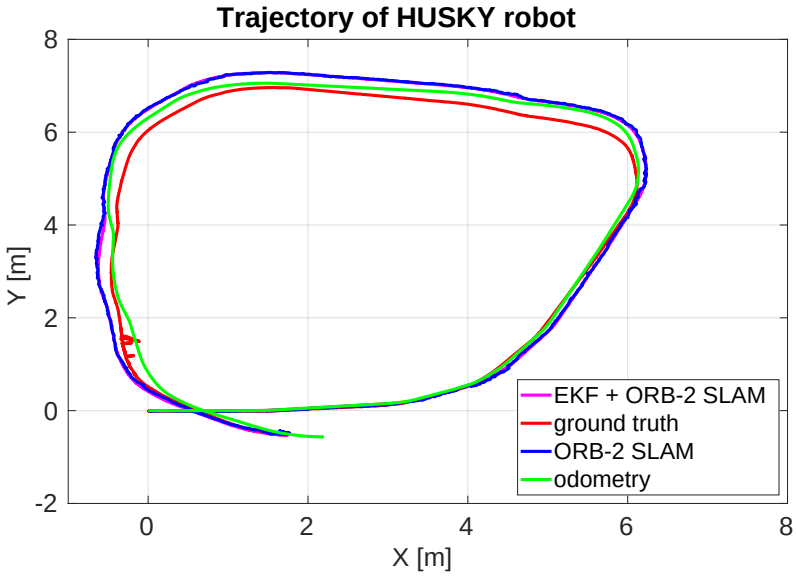


Figure 7.4: Trajectory comparison of ORB-2 SLAM with EKF odometry fusion.

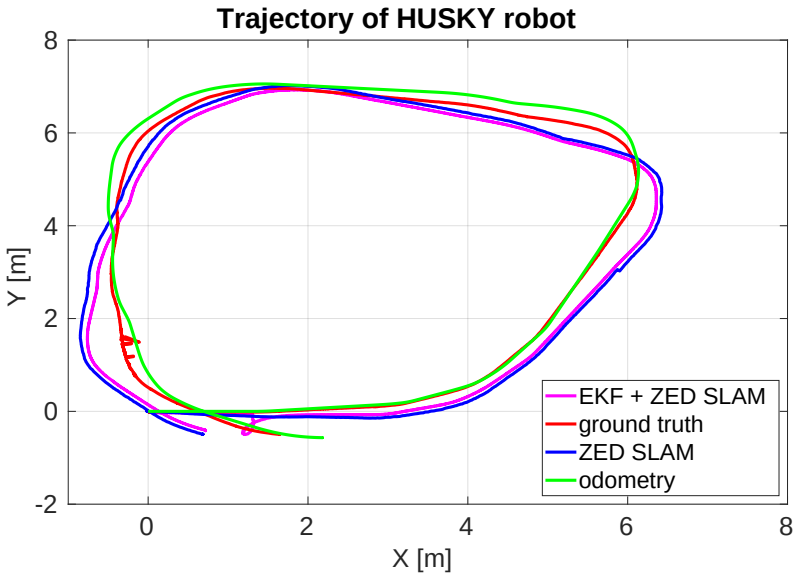


Figure 7.5: Trajectory comparison of ZED SLAM with EKF odometry fusion.

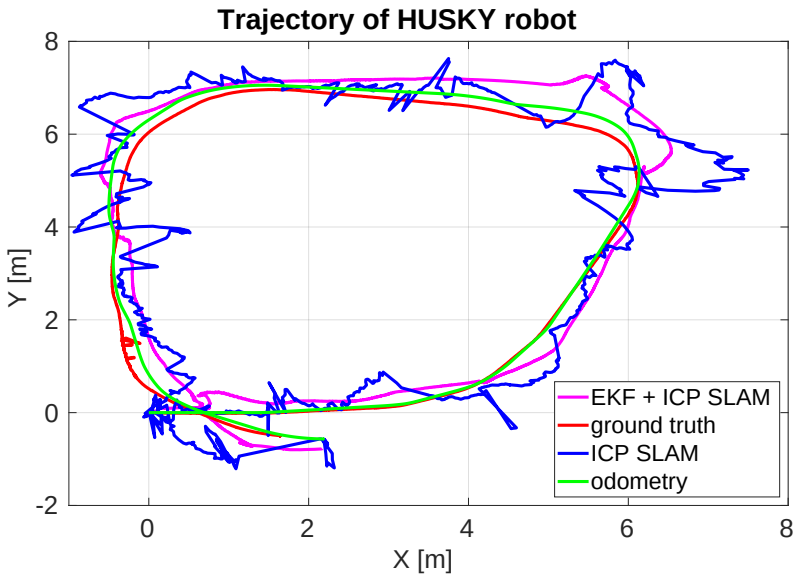


Figure 7.6: Trajectory comparison of ICP SLAM with EKF odometry fusion.

7.2.1 Discussion

While there was only a questionable increase in accuracy for ORB-2 SLAM and ZED SLAM, there was a substantial improvement for ICP SLAM in terms of overall trajectory. However considering an extremely long run time compared to the other two algorithms, it is safe to say, that even with the help of additional sensors, ICP algorithms is not the right approach for ZED Stereo Camera.

Even though the improvement was only minor for the other two algorithms, there is also a benefit of not being dependent on only one sensor. For example, imagine a scenario, with sudden occlusion in camera objective caused by some external disturbance. Without additional sensors robot would lose track of the environment and might not be able to localize again, but with odometry input disturbance recovery is quite simple.

Chapter 8
Conclusion

8.1 Evaluation

The first goal of this thesis was to get familiar with a stereo camera as a sensor providing RGB-D measurements, the second goal of this thesis was to find one or more SLAM approaches capable of working with the stereo camera and produce an evaluation in terms of accuracy and computational time. Both of them were fulfilled and, in addition, this thesis presents a real-world application with mobile robot navigation.

The most valuable contribution of this work is a detailed performance analysis of SLAM algorithms with ZED Stereo Camera based on captured ground truth. Most of the published studies about RGB-D SLAM deal with the active stereo camera like Microsoft Kinect and not with the passive stereo devices such as ZED Stereo Camera. In this contex,t my thesis presents much-needed data.

Based on the results of my study, it can be seen that ICP is generally not suitable for SLAM with the passive stereo camera as it obviously lacks in both accuracy and computational time compared to other tested algorithms. While ORB-2 SLAM performed slightly better ZED SLAM, it is interesting to note that a commercial product was almost on par with the published state of the art approach.

Last part of my thesis presented a real-world application of simultaneous localization and mapping with a mobile robot, using approaches from previous chapters, which was not necessary to fulfill my bachelor project requirements, but it fits the context of this work.

8.2 Future work

There are three main areas where this work could progress further, the sensor, RGB-D SLAM algorithm and application.

ZED Stereo Camera is still lacking in the accuracy (visible just by visual

inspection), which means results could be improved if company Stereolabs Inc. improved their product or by using different stereo camera altogether. Another problem I see with the sensor itself is its dependency on an external GPU, which poses a difficulty, especially for applications with UAVs. Not only does an external GPU increases the cost of the whole system, but it also takes spaces, power and adds weight. One possible solution to some of the problems would be to implement the 3D reconstruction process on FPGA.

While RGB-D SLAM algorithms are still a hot topic for many researchers, there is still much work to be done. Although best of the tested algorithms ORB-2 SLAM showed very promising results and could sometimes correct a drifting error by performing a loop closure, there is still an obvious bottleneck for long term navigation (distances in the order of kilometres).

The method proposed in section 7.1.2 could be extended for use with UAVs, which can prove challenging for a number of reasons. First, there is no such thing as wheel odometry for UAVs in terms of reliability and sensor cost. This could be overcome with a fusion of multiple measurements from different sensors. Another issue with EKF is lack of reliable vector space representation of rotation, although this problem could be dealt with by using the approach outlined in [28].

Bibliography

[1] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[2] Reza N. Jazar. *Theory of applied robotics kinematics, dynamics, and control*. Springer, 2 edition, 2010.

[3] Du Q. Huynh. Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.

[4] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image processing, analysis, and machine vision*. Thompson Learning, 3 edition, 2008.

[5] Duane C. Brown. Close-range camera calibration. *PHOTOGRAMMETRIC ENGINEERING*, 37(8):855–866, 1971.

[6] David Forsyth and Jean Ponce. *Computer vision: a modern approach*. Pearson, 2012.

[7] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Computer Vision – ECCV 2006 Lecture Notes in Computer Science*, page 404–417, 2006.

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

[9] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. *2011 International Conference on Computer Vision*, 2011.

[10] C. Chang and S. Chatterjee. Quantization error analysis in stereo vision. pages 1037–1041 vol.2, Oct 1992.

[11] ZED Stereo Camera. `https://www.stereolabs.com/zed/`. Accessed: 2019-04-11.

[12] ZED API Reference. `https://www.stereolabs.com/docs/api/`. Accessed: 2019-04-18.

[13] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. *Experimental Robotics Springer Tracts in Advanced Robotics*, page 477–491, 2014.

[14] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2018.

[15] Jianbo Shi and Tomasi. Good features to track. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*, 1994.

[16] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. *Computer Vision – ECCV 2010 Lecture Notes in Computer Science*, page 778–792, 2010.

[17] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*, 2013.

[18] P.j. Besl and Neil D. Mckay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.

[19] The Mobile Robot Programming Toolkit API Reference. `http://mrpt.ual.es/reference/1.5.5/`. Accessed: 2019-04-23.

[20] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119, 2010.

[21] ORB2 SLAM Github repository. `https://github.com/raulmur/ORB_SLAM2`. Accessed: 2019-04-11.

[22] ZED SDK Documentation. `https://www.stereolabs.com/docs/positional-tracking/#how-it-works`. Accessed: 2019-04-18.

[23] Vicon Vantage Motion Capture Camera. `https://www.vicon.com/products/camera-systems/vantage`. Accessed: 2019-04-23.

[24] Bruno Siciliano. *Robotics: modelling, planning and control*. Springer, 2009.

[25] Vicon HUSKY unmanned ground vehicle. `https://www.clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/`. Accessed: 2019-04-28.

[26] Dan Simon. *Optimal state estimation: Kalman, H and nonlinear approaches*. Wiley-Interscience, 2006.

[27] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35, 1960.

[28] Christoph Hertzberg, René Wagner, Udo Frese, and Lutz Schröder. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion*, 14(1):57–77, 2013.

Appendix A

List of Mathematical Notation

Symbol	Meaning
$\mathbf{r}$	Vector of numbers.
$\mathbf{A}$	Matrix of numbers.
$\mathbf{A}^{-1}$	Inverse of matrix $\mathbf{A}$.
A	Set.
$\hat{x}$	Estimated value.
$\mathbf{q}^*$	Complex conjugate.
$f(x_1, x_2, .., x_n)$	Scalar function of n variables.
$\mathbf{f}(x_1, x_2, .., x_n)$	Vector function of n variables.
Δx	Amount of change in variable x between two events.
x_k	Variable x in discrete time point k.

Appendix B

List of Abbreviation

Symbol	Meaning
DOF	Degrees of Freedom.
UAV	Unmanned Aerial Vehicles
GPU	Graphical Processing Unit
FPGA	Field Programmable Array
KF	Kalman Filter
EKF	Extended Kalman Filter
SLAM	Simultaneous Localization And Mapping
RGB	Red Green Black
RGB-D	Red Green Black - Depth
FPS	Frames per second

Appendix C

DVD Content

File	Content
F3-BP-2019-Lukas-Majer.pdf	This thesis in PDF format.
icp_slam/	Implementation of ICP SLAM.
orb2_slam/	Implementation of ORB-2 SLAM.
data_gatherer/	Software for capturing datasets.
evaluator/	Software for evaluating SLAM algorithms.
ekf/	Implementation of EKF sensor fusion.
datasets/	Captured datasets used for evaluation.