TU VIENNA TECHNISCHE UNIVERSITÄT WIEN VIENNA UNIVERSITY OF TECHNOLOGY

DIPLOMARBEIT

Implementing a Time-of-Flight Camera Interface for Visual Simultaneous Localization and Mapping

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
Dipl.-Ing. Peter Gemeiner
E376
Institut für Automatisierungs und Regelungstechnik

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik

von

Peter Jojic
9626808
Wiedner Hauptstrasse 135/14
1050 Wien

Wien, November 2008

Peter Jojic

Abstract

To navigate successfully in an unknown environment, mobile robots have to know their location, and they need a map of the scene. These two necessities cannot be separated and for navigation purposes they have to be solved simultaneously. The combination of these tasks is known within the robotics community as Simultaneous Localization and Mapping (SLAM).

Different sensors can be used to solve SLAM, but we think that a camera is the most appealing option, this is because it provides dense information content. Using the standard single perspective-projective camera as the only SLAM sensor has two major disadvantages. First, the depth information is immediately lost. To estimate the robot's location and positions' of scene landmarks, the camera has to move and perceive the environment from several different views. Second, the features lying at occlusion boundaries can not be distinctively rejected. However, false features can cause SLAM to collapse.

In this thesis, a recently developed Time-of-Flight (ToF) camera is used as the only sensor input for SLAM. The ToF sensor provides 2D images as the standard perspective-projective camera, but it can also measure the position of the scene features directly. Presented in this work is a new interface for a vision SLAM framework, which incorporates ToF sensor readings in real-time. However, the ToF cameras suffer from several noise effects, e.g. scattering, mixed pixels etc. We present how these various noise effects influence the previously mentioned localization and mapping problem.

Initially the experimental results for the selected vision SLAM framework using the ToF camera performed well, when enough near distant features have been available. In case new features were not detectable, SLAM usually gets instable or lost.

To tackle the problem of false scene landmarks lying at occlusion boundaries a concept is presented. The idea of this concept is to straightforwardly use the measured 3D information to analyze the cornerness of a landmark. Simulated results show that landmarks can be identified using the analysis based on the eigen decomposition, and this can improve the real-time feature initialization.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

An important task of a mobile robot such as autonomous ground vehicle on Mars, driverless car or a robot at home is to localize its position in an unknown environment that the robot is placed in. To fulfill that task the robot needs to perceive and estimate the environment

Figure 1.1: Nasa Mars Rover, courtesy NASA/JPL-Caltech; Stanley, DARPA Grand Challenge (2005); Cleaning robot, ©RoboCup Federation Suzhou (2008).

by searching for new landmarks (features) to update its position within a map. This problem is commonly known as localization and mapping problem and can be addressed by a sequential probabilistic approach known as Simultaneous Localization and Mapping (SLAM). With sensors such as laser range finders, sonic sensors or digital cameras the robot can receive information from the surrounding environment and analyze the data as input for a SLAM algorithm. Laser scanners and sonic sensors provide typical distance measurements in a range from approx. 1-10 metres whereas digital cameras capture the reflected light as illumination and colour within an image sensor. They are not able to directly measure depth information, due to their projective nature. In fact one can only derive distance from an image by using two cameras, known as computer stereo vision or with a single camera due to motion based on structure from motion methods. By using a single camera multiple images are taken over time and from different position to reconstruct three-dimensional object-information. Vision-based sensors have been compared to lasers and shown several advantages. They are cheap, non-invasive, have no moving parts, can be used in various scenarios and provide a perception of the entire scene.

Solving the SLAM problem with a single perspective-projective camera as sensor input has several disadvantages regarding its missing direct depth measurement. With a single camera system a fixed target at a known distance is required for starting the localization and mapping process. Without a fixed landmark we have no information about the scale of the distance, and the robot could not build up a precise map of the environment and determine its position relative to it.To obtain depth information from the image landmarks, we need first to detect them. To get a good estimate of a landmark position in space, we need to observe it from different viewpoints. Therefore the camera always needs to move. In fact, direct depth measurement as additional information would help to relax these constraints and to improve the field of application for a single camera SLAM algorithm.

Recently developed range image cameras (RIM) measure in addition to the light intensity the distance by the Time-of-Fight (ToF) method. The principle is based on modulated light that is sent out by a signal emitter, partially reflected by an object and mapped onto a sensor array by optics. Using this sensor as an input for a single camera SLAM algorithm provides additional measured depth information that the algorithm can benefit from. In this work we investigate the benefits and drawbacks of using the ToF sensor as the only input for Simultaneous Localization and Mapping.

1.2 Problem Statement

In this work it shall be investigated if a Time-of-Flight camera can be used for Visual Simultaneous Localization and Mapping (SLAM). The following aim is to investigate the influence of the sensor properties onto the localization and mapping process. As a basis the sensors 2D image and its usability is investigated. The second aspect is to analyse how the 3D data using particular distance information can improve the localization and mapping algorithm and how available 3D information can be incorporated to detect features within the surrounding scene.

1.3 Approach Description

Using the depth information as a source for additional measurement data updates within existing Simultaneous Localization and Mapping algorithms, could help to eliminate some restrictions as mentioned above. The approach is based on existing hard and software components that supply: An existing MonoSLAM software framework provided by Andrew Davidson [12] implementing an algorithm that work based on a single camera as sensor. Including a graphical user interface displaying the robots (camera) location within a 3D map. A Time-of-Flight camera developed by Mesa Imaging [24] delivering amplitude and depth measurement as digital data is used to create 2D and 3D images of the recorded scene. A standard Linux based computer processing the framework software, camera Application and Programming Interface (API). The camera is connected via a Universal Serial Bus (USB) cable to the computer. Because the framework only works with FireWire and Gigabit Ethernet cameras a development of a new interface is required. In the first step the camera API is incorporated into the framework and the camera amplitude data is transformed into illumination data to represent a 2D image. Therefore the framework algorithm do not change and perform localization and mapping tasks only based on the intensity image. The influence of the sensor characteristics onto the detection, initialization and measurement of features from the acquired intensity images is evaluated. The second step shall evaluate the use of the additional depth data measurements. In particular to determine corners lying in the plane and corners having a 3D structure. The mathematical calculation that has been chose is the method called "structure tensor". It is based on gradient derivative of the distance data and enable it to differentiate different object structures. Such as corners, planes and lines. If possible a method for features depth initialization based on measured distance data shall be investigated and implemented. Another aim is to implement a 3D corners detector based on the consideration taken by the structure tensor analyse. It is expected that this improves the quality of the detected feature and shall help to identify occlusion boundaries.

1.4 Related Work

Different kind of sensors can serve as input for SLAM algorithms. Weingarten and Siegwart [45] demonstrate robot tracking and map building with a rotating 2D laser scanner generating 3D point clouds. Sonar sensors mounted on a robot applied to SLAM have been used by Zunino and Christensen [46]. Davison et al. [11] introduces a real-time SLAM algorithm (MonoSLAM) for feature tracking and mapping based on a single camera moving through an unknown scene. The algorithm calculates the camera motion and feature position in a 3D map with a standard Extended Kalman Filter (EKF). An alternative implementation is based on a Particle Filter algorithm (FastSLAM) by Pupilli and Calway [33]. Its main focus lies on effectively compensating unpredictable motion, which violate predictive motion models as used within Davisons MonoSLAM approach. Both approaches have in common that depth is not measured directly and thus for initialization a known target is needed. Also the depth of new features needs to be initialized over several image frames. Chekhlov et al. [6] introduce a SIFT-like spatial gradient feature description capable of robust matching over a wide of viewing angles and scales in real-time. The approach is based on an Unscented Kalman Filter (UKF). At initialization, a number of feature descriptors need to be generated, where in subsequent frames, descriptors at potential corresponding points are generated and used to predict the original point. Recent investigations in the area of Time-of-Flight cameras by May et al. [22] using visual odometry to estimate the camera pose based on the 3D measurements data. Mapping is performed by directly using the distance information from the sensor, and filtering methods are applied to enhance the measurement precision of it. Due to the partial offline processing of the map is the algorithm only partially suitable for online applications.

Chapter 2

Basic Theory

2.1 Probability Concept

The principles of random variables and probability are a prominent mathematical concept frequently used as a scientific method to analyse information in a non deterministic environment. In this section a short abstract is given this concept, this is not intended to be exhaustive. Starting with the definition of random variables, following three important probability concepts are explained: (1) The classical concept and its extension to the frequency definition, (2) Kolmogorov axiomatic definitions and (3) Bayesian probability.

In probabilistic robotics measurement data is received from sensors. Environment data like the robots or the objects position and orientation, are all described by the use of the Bayesian probability. Based on that, the concept of belief representation is introduced in Sec. 2.2.3 to derive the robots state and position. In this context are the distribution and density function used to describe probability in an analytic form. Another key concept is the conditional probability derived from the Bayes' theorem, which is denoted as the posterior probability.

2.1.1 Random Variables

As stated by Peebles [32] "A random variable X can be considered to be a function that maps all elements of the sample space into points on the real line or some parts thereof." A prominent example for this explanation is to toss a dice. The six possible outcomes $\{x_1, x_2, \ldots, x_6\}$ define a sample space S . To every outcome the function $X(\xi)$ assigns the numbers $\{1, 2, \ldots, 6\}$. $X(\xi)$ or the more common abbreviation X is than called a random variable.

Random variables can be described in a discrete or continuous manner [31] . A discrete random variable is having only discrete values as shown in the example above, whereas for continuous variables the outcome of the function has constant values. An important characteristic of random variables is its probability, we write it here

$$P(X = x) \; and \tag{2.1}$$

$$P(X \leq x). \tag{2.2}$$

$P(X = x)$ denotes the probability of the random variable X for a given value x. For example, in a game with a fair die the probability of X equals 1 is denoted $P(x = 1) = P(1) = 1/6$. In case the outcome may be 1, 2 or 3, respectively $\{X \leq 3\}$ the probability $P(X \leq 3)$ equals 3/6. $\{X \leq x\}$ is defined as the event of the probability $P(X \leq x)$. In case of a continuous random variable an event includes all values lower than x (including x). Therefore, if R is a set of real numbers on the x axis, then $x \in R$ [31]. Note that for continuous variables we get $P(X = x) = 0$.

2.1.2 Probability Definitions

The idea of probability follows several concepts. The classical definition defines probability as favorable outcomes in ratio to possible outcomes [31],

$$P(x) = \frac{N_x}{N}. \tag{2.3}$$

In the die experiment, this would be for the event even $P(x) = 3/6$. The are six possible outcomes three are favorable to the event. As explained by Papoulis et al. [31] this definition can only be applied to limited types of problems. Especially to those with same probability of fundamental events. For example every single dice event $\{x_1, x_2, \ldots, x_6\}$ equals the probability 1/6. A more common approach is the relative frequency definition. The probability is defined by the limit

$$P(x) = \lim_{n\to\infty} \frac{n_x}{n}, \tag{2.4}$$

where n_x is the number of occurrences of an event and n is the number of trials. The definition is fundamental in the applications of probability. However, in a physical experiment, n_x and n might be large but they are only finite. To approximate the ratio to a limit is therefore not possible. Thus, "...the limit must be accepted as a hypothesis, not as a number that can be determined experimentally." [31]

The axiomatic definition is formulated by A.N. Kolmogorov and is based on the following three postulates:

$$P(A) \leq 0, \tag{2.5}$$
$$P(S) = 1, \; and \tag{2.6}$$
$$P(A \cup B) = P(A) + P(B). \tag{2.7}$$

This first postulate states that the probability of event A, in terms of random variables for example $\{X \leq x\}$ or $\{X = x\}$, is a non-negative number. The second defines that the probability of the certain event, which is the sample space S, equals 1. And the third one that if the probability that two events A and B are mutually exclusive, then their probability is the sum of their single probabilities.

Bayesian probability defines probability as the "degree of belief". It denotes the truth of an event given incomplete knowledge. A prior state of knowledge is given before the experiment is performed. The posterior information (after the experiment) can be expressed by the Bayes' theorem as the posterior probability. There are two different interpretation of the Bayesian probability. The "objective" derives probability in a logical form based on the Cox theorem that postulates three assumptions: (1) Divisibility and comparability, (2) Common sense, (3) Consitency. The "subjective" interpretation is based on the personal degree of belief an individual has in the truth of the proposition.

2.1.3 Distribution and Density

Probability depends on a function called Cumulative Distribution Function (CDF) $F_x(x) = P(X \leq x)$. The Probability Density Function (PDF) is defined as the derivative of the distribution function as $p(x) = \frac{dF_x(x)}{dx}$. Thus, the PDF describes the probability [32]

$$P(x) = \int p(x)dx. \tag{2.8}$$

A common density function is the normal distribution with mean μ and variance σ^2. The PDF is given by the Gaussian function [44]

$$p(x) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right\}. \tag{2.9}$$

The equation assumes that the random variable X, respectively x is a scalar value on one-dimension. For multi-dimensions vector the normal distribution is called multivariate. Following Thrun et al. [44] it is characterized by the following form

$$p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}. \tag{2.10}$$

Σ denotes a positive semidefinite and symmetric matrix called the covariance matrix. The vector μ represents the mean. Because of its major role, normal distributions are often abbreviated as $\mathcal{N}(x;\mu,\sigma^2)$ or $\mathcal{N}(x;\mu,\Sigma)$. As postulated in (2.6) the PDF integrates always to 1 ($\int p(x)dx = 1$).

The joint distribution combines the occurrence of two events mapped onto two random variables. $P(x,y)$ denotes the joint probability event that random variable X and Y appear concurrently. In case of statistical independency the probability and density is given by

$$P(x,y) = P(x)P(y),\ and \tag{2.11}$$

$$p(x,y) = p(x)p(y). \tag{2.12}$$

2.1.4 Conditional Probability and Bayes' Theorem

Conditional probability reflects the fact that the probability of an event may depend on a second event [32]. For two different random variables X and Y it is defined as

$$P(x|y) = P(X=x|Y=x) = \frac{P(x,y)}{P(y)},\ and \tag{2.13}$$

$$p(x|y) = p(X=x|Y=y) = \frac{p(x,y)}{p(y)}. \tag{2.14}$$

Note that the term for the density function $p(x|y)$ only show the discrete case for two single events x and y. The term for point conditioning density function and interval conditioning is in more detail explained by Peebles [32] and is not further considered. The Bayes' theorem is derived from the conditional probability and can be written

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} = \frac{P(y|x)P(x)}{\sum_{x'} P(y|x')P(x')} \quad \text{(discrete)},\ \textit{and} \tag{2.15}$$

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} = \frac{P(y|x)P(x)}{\int P(y|x')p(x')dx'} \quad \text{(continuous)}. \tag{2.16}$$

$P(x)$ is denoted as prior probability distribution whereas $P(x|y)$ is called the posterior probability distribution over X, $P(y|x)$ the inverse probability and the event y is related in robotics to the acquired measurement data. To determine the posterior distribution, we are evaluating the inverse probability, which specifies the probability of the measurement data y assuming that a certain event x was true [44].

If we introduce another random variable Z with the given values $Z = z$ we can formulate the Bayes' theorem as

$$P(x|y,z) = \frac{P(y|x,z)P(x|z)}{P(y|z)}. \tag{2.17}$$

Considering the conditional independence $P(x|z) = P(x|y,z)$ and $P(y|z) = P(y|x,z)$ the equation can be written as $P(x|y,z) = P(x|z)$.

2.2 Environment Representation

The mobile robot is surrounded by the real world. To operate safely, it needs to gather knowledge about its environment. It does this by using an abstract model describing the interaction between itself and the environment.

This section defines some basic characteristics of the robot and its environment. The state, control and measurement are introduced as some of the basic terms. Based on that a structured dynamic system is defined explaining the robot interaction with its environment. Finally, the definition of the posterior probability, it is derived from the Bayes' theorem as denoted in Sec. 2.1.4, and represents the robots belief about the environmental state.

2.2.1 State, Control, Measurement

In mobile robotics the robot and the environment is described by its state. A state can be either static, such as the location of walls, or dynamic such as moving people in a room [44]. The robot postion and orientation is considered as a dynamic state. This is commonly known as the robot's pose (if it is given relative to a global coordinate framework). The size of the state is not limited in its pose dimension. Location of objects in the environment, robot velocity or odometry and configuration of robot's actuators can be included into the state variables [44]. The state variable x_t is time dependent, with time discrete steps $t \in \{0, 1, 2, 3, \dots\}$.

Control actions are changes to the robots environment. During the interaction with the real world the robot performs different actions. For example, moving from one location to another or transporting an object from on postion to another. Therefore control data u_t denotes information about the change of state in the environment with the control sequence $u_{t_1:t_2} = u_{t_1}, \dots, u_{t_2}$ [44].

Senor data processing is used by the robot to obtain information about its environment. For example, range scan or camera image data is acquired to perform distance measurements. This step is called observation or perception. Following measurement data is denoted as z_t with the measurement $z_{t_1:t_2} = z_{t_1}, \dots, z_{t_2}$ [44].

2.2.2 Dynamical System

In theory, the interaction between the robot and it's environment is described as a dynamic system [35]. It can be viewed as two independent automatons as depicted in Fig. 2.1. The environment automaton, which obtains input or actions and transforms them into output signals. The controller automaton (robot), that gathers the environment output to perform actions.

We now divide the environment into an environment and sensor process. The more commonly used terms are system or motion model and measurement or observation model [35]. The environment process changes and updates the state of the environment, according to the input actions. The sensor process maps the current state of the environment into an output information that can be received by the controller. Likewise the controller consists of a state estimation and an action selection process. Commonly called inverse observation or measurement model and action selection model [35]. When receiving a measurement input the process tries to estimate $\hat{x}_t$ the state of the environment. Once performed the action selection process uses the estimation $\hat{x}_t$ to derive the output actions u_t.

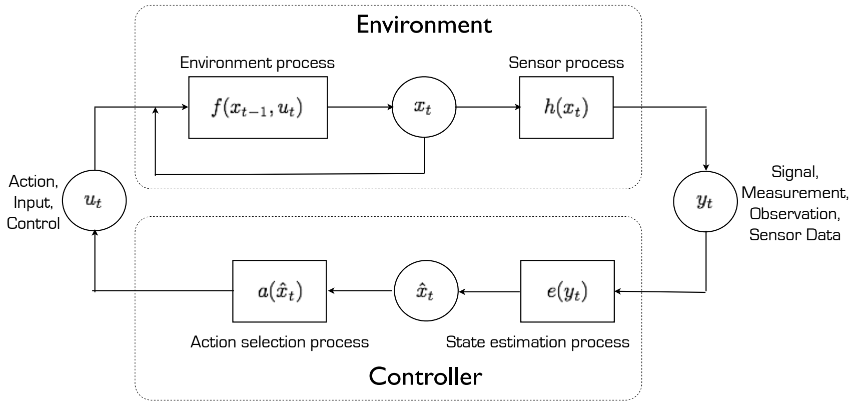


Figure 2.1: State estimation model as used in structured dynamical systems [5].

Formally, following mathematical notation is considered:[1]

$$x_t = f_t(x_{t-1}, u_t, v_t) \quad \text{(system model)}, \tag{2.18}$$
$$z_t = h_t(x_t, w_t) \quad \text{(measurement model)}, \tag{2.19}$$
$$\hat{x}_t = e_t(z_t) \quad \text{(inverse measurement model), and} \tag{2.20}$$
$$u_t = a_t \hat{x}_t \quad \text{(action selection model).} \tag{2.21}$$

2.2.3 Belief Representation

The belief indicates the robot's internal knowledge about the state of the environment [44]. The robot infers its belief about its postion and orientation (pose) indirect from measurement data. Therefore the posterior probability or density function is assigned, which conditions all measurement, control data and hypothesis over the robot state. We denote the belief as

$$bel(\hat{x}_t) = p(\hat{x}_t | z_{1:t}, u_{1:t}). \tag{2.22}$$

Further explained by Siegwart et al. [37] it can be distinguished between a single-hypothesis $bel(\hat{x}_t)$ and multiple-hypothesis belief $bel(\hat{x}_t|\mathcal{H})$. A single hypothesis $\mathcal{H}$ represents the direct possible proposition. For example the belief of a single-hypothese about the robot position is represented as a single unique point on the map. Whereas a multiple-hypothesis belief has more than one possible representation. Consequently there is more than one hypothesis about the robot postion [37].

It is worth noting that the belief distributions are based on the Bayesian objective definition of probability using the Bayes' theorem to determine its value as highlighted in Sec. 2.1.4.

[1]Note that process noise v_t and measurement noise w_t are commonly added as the model is assumed to be stochastically.

2.3 Basic Filter Algorithms

The following section introduces the subject, how the robot can calculate its state by using the method of belief representation as explained in Sec. 2.2.3. It starts with the characterization of the Bayes Filter as the most general algorithm for calculating the belief distribution [44]. The main features are the prediction and update step, restricted by the Markov assumptions defining the boundary conditions of the filter performance. The following Kalman Filter, invented by Swerling (1958) and Kalman (1960) shows the most common implementation of a Bayes Filter. The Kalman Filter presumes that all processes are linear and thus the probability density distributions are following a gaussian distribution as introduced in Sec. 2.1.3. Further it is only applied to continuous state random variables and not to discrete ones [44].

The concept of Kalman gain and innovation defines the main advantage of this filtering method. Its major downside is that it is for real problems not applicable, because system and observation models derived from the real world are in most cases not linear. E.g. a robot moving with constant translational and rotational velocity follows a trajectory, which can not be described by a linear model [44]. The Extended Kalman Filter (EKF), e.g. proposed by Jazwinski (1970), tries to overcome this by linearization that finally determines the performance of the EKF [35].

It is worth noting that further variants to the Bayes Filter exist like the Iterated (nonlinear) Kalman Filter, the Unscented (nonlinear) Kalman Filter (UKF), the Multiple Hypothesis Kalman Filter and Monte Carlo Bayes Filter, which are explained in further detailed in e.g. Thrun et al. [44] and Schmitt [35].

2.3.1 Bayes Filter

The idea for this filter is based on the recursive state estimation. The state is considered as a estimated belief, modeled as probability density like in Eq. 2.22. Each calculation step of the algorithm is time discret using previous belief and measurement data as input derived from the observed environment state. The Bayes Filter calculates then the current belief $bel(\hat{x}_t)$, from the prior belief $bel(\hat{x}_{t-1})$, based on the input from measurement z_t and action u_t. For the next state estimation $bel(\hat{x}_{t+1})$ the previous calculated belief $bel(\hat{x}_t)$ is used as the new input for the next iteration step.

The Bayes Filter algorithm requires an initial belief $bel(\hat{x}_0)$ [44]. If not known it can be initialized by using a uniform distribution. Considering how the algorithm perform its calculations, we derive the following main steps [44]: (1) A prediction step based on the prior state belief $bel(\hat{x}_{t-1})$ and the probability that the robot takes control step u_t as next action from state $\hat{x}_{t-1}$ to $\hat{x}_t$. (2) A measurement update that weights the estimated prediction by the probability that the observation z_t may be true. This is obtained [44] from the following two equations:

$$\overline{bel}(\hat{x}_t) = \int p(\hat{x}_t|u_t, \hat{x}_{t-1})\ bel(\hat{x}_{t-1})\ d\hat{x}_{t-1} \quad \text{(prediction step), and} \tag{2.23}$$

$$bel(\hat{x}_t) = \eta\ p(z_t|\hat{x}_t)\ \overline{bel}(\hat{x}_t) \quad \text{(update step).} \tag{2.24}$$

As pointed out by Schmitt [35] the conditioned probability density $p(\hat{x}_t|u_t, \hat{x}_{t-1})$ in Eq. 2.23 is derived from the environmental process as defined by Eq. 2.18 and the known statistics of the process noise v_t. Hence, it refers to the system or motion model as introduced in Sec. 2.2.2, and can be denoted as a state transition probability/density as noted by Thrun et al. [44]. Further the density $p(z_t|\hat{x}_t)$ (measurement probability/density [44]) derived from Eq. 2.19 and the measurement noise w_t, can be considered to refer to the observation or measurement model in this context.

Bayes filtering requires that the process follows the Markov assumption. It postulates that the past and future data is independent if one knows the current state x_t [44]. Thus, a state $\hat{x}_t$ is called complete if all the past measurements and controls have no influence on future states $x_{t+1,t+2,...}$. In practice this assumption is violated and therefore mainly theoretically important. In this case a state is called incomplete. For example not considered dynamics of moving objects have an effect on the future measurement statistics or unmodeled control variables, which may influence the robot target pose.

2.3.2 Kalman Filter

The Kalman Filter assumes that the observed system is a linear Gaussian system. Therefore it represents the beliefs about the estimated environment state $\hat{x}_t$ as multivariate normal distribution $\mathcal{N}(x_t; \mu_t, \Sigma_t)$ as defined in Eq. 2.10. Furthermore refering to Eq. 2.18, 2.19, the system and observation model is expressed by the following linear equation [35] as

$$x_t = A_t x_{t-1} + B_t u_t + v_t \quad \text{(system model), and} \tag{2.25}$$

$$z_t = C_t x_t + w_t \quad \text{(measurement model).} \tag{2.26}$$

Where A_t and B_t are matrixes that describe the system behaviour with dimension $n \times n$ and C_t describes the measurement system with dimension $k \times n$. Size n denotes the dimension of the state vector x_t, whereas k the dimension of the measurement vector z_t. The Gaussian random variables x_t, u_t, z_t, v_t, w_t represents the state, action respectively the system and measurement noise [35].

Derived from this linear model, and the presence of Gaussian noise, the Kalman Filter algorithm represents the belief $bel(\hat{x}_t)$ at time t by the mean μ_t and the covariance Σ_t [44]. Likewise the Bayes Filter, is the input to the Kalman Filter the previous belief $bel(\hat{x}_{t-1})$ represented by μ_{t-1} and Σ_{t-1}. Further the control u_t and measurement z_t are required to calculate the state transition density $p(\hat{x}_t|u_t, \hat{x}_{t-1}) \sim \mathcal{N}(\hat{x}_t; A_t\hat{x}_{t-1} + B_t u_t, R_t)$ and measurement density $p(z_t|\hat{x}_t) \sim \mathcal{N}(z_t; C_t\hat{x}_t, Q_t)$. Considering the two main steps prediction and measurement update the algorithm and hence the belief can be derived as

$$\left.\begin{aligned} \overline{\mu}_t &= A_t \mu_{t-1} + B_t u_t \\ \overline{\Sigma}_t &= A_t \Sigma_{t-1} A_t^T + R_t \end{aligned}\right\} \overline{bel}(\hat{x}_t) \quad \text{(prediction step), and} \tag{2.27}$$

$$\left.\begin{aligned} K_t &= \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1} \\ \mu_t &= \overline{\mu}_t + K_t(z_t - C_t\overline{\mu}_t) \\ \Sigma_t &= (I - K_t C_t)\overline{\Sigma}_t \end{aligned}\right\} bel(\hat{x}_t) \quad \text{(update step).} \tag{2.28}$$

As pointed out by Thrun et al. [44], the predicted mean $\overline{\mu}_t$ is obtained by inserting the previous mean μ_{t-1} in the deterministic version of Eq. 2.25. The predicted covariance $\overline{\Sigma}_t$ depends on the previous state represented by Σ_{t-1}, which is transformed to the current state by the system matrix A_t and the covariance R_t, that defines the state transition density as explained in Sec. 2.3.1.

In the measurement update step the measurement z_t is incorporated into the predicted belief $\overline{bel}(\hat{x}_t)$ to obtain the final belief $bel(\hat{x}_t)$ [44]. Therefore the Kalman gain K_t is calculated. It weights the measurement that influences the new estimation. The desired belief distribution represented by μ_t and Σ_t is finally obtained by adjusting the predicted covariance $\overline{\Sigma}_t$ and mean $\overline{\mu}_t$. The goal is to calculate the difference between the actual measurement z_t and the expected measurement $C_t\overline{\mu}_t$ or $C_t\overline{\Sigma}_t$. This step is called innovation. To obtain the next state estimation $bel(\hat{x}_{t+1})$ the iteration starts again with the prediction step. Thus, the previous calculated belief $bel(\hat{x}_t)$ is used as the new input.

2.3.3 Extended Kalman Filter

The Extended Kalman Filter (EKF) upgrades the Kalman Filter to a nonlinear Gaussian system. System (cf. Eq. 2.25, 2.18) and measurement (cf. Eq. 2.26, 2.19) model are now defined as nonlinear functions $g(u_t, x_{t-1})$ and $h(x_t)$. The key idea of the EKF is the approximation of $g(u_t, x_{t-1})$ and $h(x_t)$ by the Taylor series, which is the first-order approximation derivative. This method is called first order Taylor expansion. The prediction and measurement update are now defined by

$$\left.\begin{aligned} \overline{\mu}_t &= g(u_t, \mu_{t-1}) \\ \overline{\Sigma}_t &= G_t \Sigma_{t-1} G_t^T + R_t \end{aligned}\right\} \overline{bel}(\hat{x}_t) \quad \text{(prediction step), and} \tag{2.29}$$

$$\left.\begin{aligned} K_t &= \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1} \\ \mu_t &= \overline{\mu}_t + K_t(z_t - h(\overline{\mu}_t)) \\ \Sigma_t &= (I - K_t H_t)\overline{\Sigma}_t \end{aligned}\right\} bel(\hat{x}_t) \quad \text{(update step).} \tag{2.30}$$

The predicted mean $\overline{\mu}_t$ is now approximated through the first term of the Taylor expansion at the posterior mean μ_{t-1}. That is, $g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1})$ [44]. Considering the predicted covariance $\overline{\Sigma}_t$ the linear matrix A is replaced by its nonlinear equivalent. Further the Jacobian G_t (size $n \times n$, state dimension), which corresponds to the derivation of the nonlinear system function at state $x_{t-1} = \mu_{t-1}$ (thus, $G_t = g'(u_t, \mu_{t-1})$).

Comparing the update step of the EKF with the Kalman Filter, the linear measurement function is replaced with the approximation of its non linear pendant $h(x_t) \approx h(\overline{\mu}_t) + H_t(x_t - \overline{\mu}_t)$ [44]. $H_t = h'(\hat{\mu})$ denotes the Jacobian matrix, that corresponds to the linear measurement matrix C_t.

2.4 Motion

The section starts with an introduction to a common definition of the robot's motion model. We explain the uncertainty about the robots pose and how it is modeled in probabilistic terms. An important insight is to show that multible robot movements reduce the uncertainty of the robots pose.

2.4.1 Motion Model

In mobile robotic the motion model can be infered from the dynamic system, as introduced in Sec. 2.2.2. The interaction with the environment is described through the environmental process with respect to the robot's motion. It describes the dependency of the state variable x_t on the robots locomotion. The robot action (cf. Fig. 2.1) performs a movement step that changes the pose of the state x_t, in an environment that is assumed to be rigid.

Kinematics is describing the effect of this control actions onto the robot state, respectively the pose in an analytic mathematical form. As stated by Thrun et al. [44], the pose for a rigid mobile robot is commonly described by six variables, its three-dimensional Cartesian coordinates x, y, z and its three Euler angles Ψ, Θ, Φ (yaw, pitch, roll). To simplify matters, it is assumed that the robot is restricted to operate in a plane. The pose (x, y, Θ) is reduced to two-dimensional planar coordinates relative to an external coordinate frame, along with its angular orientation [44].

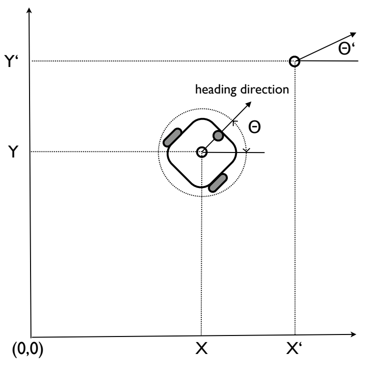


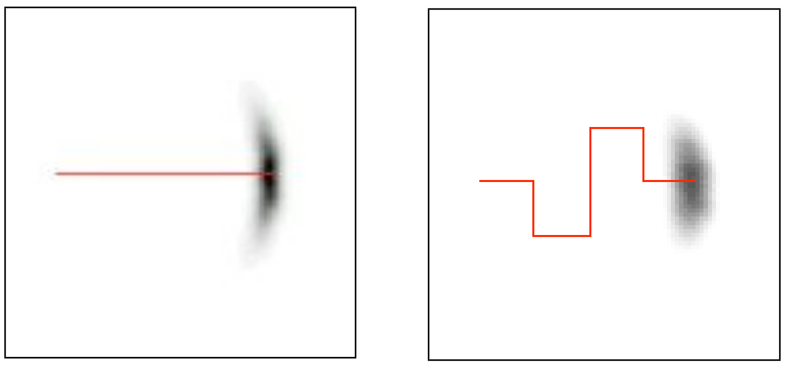
Figure 2.2: Robot current (x, y, Θ) and next (x', y', Θ') pose, shown in a global coordinate system.

To determine motion parameters the robot action is commonly considered as velocity or odometry model. While the velocity is defined by translational and rotational variables $(v_t, w_t)^T$, the odometry comprises wheel information. With odometry it is possible to derive the robot pose relative to its starting point based on the integrated wheel rotation, respectively wheel encoder. The main difference between this two models is that odometry information (such as traveled distance) is only available after executing the action, where as the given speed is adjusted by the given control command. In the following, only the velocity motion model is considered. Due to the missing relevance for this thesis, the description of the odometry motion model is referred to [44].

2.4.2 Motion Probabilistic

The probabilistic motion approach considers that the outcome of a action implicates a certain degree of uncertainty, due to the control noise or unmodeled external effects. The deterministic motion model is enhanced by adding noise variables, that models the uncertainty. As noted in Sec. 2.3.1 the system, respectively the motion model can be defined in probabilistic terms as state transition density $p(\hat{x}_t, u_t, \hat{x}_{t-1})$. Here it states x_t and x_{t-1} defining the robot poses and $u_t = (v_t, w_t)$ the motion control that is assumed to change the robots velocity.

Real robot motion differs from deterministic control commands due to inaccurate execution. This additive errors are modeled with zero-mean Gaussian distribution. Fig.2.3 shows two different robot movements and the probability distribution of the robot's pose after some control actions.



(a) (b)

Figure 2.3: Two posterior pose distributions provided by [44]. (a) Forward movement and (b) more complicated motion with a larger spread of uncertainty. Darker poses are more likely. The plot has been projected into a 2D plane. It is worth noting that in (b) the vertical uncertainty is smaller than as for (a). This is due to the fact that the different control steps reduce the vertical uncertainty

2.5 Perception

The Measurement Model as introduced in Sec. 2.2.2 designs the process of generating sensor data in the real world. The section begins with a short overview of a variety of different sensor methods, such as range sensors, camera, or tactile sensors. The type of the sensor has an influence on the characteristics of the measurement model. An image is described by projective geometry best, whereas sonic models reflection on surfaces in the environment [37].

In the following section a short introduction into the probabilistic of perception is provided. Measurements always have errors, therefore uncertainty associated with them needs to be defined in probabilistic terms. Next, feature extraction is explained as useful methode for object recognition. It reduces a large quantity of measurement information to detectable landmarks. As an example for a vision based sensor the camera sensor is explained. Finally the Time-of-Flight concept is introduced as a prominent sensor approach for distance measurement.

2.5.1 Sensors

There is a large selection of senors available in mobile robotics. They can be divided into two main groups [37]. Sensors that are measuring internal values like, motor speed, wheel postion or robot temperature and external senors gathering information like object distance or intensity. External senors are used by the robot to extract information about its environment and objects respectively features.

There exist various types of external sensors with different physical properties. For example, laser range finder or sonic sensors, which perform an active range measurement. Vision based sensors like CCD/CMOS cameras, that perform a passive measurement are applicable for object recognition and object tracking. A more comprehensive overview about sensors in mobile robotic can be found in Siegwart et al. [37]

2.5.2 Perception Probabilistic

Perception data are imprecise due to measurement noise introduced by the sensor. The measurement process, confer Eq. 2.19, is general assumed as Gaussian distribution. Measurement data consists of more than one numerical value. For example, cameras generating an entire array of values (intensity, color per pixel). Following Thrun et al. [44], the measurement values and probability densities, which are assumed to be independent, are extended to Sec.2.2.1 as

$$z_t = \{z_t^1, \ldots, z_t^K\}, and \tag{2.31}$$

$$p(z_t|\hat{x}_t) = p(z_t^1|\hat{x}_t) \cdot p(z_t^2|\hat{x}_t) \cdot \ldots \cdot p(z_t^K|\hat{x}_t). \tag{2.32}$$

2.5.3 Feature Extraction

Feature extraction is commonly used for map building and localization tasks. Uncertainty is always associated with Sensor measurements. To minimize it, features are extracted from the environment. Features are structures of elements in the environments that can be recognized by processing the sensor data [37], e.g. lines, circles (low level) or edges, doors and tables (high level). This abstracted information can be incorporated easily into the robots's system or motion model to calculate the next selected action. By contrast, raw measurements are direct applied to the model and therefore infer directly unprocessed actions. For example, close obstacles, avoidance tasks or an emergency stop.

In case of vision based systems, features are extracted only from subregions of the image [37]. This is commonly used due to mobile robots requirement of the real time processing. Thus, the amount of data is reduced by the feature detection algorithms. A feature can be represented by a feature vector [44] as

$$y(z_t) = \{y_t^1, y_t^2, \ldots, \} \tag{2.33}$$

with its uncertainty given by the posterior measurement density

$$p(y_t|\hat{x}_t) = p(y_t^1|\hat{x}_t) \cdot p(y_t^2|\hat{x}_t) \cdot \ldots \cdot p(y_t^K|\hat{x}_t). \tag{2.34}$$

2.5.4 Camera as Sensor

Camera as a vision-based sensor is a common passive sensoring device. The image acquisition is based on CCD (Charged Couple Device) and CMOS (Complementary Metal Oxide Semiconductor) senors. The basic idea of both methods is that for light-sensitive elements (pixels) incoming photons free electrons for each element of the array. Integrating the total numbers of electrons over a period of time results in a certain amount of electric charge per pixel. It finally determines the intensity or colour of the incoming light and thus the digital image of the scene. There are certain differences and limitations due to that kind of used senors. First, CCD are more sensitive than CMOS devices. Second, CMOS chips are cheaper in production, much simpler in their design and therefore less power consuming.

Both technologies are limited through their dynamic range, and light capture parameters e.g. iris position or shutter time. The dynamic range specifies the electron capacity per pixel, thus their sensitivity, whereas maximum, minimum iris postion and shutter speed regulate the amount of incoming light. Camera output is inherently digital and therefore easily transfered through common standard transport protocols as IEEE 1394 (FireWire) and USB (1.1, 2.0) onto preprocessing robot hardware. A more detailed description about that topic can be found in Siegwart et al. [37].

2.5.5 Time-of-Flight Principle

The Time-of-Flight (ToF) principle uses the propagation of electromagnetic waves or sound waves [37] to measure the distance of perceived objects. A source emits a wave that is reflected back from the target object. The receiving sensor then measures directly or indirectly the runtime of the signal. ToF with sonic sensor is based on the propagating of sound waves. The distance is directly calculated by the measured time of flight and the propagation speed of sound [37]. The low propagation speed of sound compared to light

($\sim 300\ vs.\ 300 \cdot 10^6\ m/s$), restricts the robot locomotion to certain speed. The maximum speed is reached, when the robot has just enough time to detect and process the signal to avoid obstacles. Systems based on Laser light or light-emitting diodes (LED) are not limited by this.

According to Siegwart et al. [37] three measuring ToF principles can be applied. (1) The direct measuring of the runtime of the light beam, e.g. when a pulse laser is used. The need of a receiver that works at a time resolution in picosecond is a downside of this. (2) Measuring the beat frequency of a frequency modulated signal, emitted by the sensor source and its received reflection. (3) Measuring the phase shift of the reflected light that is perceived by the sensor. Only the latter method in the following is considered due to its relevance to the sensor used in this thesis. As shown by Oggier et al. [28], the indirect calculated distance is proportional to the phase difference ϕ and the modulation wavelength λ. Eq. 2.35 and 2.36 depicts the proportional relation between the distance and the derived signal properties.

$$D = L\frac{\phi}{2\pi} \tag{2.35}$$

$$L = \frac{c}{2 \cdot f} = \frac{\lambda}{2} \tag{2.36}$$

Where L is the non-ambiguity distance defined by the modulation frequency and the speed of light. The factor 2 in the denominator considers the fact that the traveled distance comprise the round-tip that is twice the object distance.

2.6 Localization and Mapping

Localization and mapping enables the robot to navigate in its environment. It can be treated as different tasks, where for localization the map is given and the map building can be considered independently. On the other hand, if the robot tries to navigate in an unknown environment with unknown pose and map, the localization and mapping needs to be solved simultaneously.

This section starts with the introduction to the localization problem. Further the section Map Representation introduces the concept of feature based maps, landmarks and correspondence variables. Following that, the Markov Localization is shortly mentioned as the basis of the following Gaussian filter applied to the localization process. The EKF explains how this particular Gaussian algorithm can be used to determine the robot's pose, using a given map. This leads to the next section where the EKF is applied to the the more advanced Simultaneous Localization and Mapping problem (SLAM).

For the relevance of this thesis only EKF and online SLAM with EFK has been introduced. Other filter solutions such as UKF Localization, Multi-Hypothesis Tracking, Occupancy Grid Mapping, FastSLAM, GraphSLAM can be found in Thrun et al. [44].

2.6.1 Localization

The mobile robot localization problem is derived from the general question of how to determine positons of objects relative to an initial coordinate framework. We assume a model that tries to determine the pose of the robot relatively to a given map of the environment. As Thrun et al. points out [44], there exists certain difficulties when performing localization. Sensor readings are noisy and in some cases ambiguous assigned to distances or objects. For example, look-alike rooms in a building may not be able to distinguish with one sensor reading. Therefore the robot needs to integrate data over time to determine its pose [44].

Position tracking can be easily addressed by knowing the initial pose. Uncertainty is introduced by motion which has usually a small effect on the pose in case of smooth movement. The pose error is assumed to be small and thus the belief is represented by a narrow unimodal gaussian distribution.

In case of global localization, the initial pose is unknown. The problem is more difficult to address since Gaussian distributions are usually inappropriate to address it [44]. Another difficulty is that surrounding environment can contain moving objects, e.g. walking people or moving doors, whereas in a static environment the only moving object is considered to be the robot itself. An important insight of this is the fact that the robot activities can minimize localization errors [44]. If a robot only passively observes the environment but not optimizes its control action, it takes no advantage of the different driving paths with less expected errors.

It is worth noting that the following discussed EKF algorithm is considered to perform a passive localization, because it does not generate robot control actions [44].

2.6.2 Map Representation

Maps can be represented in two different ways. Feature based maps present a sparse sample of objects that have been identified as features, which are represented by an index in the map. Location based maps, index the measured locations to the corresponding index of the map. Both representations can formally be written as $m = \{m_1, m_2, \cdots, m_k\}$. In the following we will only refer to feature based map representation.

A feature can be represented by its distinct object, commonly called landmarks. Each feature can be identified by its pose and a signature s. Due to the imprecise sensor measurement the obtained landmarks cannot be uniquely identified [44]. The definition of a correspondence variable $c_t = \{c_1, c_2, \cdots, c_k\}$ between feature y_t^i and landmark m^i help to uniquely identify the observed feature. It is worth noting that c_t can be determined in two ways. Firstly by reducing the uncertainty by using identical landmarks due to system design and second through estimations via maximum likelihood techniques [44].

2.6.3 Markov Localization

The derived implementation of Bayes Filter algorithm is called Markov Localization. Due to the fact that a given map m is prerequisite, it is incorporated into the measurement model $p(z_t|\hat{x}_t, m)$. For the position tracking an initial belief $bel(\hat{x}_0)$ is required. In practice a Gaussian distribution is used, small centered around the initial pose $\hat{x}_0 = \overline{x}_0$. For global localization the unknown belief is assumed to be uniformly distributed.

2.6.4 EKF Localization

The EKF algorithm approaches the localization problem by assuming that the map is represented by a collection of features. For further discussion the landmarks are assumed to be uniquely identified and there is no data association ambiguity. Thus, the identity of a feature can be expressed by its correspondence variable c_t^i [44].

The filter input can than be denoted as: Previous calculated mean μ_{t-1} and covariance Σ_{t-1}, known feature correspondence vector c_t and feature set y_t, respectively in measurement notation z_t. Further it requires a control step u_t and a feature based map m. The filter calculates the new estimated robot pose as mean μ_t and covariance Σ_t. Starting with the prediction step the estimated mean $\overline{\mu}_t$ and the covariance matrix $\overline{\Sigma}_t$ of the state vector are defined and approximated based on the motion model (cf. Sec. 2.4.1) of the state transition probability. Unlike the EKF as explained in Sec. 2.3.3 is the measurement update performed per obtained feature. For each feature a measurement prediction is computed from the measurement model (cf. Eq. 2.19). The predicted measurement $\overline{z}_t$ is calculated from the relative position and orientation of landmark and predicted mean $\overline{\mu}_t$. The innovation vector is defined as the difference between the observed and predicted measurement (cf. Sec. 2.3.2). In the following update, respectively correction step the Kalman gain is weighting the innovation vector and covariance. Thus, the Kalman gain scales the innovation vector based on the measurement prediction uncertainty [44]. The predicted mean and covariance are corrected by this calculation. The higher the Kalman gain the stronger the resulting correction. The lower the less certain is the observation. For more then one feature the measurement prediction and update steps are recursively repeated with the previous calculated mean and covariance. The more features are observed in one state transition step, the better the final corrected mean and covariance, due to the additional measurement information. For a complete derivation of the EKF localization filter, the referred literature is recommended [44].

2.6.5 EKF Simultaneous Localization and Mapping

A robot that is placed in an unknown environment faces the problem that it has no information about pose nor its surrounding. To navigate successfully, it needs to build up a map of its environment while simultaneously localizing itself relative to this map [44]. This problem is called Simultaneous Localization and Mapping problem (SLAM).

Considering online SLAM with extended Kalman Filter it can be expanded from the EKF localization approach described before. EKF SLAM uses feature-based maps, and models the robot motion and perception as explained in Sec. 2.4 and 2.5 with Gaussian noise. As described in Sec. 2.6.2 we only consider known correspondence for detected features. One of the key differences is the absence of the map. Due to the map being unknown it needs to be built up incremently and therefore it is included in the state vector. Thus, $X_t = (x_t, m)$ denotes the new state vector. The EKF input remains the same as for the localization case only with the difference that the map is not included. The filter output contains the mean μ_t and the covariance matrix Σ_t of the state vector X_t.

Considering now the algorithm sequence, first the estimated mean $\overline{\mu}_t$ and the covariance matrix $\overline{\Sigma}_t$ of the state variable X_t are calculated. This estimation is only calculated for the pose vector, whereas the mean and covariance of the landmarks are unchanged, along with the pose-map covariances [44]. The following steps iterate through the all observed features. As mentioned in Sec. 2.6.4, each feature has a predicted measurement calculation $\overline{z}_t$ that is performed. Furthermore the Kalman gain and innovation is calculated per observed landmark. In case a new feature is detected the corresponding landmark needs to be initialized. Due to the fact that the Kalman gain contains information for the entire state variables, the measurement update of one landmark updates the pose and all other landmarks. Observing a landmark does not just improve the postion estimate of this very landmark, but that of the other landmarks as well. [44]. As mentioned by Dyran-Whyte et al. [14], the important insight of this is the strong correlations between the different observed landmarks. This means the relative location between the landmark is highly accurate, because the measurement error is common due to the single source (robot). The more landmarks that are observed the more the correlations among them increase [14]. Due to of the strong correlations the measurement update for one landmark propagate back to other landmarks.

Chapter 3

Time-of-Flight Sensor

3.1 General Description

This general section introduces concisely the principle of range image cameras. The Swiss-Ranger SR3000 is briefly described as an implementation for this kind of sensor an overview of important parameters is presented.

3.1.1 Range Image Sensor

The Time-of-Flight (ToF) principe is widespread in the area of laser and ultrasonic sensors. Recent developments have implemented this measurement method into camera sensors. This 3D or range Image (RIM) cameras measure additionally to the light intensity or color the distance. The principle is based on modulated light that is sent out by a signal emitter, partially reflected by an object and mapped onto a sensor array by optics. The important insight here is that for every object point the correspond pixel in the sensor array, calculates [18] the distance information. The key advantage of this CCD/CMOS technology 3D cameras is the concurrent acquisition of a 2D and 3D image in one frame. Further moving parts, as used by laser scanners, are not needed to acquire the scene.

3.1.2 SwissRanger SR3000

The SR3000 was developed by Centre Suisse d'Electronique et de Microtechnique SA (CSEM) in Zurich and is commercially distributed by MESA Imaging AG [24]. It is based on a phase-measuring ToF principle using a light source, that emits continuous amplitude modulated near-infrared (NIR) light in the range of tens of MHZ [30]. The reflected light of the scene is imaged by optical lens onto the 3D sensor. A built in optical band pass filter is placed before the sensor to filter out the received daylight. The combined CCD/CMOS 3D sensor is integrated onto the camera electronic board that controls the sensor, processing the image and transfering the depth map to the computer via a USB communication board [30]. The illumination is based on a set of LEDs mounted symmetrically around the camera lens. For the purpose of cooling, the camera it is equipped with a built in fan. The camera software interface enables it to acquire a 2D and 3D image.

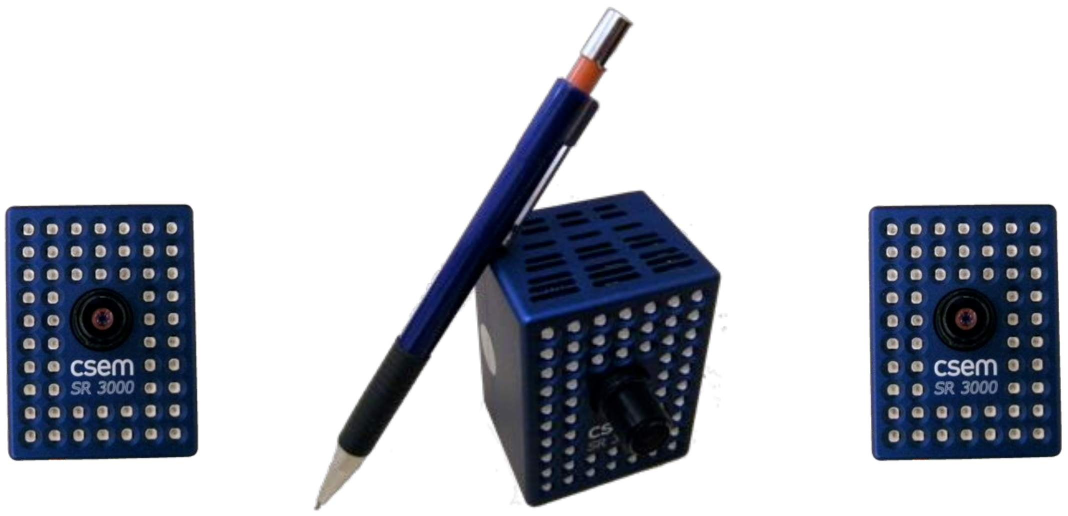


Figure 3.1: Front pictures of SR3000. LEDs are placed around the mounted optical lens. Slots on the top provide a cooling facility.

3.1.3 Specifications

A list of selected parameters of the SR3000 can be found in table 3.1. The list highlights important optical and image processing parameters. A full list of specification can be found in the SR3000 manual [29] and data sheet [23].

Pixel Array Size	176 x 144 (QCIF)
Field of View	47.5 x 39.6 degree
Frame Rate	25fps, typical
Distance Resolution	1% of range, typical
Number of LEDs	55
Illumination Power	1 Watt (average)
Modulation Frequency	20MHz, standard
Wavelength	850nm
Optical Filter Bandwith	35nm (FWHM)
Non-ambiguouse range	7.5m
Focal lenght	8mm
Aperture diameter	5.7mm

Table 3.1: SwissRanger SR-3000 selected parameters.

3.2 Sensor Characteristics

In this section, the sensor specific amplitude sampling is introduced as phase difference measurement method. An important configuration parameter that influences this measurement is the integration time. In the following the range accuracy explains the dependency of the measured distance precision from noise. Beside other noise factors the quantum shot noise is determined as the most influencing factor of the distance accuracy, respectively the measured amplitude value. The section target reflectivity depicts the effect of different received amplitude values of the distance measurement. The last section explains briefly the sensor parameters with respect to distance calibration.

3.2.1 Phase Difference Measurement

Following the general Time-of-Flight principle (cf. 2.5.5), a specific sensor implementation for the phase difference measurement is considered. As explained in 3.1.1 the reflected light signal of an object point is mapped onto an corresponding pixel in the sensor array. The phase difference between the emitted and detected signal is measured by each individual pixel, meaning that each pixel is able to demodulate the incoming modulated light field [28]. This is achieved by taking at least three sampling points (here four) per modulation period to completely reconstruct the received signal [21]. As further pointed out by Lange et al. [21] following the rules of discrete Fourier transform one can calculate the phase, amplitude and offset of the received signal as

$$\phi = arctan \frac{c(\tau_3) - c(\tau_1)}{c(\tau_0) - c(\tau_2)} \quad \text{(phase)}, \tag{3.1}$$

$$A = \frac{\sqrt{[c(\tau_3) - c(\tau_1)]^2 + [c(\tau_0) - c(\tau_2)]^2}}{2} \quad \text{(amplitude), and} \tag{3.2}$$

$$B = \frac{c(\tau_0) - c(\tau_1) + c(\tau_2) - c(\tau_3)}{4} \quad \text{(offset)}. \tag{3.3}$$

Inserting the phase ϕ into Eq. 2.35 provides the measured distance D towards the object. Furthermore the non-ambiguouse range L can be calculated based on Eq. 2.36, which result is $L = 7.5m$ for a modulation frequency of 20MHZ.

As shown in Fig. 3.2 the variables $c(\tau)$ denote 4 samples taken at time $\tau_0 \cdots \tau_3$. The thickness of the sample bars indicate the integration intervall for the collected electrons of that sample. The repeated sampling over one modulation period is necessary to accumulate additional electrons [28]. This is needed to gain a significant number of electrons to represent a meaningful sample value.

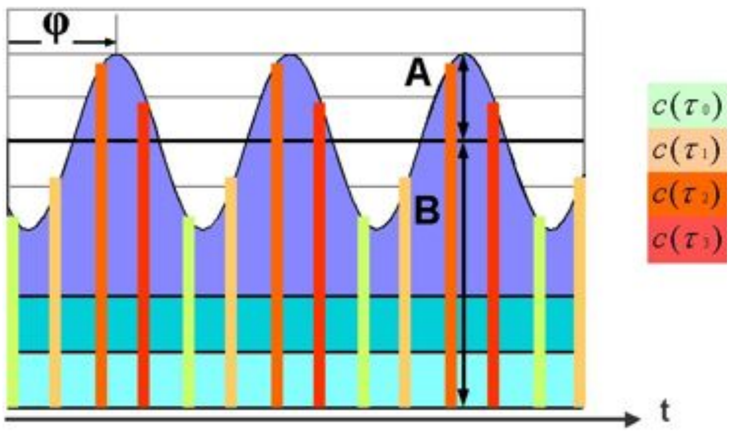


Figure 3.2: Light signal [29] with four repeated samples $c(\tau)$, amplitude, phase shift and offset.

3.2.2 Integration Time

The integration time allows to control the number of measurement cycles taken to accumulate the measured samples $c(\tau)$. This is accomplished per pixel and per specific sample at time $\tau_0 \cdots \tau_3$. As Fig. 3.2 shows each sample can be performed more than once for different phases $2\pi N$.

The integration time can be set in the ranges from [0,255], where 0 correspond to integration time $200\mu s$ and 255 to $51.2ms$ [29]. An increase in the integration time has following effetcs [18]: (1) a better SNR and therefore more precise data, (2) the measured amplitude increase, thus the pixel saturation level is reached faster, (3) lead to a shorter measured distance and (4) the acquisition speed decreases. It is therefore important to find the right adjustment between acquisition speed and distance accuracy.

3.2.3 Range Accuracy

There are several limiting noise factors to the range accuracy of a distance measurement of the 3D sensor. As derived by Lange [20] the essential ones are electronic and optical shot noise, thermal noise, rest noise, $1/f$ noise. The following equation for depth accuracy is given by:

$$\Delta L = \frac{L}{\sqrt{8}} \frac{\sqrt{B + N_{Pseudo}}}{2A}. \tag{3.4}$$

Where A and B denote the demodulated amplitude and offset as calculated in Eq. 3.2 and Eq. 3.3, with respect to the number of photoelectrons per pixel and sampling point. L is the non-ambiguity distance and N_{Pseudo} the noise according to rest, 1/f and termal noise [20]. The term $\frac{\sqrt{B}}{2A}$ compute the phase error $\Delta\phi$ which depends on the number of generated photoelectrons. Lange [20] conclude that the quantum noise can be considered as a limiting factor for the phase error, respectively to the range accuracy. As pointed out by Oggier et al. [28], the offset B can be defined as background and the active modulated illumination offset according to Fig. 3.2. The amplitude A depends on the emitted optical power, modulation depth, the spectral transmission properties (filtering), the camera optics (f-number, aperture ratio), the performance of the charge separation and transport per pixel (demodulation contrast), the target reflectivity and final target distance.

Considering Eq. 3.4 the amplitude A is the most affecting parameter on the distance accuracy. As shown in Fig. 3.3 for small amplitudes the distance resolution, respectively the standard deviation of the amplitude increase due to photon shot noise, due to high values the standard deviation converge to a smaller number. This implies that for long distances, where low optical energy is backscattered the measured distance offset increases [18]

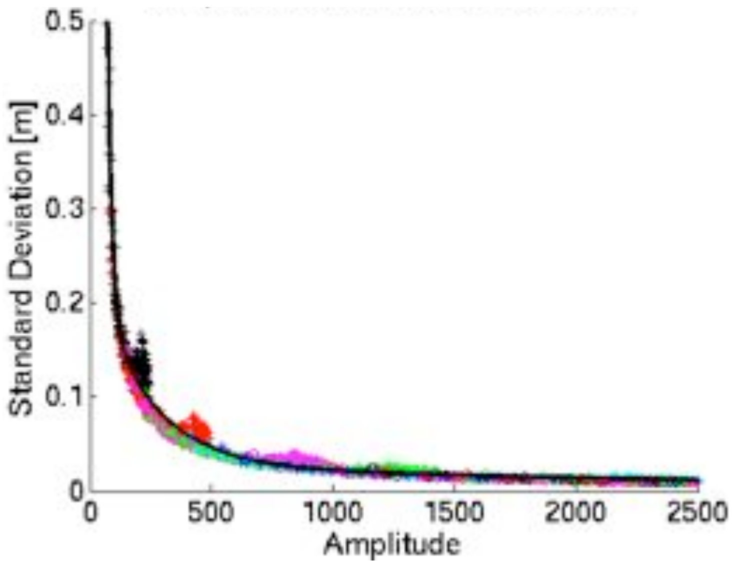


Figure 3.3: Fitted curve into measured amplitude values vs standard deviation. The mean and variance of an amplitude value was calculated per pixel. Afterwards the point was plotted. The different colours indicate a different series of measurement. It includes the effects of light scattering and multiple path reflections, which add additional noise to the amplitude A.

3.2.4 Target Reflectivity and Angle of Incidence

Different target reflectivity, respectively different demodulated amplitudes A have different effect's on the distance measurement. As shown by Kahlmann [18], for a high targets reflectivity the measured distance decrease, whereas increase for low amplitudes. The similar effect occurs with the angle of incidence. A large angle decreases the backscattered engery. Thus, the measured distance become longer. The important inside here is that for a normal reflection to the sensor plane both effect compensate each other to a large degree [18]

3.2.5 Calibration

Camera calibration is used to determine which object coordinates are mapped onto which pixel of the sensor array. This is performed by the a camera model that defines intrinsic and the extrinsic parameters. The simplest and ideal model is the pinhole camera, where the lens is exchanged by a very small hole. The important intrinsic parameters are the focale length and principal point. The extrinsic parameters denote the distortion of the optics, as radial and decentral parameters [18]. The SR3000 provides a calibration file that consider principal point, focal length, pixel size and pixel number for transforming the distance data into cartesian coordinates [1].

Further a fixed pattern noise (FPN) correction is assigned for each pixel. As pointed out by Kahlmann [18], the FPN can be calculated with respect to following procedure. The true distance is measured based on geodetic total station and for every pixel the nominal distance is calculated. A number of camera measurements are used to calculated the difference between the calculated and measured distance. The difference is stored as offset in the calibration file. An important preliminary is that the measured camera distance per pixel is corrected, with respect to the angle of incidence, the target's reflectivity and the distance-to-distance dependencies [18].

[1]The calibration file is only applied in case the provide software API function is used.

3.3 Interferences

The main disturbing effects are explained, with respect to the distance measurement. Scattering and temperature are determined as the most influencing ones. Following that, multiple reflection show the dependency of the scene on the measurement result. Finally the effects of interference on the image acquisition is presented as image noise, motion blur and over saturation.

3.3.1 Scattering

Light scattering is an effect that occurs due to the internal setup of the camera [29]. Not all incoming light that strikes the surface of a 3D sensor array is absorbed. Some part of it is reflected back to the lens. This light is then partially reflected back to the sensor. This can happen several times, thus a pixel also contains information which belongs to neighboring pixels [18].

The result of this effect has a major impact on the distance accuracy. Measurements that are performed by Kahlmann [18] show a distance variation in for range of a few centimetres. For example in case in case of over saturation. The amplitude peak was spread to neighboring pixels, thus a shorter distance was measured. But even pixels that are far away form the reflected target object are influenced. Scattering disturbs the whole measurement process due to its influence on almost all the sensor arrays at once. I [18].

3.3.2 Mixed Pixels

Similarly to the scattering phenomena is the mixed pixel effect. They are well known by range finder applications such as the laser systems. They cause false data within a distance measurement because of overlapping measurement form different targets [18]. For example, if two targets reflect to one and the same pixel, the energy of both combine to one distance information.

Mixed pixel can be differantiated in spatial and amplitude based pixels [18]. Spatial mixed pixels can be explained as a vector addition of two target vectors. The vector length represents the received signal amplitude, the centre angle and the measurement of the demodulated phase difference from the emitted signal. The addition provides the final result of the pixel vector. The amplitude mixed pixel results in a overlapping, neighbouring pixels with a small distance difference. The higher signal amplitude of one pixel exposes the lower amplitude of the neighbour pixel and thus leads to a false distance and amplitude data [18].

3.3.3 Temperature

The sensor is influenced by internal and external temperatures. The internal temperature influences the distance measurement during the a warm up period. The measured amplitude A decreases during the first minutes and converges to a lower value afterwards. Otherwise distance starts to increase and converger to a higher value. In case of external temperature changes the higher the temperature the greater the distance measurement. For room temperature the SR3000 seem to be stable [18].

3.3.4 Multiple Reflections

The problem of multiple reflections or multi path scattering occurs in scenes where the backscattered light from the object has more than one path to propagate back to the sensor [29]. Fig. 3.4 illustrates a corresponding scene. As investigated by Guomundsson et al. [16] can be the corners depth measurement within two walls very distorted. The measured walls are very rounded at the corner and the angle between the walls is not 90°.

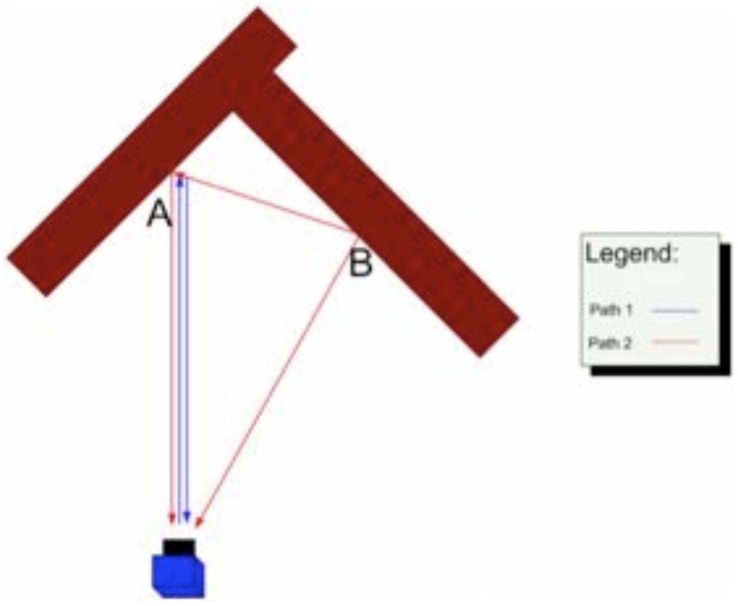


Figure 3.4: The figure [29] depicts schematically the interference from B to A with respect to two different reflection points.

3.3.5 Image Noise, Motion Blur and Saturation

Changing the integration time can lead to one of the following described effects. If the integration time is too small the accumulated amplitude charge is also small. Thus the range accuracy is degraded and the effect of photon shot noise is noticeable (cf. 3.2.3). This can be seen in a noisy amplitude and distance image. By choosing a long integration time the distance accuracy improves (cf. 3.2.3, 3.2.2) and and the images contain less noise. For moving objects this results in a overlay of different object points in one pixel. The objects in the images become blurred. Due to the longer accumulation of amplitude charges the pixel gets saturated faster. Hence, object points with high reflectivity or close object points gets overexposed.

One camera feature is the auto integration time. It adjusts automatically the time that lowers the noise to a sufficient level. The downside is that the frame rate varies to a wide range, depending on the used integration time.

3.4 Comparable Vision Based Systems

A short overview of important parameters is provided to compare the SR3000 with other vision based systems. With respect to the field of mobile robotics, two comparable vison systems are chosen. This optical imaging systems have proven in related works similar implementation to the investigating subject of Vision SLAM (Sec. 4). Figure 3.2 represent typical values with no claim of completeness.

	SR3000	nonocular camera	stereo vison cameras
Measurement Principe	ToF	Intensity	Intensity
Sensor Type	CCD/CMOS	CCD or CMOS	CCD or CMOS
Pixel Array Size	176x144	$>$ 640x480	$>$ 640x480
Field of View	47.5°x39.6°	50°-100° (H)	50°-100° (H)
Frame Rate	25fps, typical	30-100fps	30-60fps
Distance measurement	yes	no	yes, with triangulation
Distance Resolution	cm	non	mm
Non-ambiguouse range	7.5m, typical	no distance	$> 7.5m$
Motion blur	yes	fast movement only	fast movement only
scattering effect	yes	low	low
multipath effect	yes	no	no

Table 3.2: Comparison of SR3000 with monocular cameras and stereo vision systems in the field of Vision SLAM.

Chapter 4

Vision SLAM

In the last decade the mobile robot community has been investigating SLAM in general at a theoretical and practical level. Whereas the theory and concept can be considered as a solved problem, practical solutions suffer from the lack of generality and are mostly restricted to particular implementations and restriction [14]. E.g. slow or smooth robot motion, multiple, specialized and accurate sensors, simple unambiguous landmarks [9]. Specially the building of rich and large feature maps is still a limitation factor for real time processing.

The Simultaneous Localization and Mapping problem has been implemented in various areas. From indoor and outdoor robots to underwater and airborne systems [14]. Most commonly used sensors are laser range finders, sonic and recently to some extent camera sensors. In the following, only the camera is considered as a measuring sensor, which falls into the realm of vision based SLAM. The following sections provide a brief overview of some published works in that area. In particular the MonoSLAM algorithm of Davison [11] is explained in more detail, because of the relevance to the subject of this thesis.

4.1 General Overview

During the time the SLAM problem was theoretically well determined researchers began to expand their implementation in the domain of vision based sensors. One of the first works was presented by Davison and Murray [8], using stereo cameras on an active head mounted on robot with goal-directed navigation. An automatic feature detection, mapping and tracking was presented at a real time processing rate of 5Hz. Davision and Kita [7] extended this method to the presence of unknown slope variations. An alternative approach to Visual SLAM was suggested by Drocourt et al. [13] using omni-directional stereo vision system. Following works of Davison [9], [11] presented an algorithm working with a monocular camera. So far several contributions have been proposed based on the stereo vision cameras e.g. [43], [42], on monocular camera e.g. [15] as well on mixed solutions with camera and other sensors e.g. [19]. More recently published work by Sim et al. [39], [38] used SIFT features in combination with a Rao-Blackwellised Particle Filter (RBPF) to build a large scale map from stereo vision camera and by Milford and Wyeth [26] demonstrating with a single consumer camera mounted on a driving vehicle, localization and mapping using a computationally expensive process of tracking landmarks.

4.2 MonoSLAM

MonoSLAM is a algorithm introduced by Davison et al. [11] based on a sequential approach to map building and localization with a step-by-setup fashion. Among other visual based SLAM methodes its specific characteristic contribution is the feasible usage of a single camera as a sensor, using a EKF based algorithm to calculate the first order estimation of the robot (camera) and map feature positions [9] in a sparse map of visual landmarks. Furthermore it proposes a technique of active feature measurement and mapping, the use of a general motion model for smooth camera movement, and solutions for monocular feature initialization and feature orientation estimation [11].

Being restricted to sparse feature maps rather than dense visual maps, which are computational intensive, the single camera SLAM algorithm can be implemented within real time systems. Possible applications are a single camera used as a sensor for robot navigation, a wearable robot for motion estimation of a device worn by humans to assist in tasks such as industrial inspection, or television used to provide camera motion estimation for on-line augmented reality [9]. In the following a general overview of the MonoSLAM algorithm is presented with no claim to be complete.

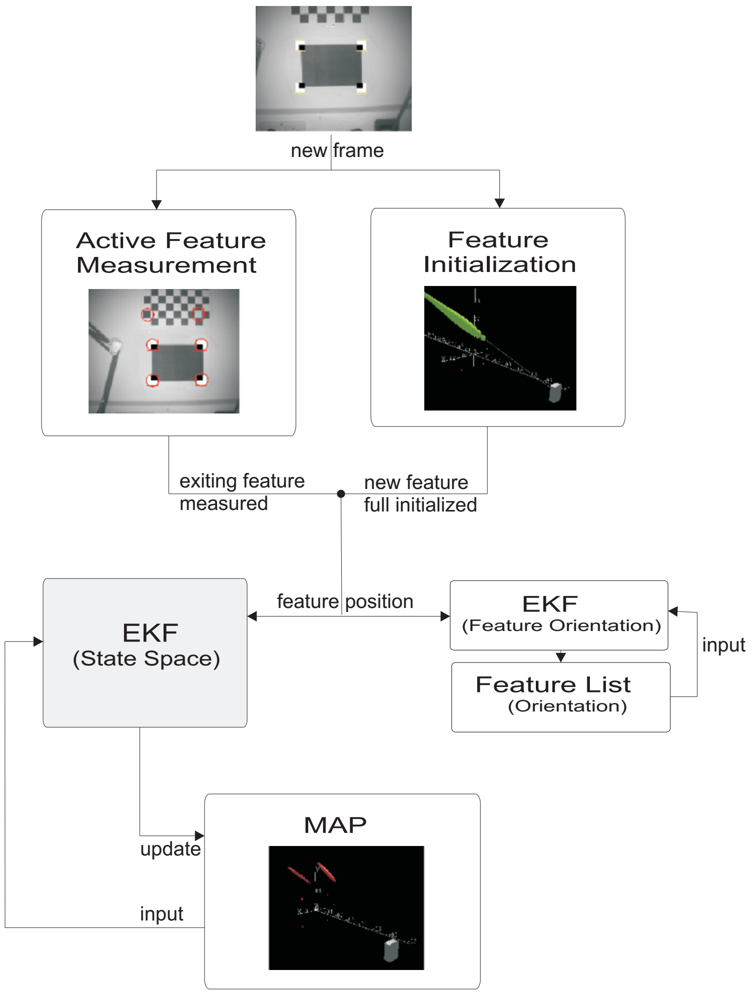


Figure 4.1: Overview of the MonoSLAM algorithm as presented by Davison et al. [11].

4.2.1 Map Representation

Following Davison et al. [11] a map represents the position and orientation of the features and camera position in a 3D coordinate system. The location and bearing are probabilistic estimations and continuously updated during camera motion and feature observation, using a EKF based algorithm to minimize their uncertainty (cf. Sec. 2.6.5, 2.3.3 2.3.2). New features can enlarge the map whereas unstable features can be deleted. In a probabilistic map features can be displayed as points enclosed by a 3D ellipse expressing their uncertainty of the current position. Mathematically the map can be expressed by its state vector $\hat{x}_t$ and covariance matrix P_t for the discrete time step t (cf. Sec. 2.2.1, 2.1.3 and for more detail to [11]),

$$\hat{x}_t = \begin{pmatrix} \hat{x}_v \\ \hat{y}_1 \\ \hat{y}_2, \\ \vdots \end{pmatrix}, \qquad P_t = \begin{bmatrix} P_{xx} & P_{xy_1} & P_{xy_2} & \dots \\ P_{y_1x} & P_{y_1y_1} & P_{y_1y_2} & \dots \\ P_{y_2x} & P_{y_2y_1} & P_{y_2y_2} & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix}. \tag{4.1}$$

Where the estimated camera pose is defined as $\hat{x}_v$ with 3D position, orientation, velocity and angular velocity, and estimated feature state $\hat{y}_i$ denote its 3D position [11]. The covariance matrix consists of a sub matrix for the camera state uncertainty P_{xx} and feature state uncertainty P_{ii}. They are displayed within the 3D map in an elliptic form and updated after every EKF measurement step. The cross correlations between features and camera P_{y_ix} and among feature themselves $P_{y_iy_j}$ are not displayed in the map. It is essential to emphasize that the strong correlations between the different features are an important insight of the general EKF-SLAM problem. Meaning that, the relative feature positions among each other are very well known and thus the covariance matrix $P_{y_iy_j}$ converges towards zero, whereas the absolute position of camera and feature still remain uncertain through their motion and measurement noise (cf. Sec. 2.6.5 and refer for more detail to [14]).

4.2.2 Assumptions

Known Objects

The single camera SLAM as introduced here requires prior information for the system start-up about the surrounding environment. This can be achieved by placing a known object in front of the camera, with four features at known positions and patch [11]. For the start-up the predefined patches, e.g. corners of a black rectangle, are searched for within the image frames at known postions. In case of a match of all four patches the camera and feature position can be considered as initialized. Note that the initial camera positions is given a certain amount of uncertainty, which quickly converge to a smaller limit within the next frames. Davison et al. [11] pointed out that there are two good reasons for this kind of initialization: (1) Because of the unknown depth a known target allows to assign a precise scale to the estimated map and motion. (2) Predict-measurement-update tracking can be started without any further initialization steps.

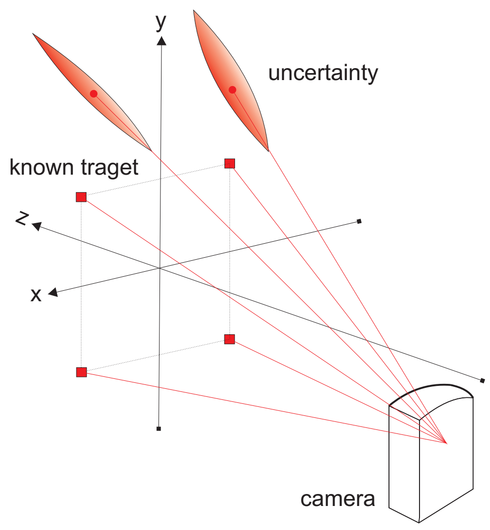


Figure 4.2: Example for the 3D map. Features are points surrounded by their uncertainty ellipses. Coordinate framwork orientated relative to the camera origin located within fix targets.

Constant Light
A prerequisite for a successful working MonoSLAM algorithm are constant light conditions. Monochrome images that vary in their intensity from frame to frame, are difficult to measure features again. For features matching using cross-correlation a templet patch will fail due to considerable differences of the pixel intensity between the template and the image search region.

Calibration
Active feature measurement and map building requires a camera model. Within the MonoSLAM approach a pinhole camera model is used, which allows the projection of the feature position onto the image plane. While the model telling us the image position gives the position of the camera and feature, it is not invertible, which means the image measurement and camera position cannot provide the feature 3D position [9].

A camera calibration needs to provide the following parameters of the standard pinhole camera model as input for the MonoSLAM algorthm: Focal length, principal point and pixel element densities per direction.

4.2.3 Motion Model

Using an agile camera as a sole sensor requires a different modelling of the motion model as for a wheeled robot moving on a plane (cf. Sec. 2.4). A main difference is the absence of knowledge about control actions as calculated by the robot and derived in Sec. 2.2.2. However, as denoted by Davison et al. [11] the unknown dynamics can be as suposed as additional noise, respectively process uncertainty and thus modeled probabilistically. The camera is assumed to move constantly over time, whereas on average undetermined accelerations are considered as a Gaussian distribution [11], where very large acceleration is assumed to be relativly unlikely [9].

4.2.4 Feature Extraction

Features are searched within a randomly placed search box in every acquired image frame. It is assured that the chosen position should not overlap with any existing feature and that any detected feature is not expected to disappear from the camera view immediately [11]. Within that search box, feature detection is performed by using the standard operator of Shi and Tomasi [36]. Following the approach of Davison and Murray [8] a 11x11 pixel image patch is extracted as a long-term landmark.

4.2.5 Initialization of New Features

One of the characteristics of the feature measurement model is that it is not invertible [9]. The 3D position of the feature is unknown and needs to be estimated before the predict-measurement-update steps can start. The method introduced by Davison et al. [11] assumes that the feature must lie along a semi-infinity line with the estimated camera position as origin set, heading to infinity (practically 0,5-5m) through the viewing direction of the feature. A set of discrete depth hypothesis are uniformly distributed along that line, analogous to a 1D particle distribution. Assuming that the line remains unchanged within the next frames, each hypothesis of a 3D position is projected into the 2D image, enclosed by an elliptical search region representing the uncertainty of the image location (based on the uncertainty of the camera and feature position).

Matching features within each ellipse is performed by a normalized cross-correlation search of the template patch that has been extracted by the detection operator (cf. Sec. 4.2.4) [11] . Features that match produce a likelihood for each particle and their probability in the distribution are reweighed [9]. The procedure - projection - seach - match - weight is repeated and the distribution shape changes towards a more Gaussian distribution. When the standard derivation of this quasi-gausian probability distribution reaches a certain threshold it is safely approximated as Gaussian and the feature can be initialized as a point into the map with its 3D Gaussian distribution [11].

4.2.6 Active Feature Measurement

For re-observed features that are already within the 3D map a measured attempt is performed within each new image frame. Before performing a measurement the feature position within the image is predicted based on the current camera and feature position in the state vector. With this intermediate step only regions of interest are searched for a possible match. This narrows down the search and maximizes the efficiency [11]. In the following the key steps are explained as introduced by [11].

- Firstly, the point feature position, - relative to the camera position based on the camera and feature estimation in the state is determined.
- Next, for measurement prediction the feature position is projected onto the image plane using the the standard pinhole model.

- The perspective-projected feature coordinates are warped with radial distortion to obtain the final measurement prediction of the feature position in the image plane.
- Finally the Covariance innovation matrix for this warped measurement prediction is calculated (cf. Sec. 2.6.5, 2.3.3, 2.3.2 for derivation of innovation within EKF).

The Innovation matrix represents the a 2D Gaussian probability over the predicted image coordinates. Choosing a standard deviation of 3σ defines an elliptical region in the image plane [11].

The key contribution by Davison et al. is to use the elliptic region as search region for re-observed feature measurement. High innovation covariance provides more information about estimated camera and feature positioning. Taking advantage of that, it is possible to select only those predicted image coordinates with high innovation covariance, and thus limiting the number of features to be searched per image frame (10-12) [11]. Finally a cross-correlation search for the template patch extracted during feature initialization is performed in the elliptic region. The patch is tested for a match at each location until a peak is found [11].

4.2.7 Estimation of Feature Orientation

Initially the assumption is that the feature corresponds to a locally planar region in 3D space [11]. The feature orientation is set during initalization as surface normal parallel to the vector from feature to camera. By re-observing the landmark the feature orientation is corrected. The updating process is separated from the main SLAM algorithm of state vector and covariance matrix EKF update.

Following Davison et al. [11] the initial camera position and local surface orientation for each point are maintained in a separate list. The prediction of a warped feature from the current viewpoint is compared to the obtained measurement. The difference between the prediction and measurement is used to update the surface orientation. The process repeats itself when the landmark is measured again.

4.3 Example MonoSLAM Framework

In the following section there are presented examples of 3D maps and camera image frames of a MonoSLAM application. The figure colour code follows as: red = successfully measured, blue = attempted but failed measurement, and yellow = not selected for measurement on this step [11]. The demonstrated sequences depicted the whole life time of a sequential build map. Starting from initializing the coordinate framework at the system start-up, to active feature measurement and feature initialization during camera movement. Also an automatic map management is presented, where the features measurement is not active or has failed and therefore features are deleted.

4.3.1 System Startup

The camera is placed in front of a black rectangular target with white background for the coordinate framework initialization. Fig. 4.3 shows the four corner template patches projected into the image frame overlapping the rectangular target. The quadratic patches appear within a yellow frame because measurement and tracking is disabled. In the corresponding 3D map the corners are depicted as point features located within a fix frame relative to the camera coordinate system.

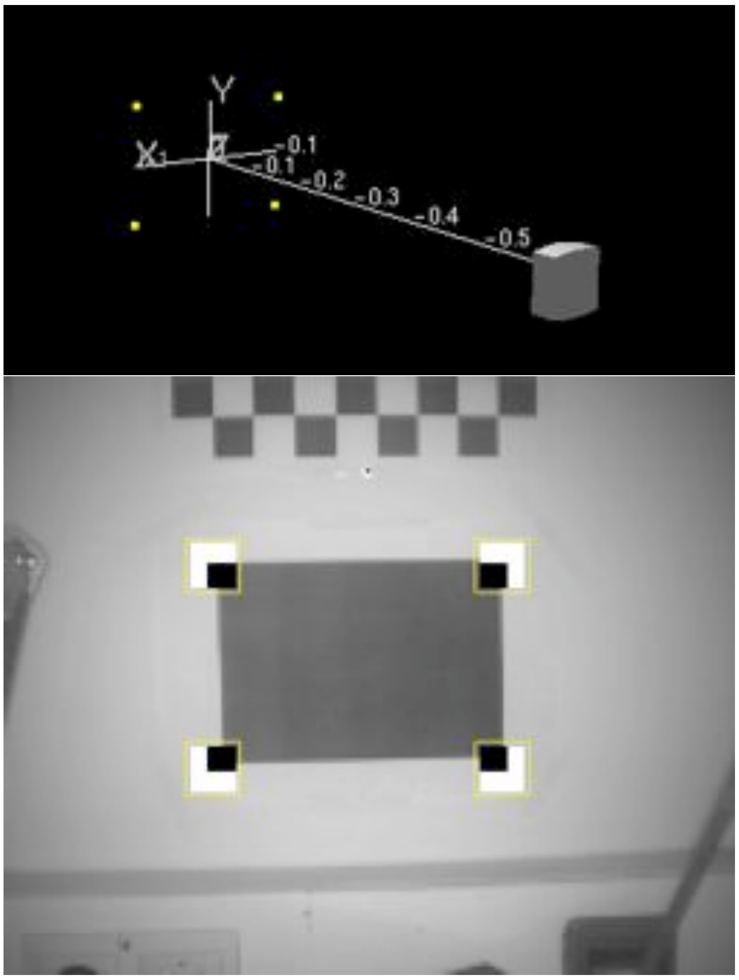


Figure 4.3: 3D map and image frame with target corners at system startup.

In the next image frame tracking and mapping has been enabled. Fig. 4.4 shows in the image frame the locked on features with their initial assumed uncertainty ellipses based on the covariance sub matrix $P_{y_iy_i}$ as explained in Sec. 4.2.1. The red colour indicates that the detected features are actively measured (cf. Sec. 4.2.6). Derived from the known postion of a fixed target in relation to the camera postion the camera pose can then be estimated. This is performed by the EKF update of state vector $\hat{x}_t$ and covariance matrix P_t as depicted in Eq. 4.1.

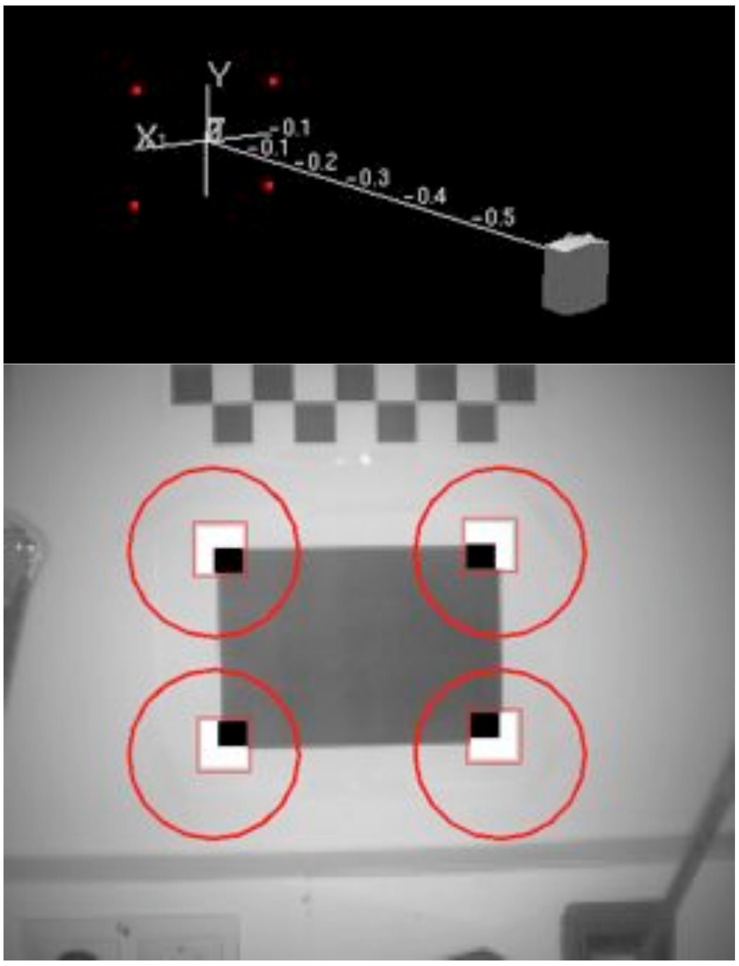


Figure 4.4: 3D map and image frame with target corners and active measurement at system startup.

Fig. 4.5 shows that due to the active measurement the uncertainty ellipse is reduced by the EKF update within the next image frames.

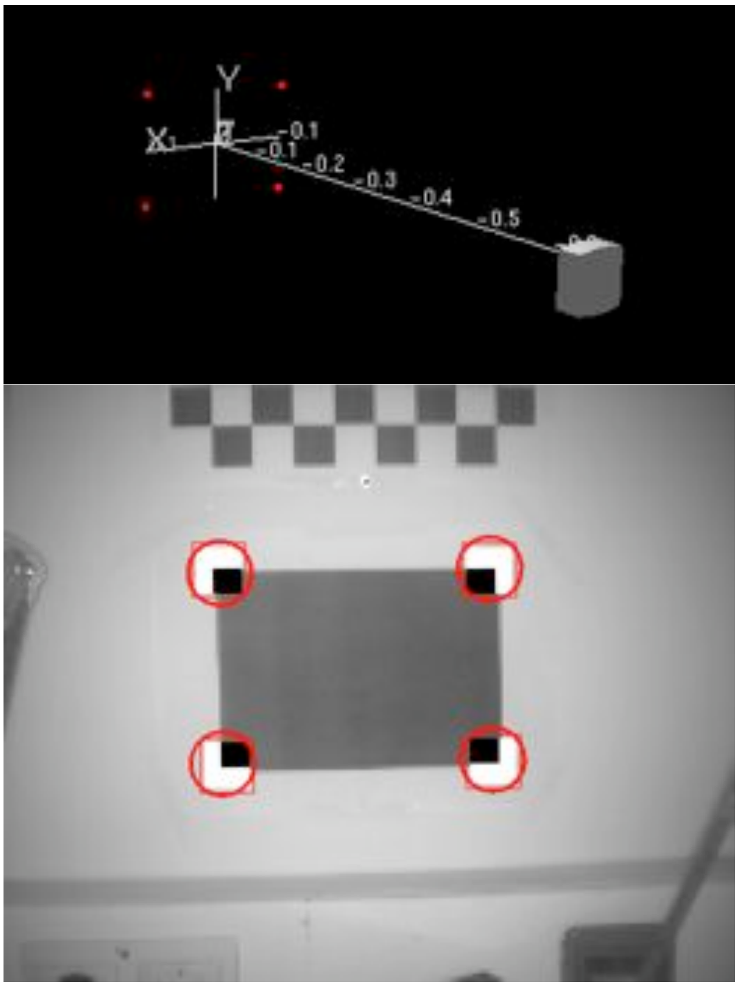


Figure 4.5: 3D map and image frame with target corners, active measurement and reduced uncertainty ellipses at system startup.

4.3.2 Feature Initialization

Each image frames is actively scanned for new features by the feature detection operator of Shi and Tomasi (cf. Sec. 4.2.4). Fig. 4.6 depicts a match within the green search box in the image frame. As shown in the 3D map, the depth of the new feature is initialized as a green particle distribution along the cyan viewing line of the feature as explained in Sec. 4.2.5. The corresponding uncertainty ellipse of each postion hypothesis is coloured cyan within the image frame.

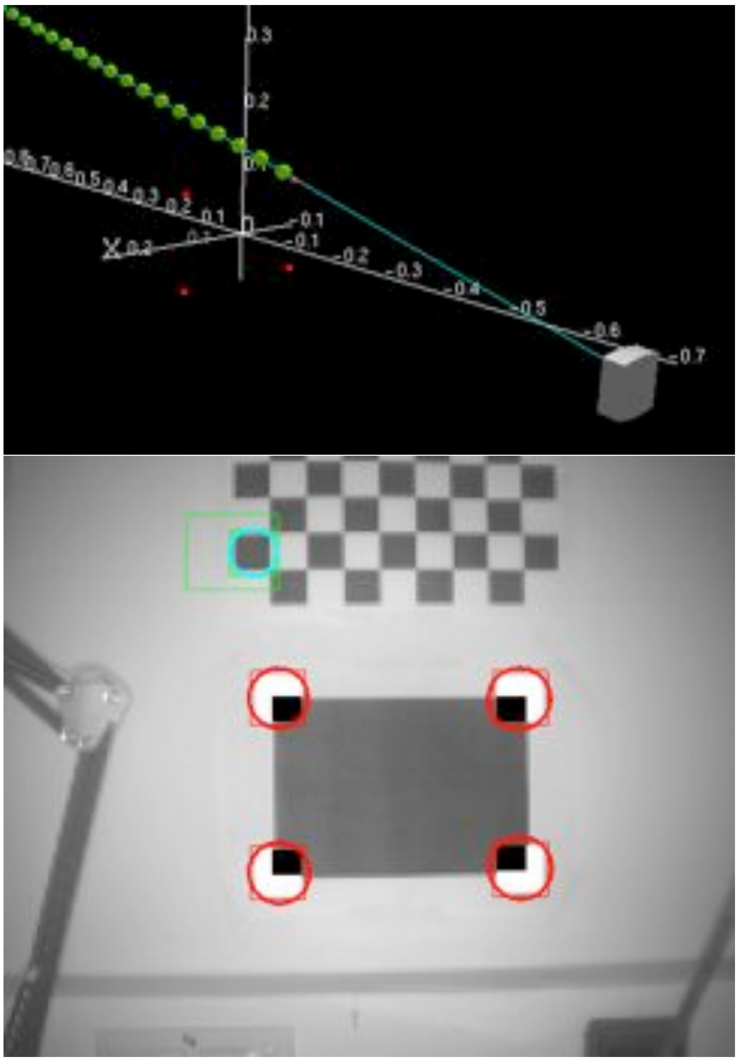


Figure 4.6: 3D map and image with partial feature initialization.

Fig. 4.7 illustrates the particle distribution in a subsequent frame in the 3D map. The particle size denotes the re-weighted probability distribution based on their matching likelihood of the template patch in the image frame within the elliptic search region. The distribution profile converge to a Gaussian distribution. The cyan ellipse in the image frame represents a search ellipse for each feature position hypothese.

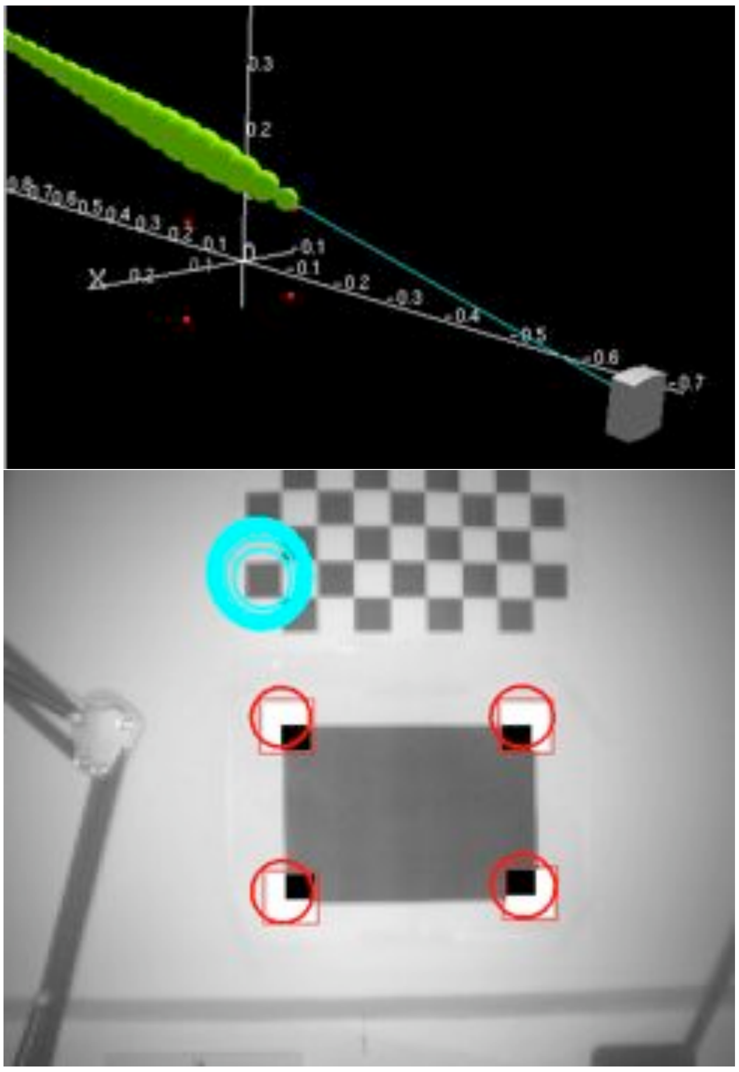


Figure 4.7: 3D map and image with partial feature initialization.

Fig. 4.8 shows, the same feature a couple of frames later. The feature did convert from a partially initialized feature into a fully initialized feature, while particle distribution reaches a certain threshold probability for a Gaussian profile. The 3D map depicts the feature postions uncertainty, the 3D ellipse is colored yellow. Note that yellow indicates that, the feature is not active, meaning not used for active measurement updates. The image patch of the detected region coloured yellow in the image frame is stored and furthermore used as a template patch when the feature becomes active.

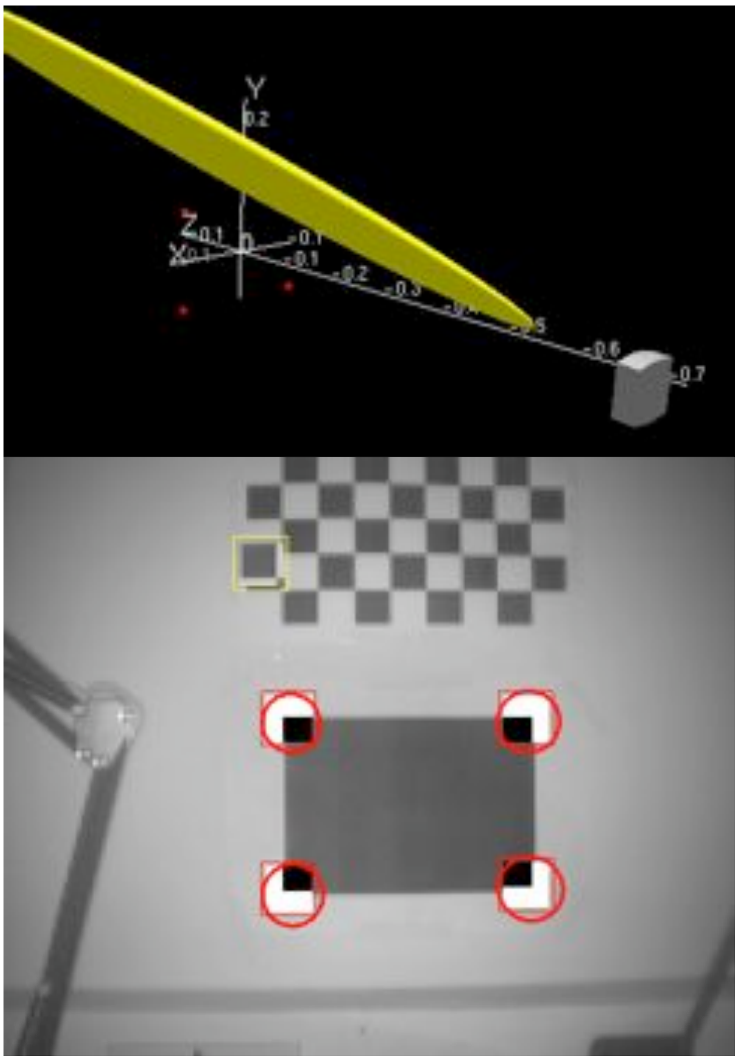


Figure 4.8: 3D map and image with a converted full initialized feature.

Fig. 4.9 depicts the same feature on frames later. The feature is actively measured (coloured red) EKF update of its pose and covariance matrix that is performed visibly by the shrinking uncertainty of the ellipses in the 3D map and image frame. The green search box illustrates the parallel performed search for new features in the arrived image frame (cf. Sec. 4.2.4)

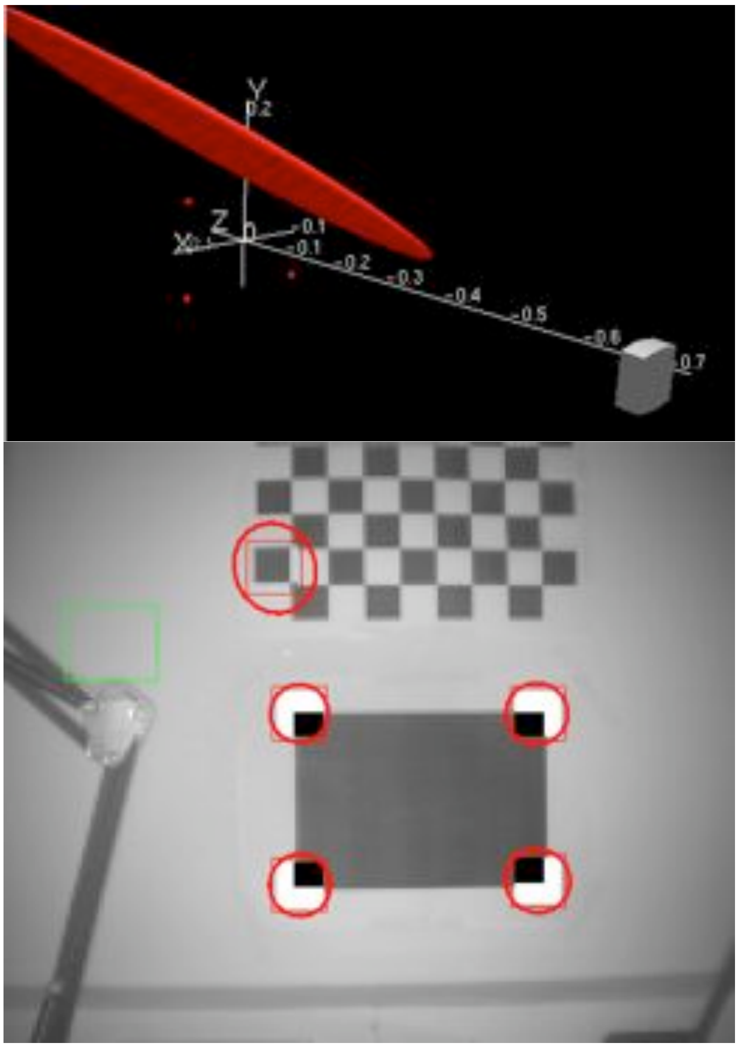


Figure 4.9: 3D map and image with full feature initialization after EKF update step.

4.3.3 Active Feature Measurement

Fig. 4.10 illustrates the active measurement of 6 re-observed features. For each feature the new feature postion in the image frame is obtained as described in Sec. 4.2.6. For each feature a sequencial EKF update of the state vector and covariance matrix is performed. Thus the measurement is updated for of each re-observed feature in the image frame helping to reduce uncertainty ellipses of all active features.

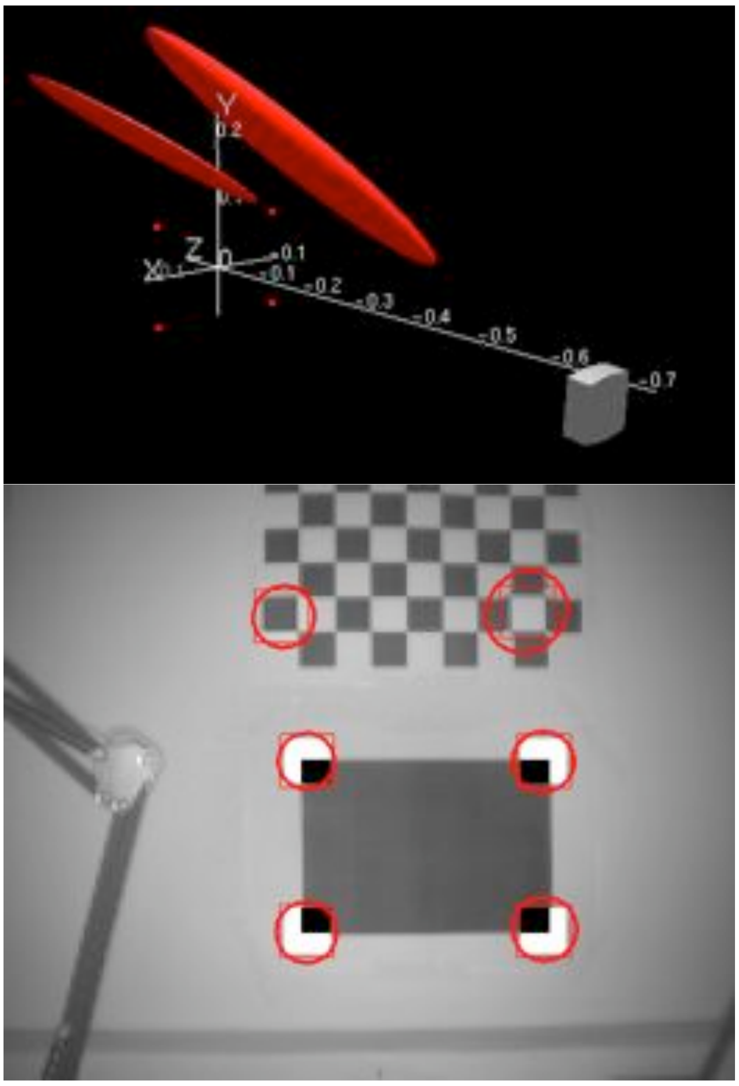


Figure 4.10: 3D map and image with ative measurement and EKF update.

Fig. 4.11 depicts the feature subsequent frames later. Compared to Fig. 4.10 are the uncertainty ellipses highly reduced with respect to the smooth camera motion (cf. Sec. 4.2.3). The changes of the camera pose is small compared to the reduction of the uncertainty ellipses. Note that the uncertainty ellipses of the observed features are reducing as the features are observed more within the image frame.

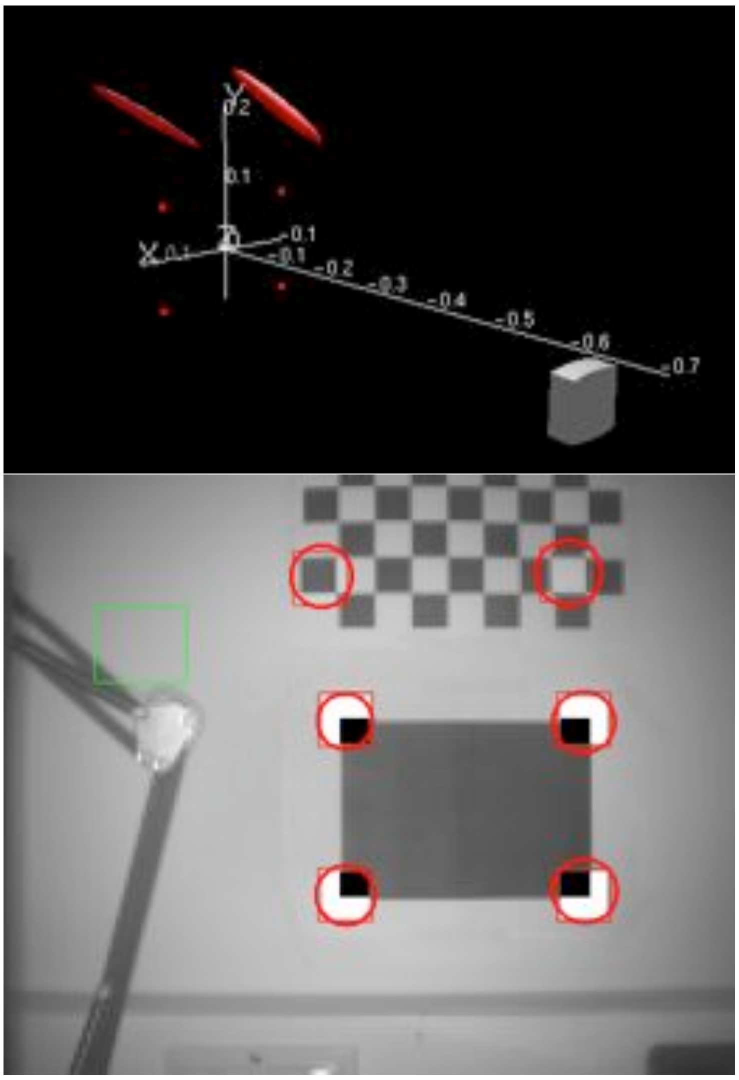


Figure 4.11: 3D map and image with active measurement and EKF update with subsequent reduced uncertainty.

4.3.4 Map Management

Not every feature can be reliably measured within the image frame. The MonoSLAM algorithm takes care of the feature state and performs only sensible measurement updates. Fig. 4.12 illustrates the different feature states. The yellow inactive features are too far on the left hand side of the image frame and thus not reliable for measurement. Only the three features coloured red are actively measured and used for a state and covariance EKF update. The measurement for the feature coloured blue on the right hand side of the image frame failed and therefore cannot be used for the measurement update either. The feature is deleted if the measurement success rate fails under a certain threshold ($< 50\%$) value.

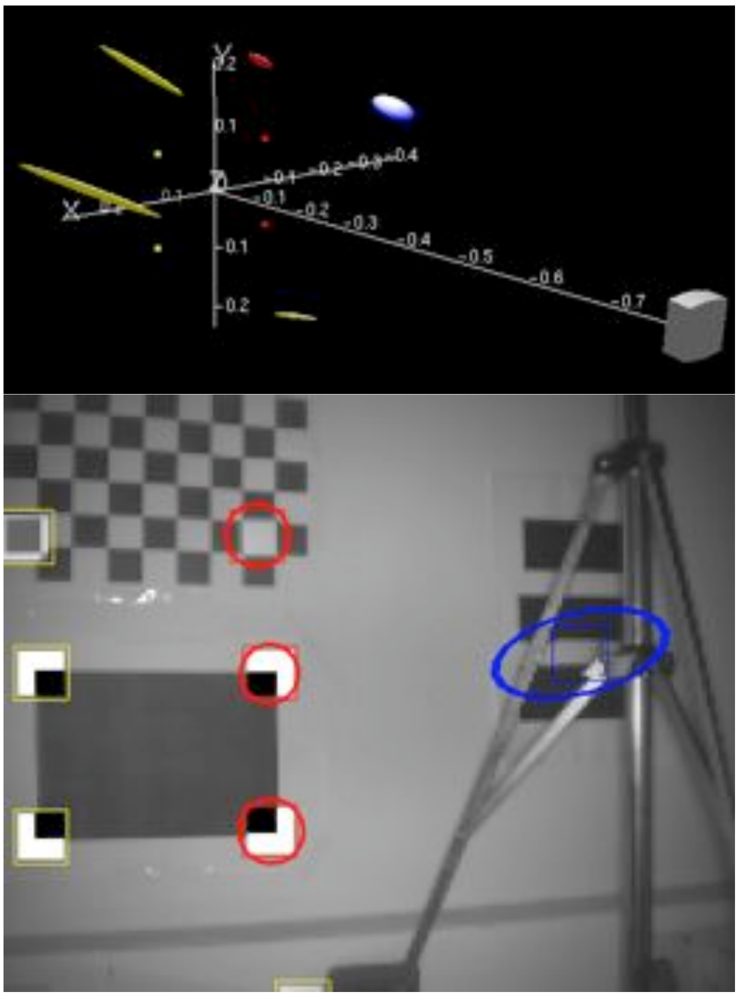


Figure 4.12: 3D map and image with different feature states.

Chapter 5

Fusion

Conventional cameras capture the environment geometry through photemetric sensors transforming light into pixel intensities. The transformation projects the 3D world into a 2D space, while losing the direct depth information. Recontructing three-dimensional structures from images falls in the realm of Computer Vision known as Structure from Motion(SFM) or Computer Stereo Vision (CSV). SFM evaluates motion for an object over time in order to regain its 3D geometry, whereas CSV calculates the distance from comparing the two images while shifting them together.

Considering non vision based system such as laser range finders or sonic systems, they are commonly used to gather depth information of an environment. The 3D camera as introduced in Chap. 3 supports both of the two apporches. A 2D image is obtained through a photometric sensor and the depth information is measured through active light measurement by the ToF principle. In this section we propose an approach that uses the camera generated distance and 3D data within the Visual SLAM algorithm descripted in Chap. 3, instead of using a 2D image to reconstruct the depth and 3D structure information.

5.1 Visualization of Distance Data

With the conventional 2D approach (SFM) it is difficult to extract features in real time and to generate reliable long-term maps. When it comes to a dense map of features or objects it seems even more difficult if not impractically. Because of the inherent distance measurement of the 3D camera, solving this problem is straight forward. For each pixel the corresponding distance point is calculated in the 3D environment as described in Chap. 3. These points can be easily mapped into a 3D map in real time, which results finally results in a dense 3D map. Fig. 5.1 outlines a 3D map including structured point clouds.

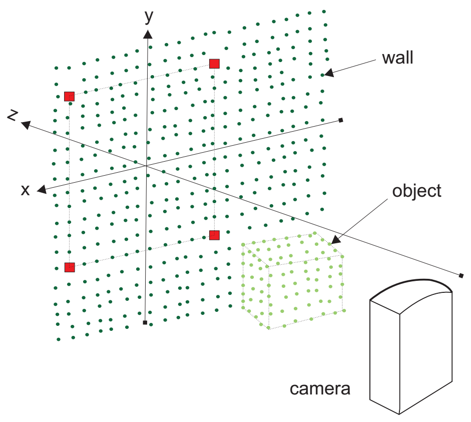


Figure 5.1: Drawing of 3D map including distance points.

5.2 Feature Initialization Using Distance Data

One starting point to actively apply the measured distance to the MonoSLAM algorithm introduced in Chap. 4 is to use depth for feature initialization as described in Sec. 4.2.5. In our approach we reduce the range that the particle distribution is assumed to lie on. By knowing where the feature has been detected in the 2D images the corresponding xy coordinates in the 3D map can be determined. Based on that we can calculate with the knowledge of the depth information the corresponding pixel in the 2D image and, the feature distance. Due to the uncertainty for the measured distance accuracy we apply the standard deviation σ to the range limits (cf. Sec. 3.2.3). Therefore the reduced range for feature initialization can be denoted as

$$d = \sqrt{x^2 + y^2 + z^2} \quad \text{(measured data), and} \tag{5.1}$$
$$\lambda = [d - \sigma, d + \sigma] \quad \text{(particle range).} \tag{5.2}$$

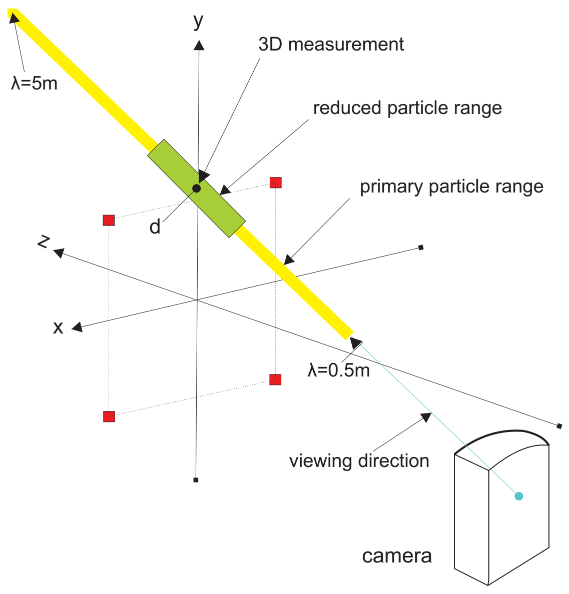


Figure 5.2: Outline of feature initialization with different particle ranges.

5.3 Identifying 3D Features

In Visual SLAM and in particular MonoSLAM (cf. Sec. 4.2) identifying a point features is based on a 2D image. The feature detector does not evaluate the 3D structure of the match. Therefore no conclusion can be made where the feature has been detected, e.g. in the plane or on the edge. To provide depth information a 2D feature needs to be initialized e.g. as described in Sec. 4.2.5. However, the resulting 3D feature location does not provide information about the feature structure.

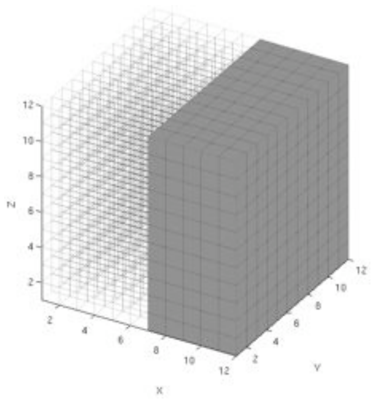
The following approach describes a method to use the inherent measured 3D coordinates by the camera, to find information about the features structure. Using this in the MonoSLAM algorithm would increase the reliability of the detected feature and thus the building of a long term maps generated in real-time. The following mathematical concept of the structure tensor describes a method to derive 3D corners from a 3D image.

5.3.1 Structure Tensor

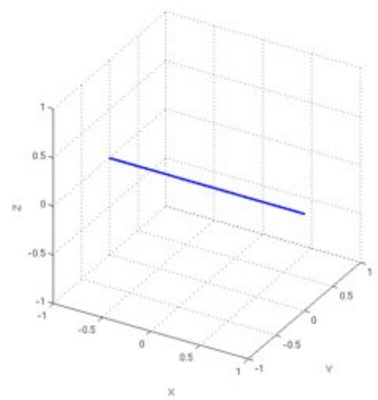
Object structure in an image can be highlighted by its gradient information. It can be used to identify features like edges or lines. A special method of representing gradient information is by using the ”structure tensor” as explained by [4]. The 3D structure tensor is formed by the matrix

$$S = \begin{bmatrix} I_x^2 & I_{xy_1} & I_{xz} \\ I_{xy} & I_y^2 & I_{yz} \\ I_{xz} & I_{yz} & I_z^2 \end{bmatrix}. \tag{5.3}$$

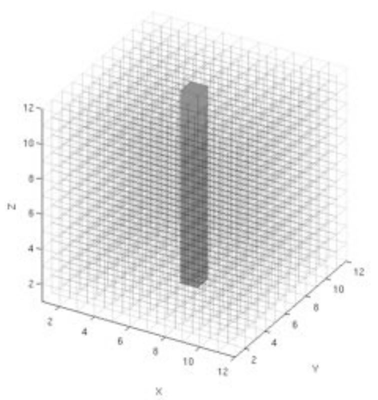
Where $I_{x,y,z}$ denotes the partial derivative of image I along the x, y, z-axis. The eigen decomposition of matrix S provide its eigenvalues $\lambda_1, \lambda_2, \lambda_3$ and eigenvectors e_1, e_2, e_3. It can be visualized as a 3D ellipse spanned by the eigenvectors with the radii equal to its corresponding eigenvalue. With this representation it is possible to determine simple 3D structures. Fig. 5.3 depicts examples for different elements.



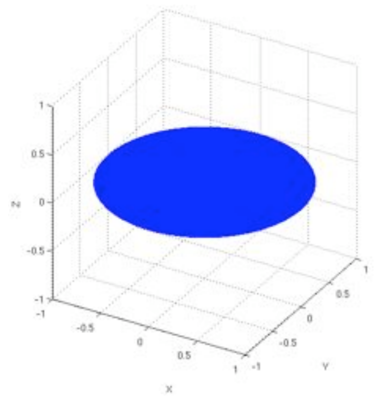
(a)



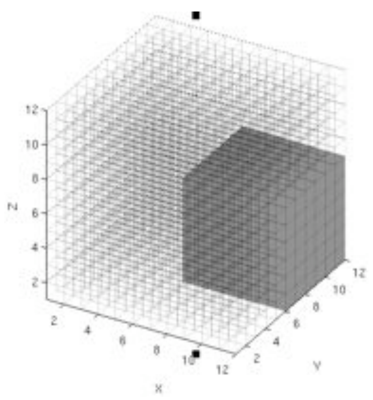
(b)



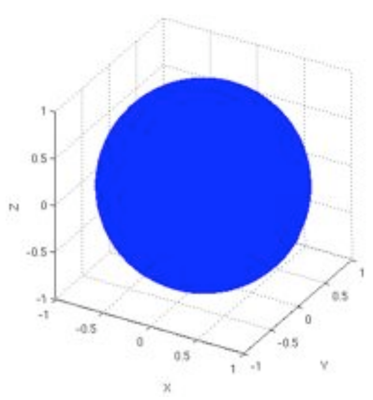
(c)



(d)



(e)



(f)

Figure 5.3: An example for ideal 3D structure elements with decomposit eigen values. (a) ideal plane, (b) eigenvalues represent line, (c) ideal curvel, (d) eigenvalues represent 2D ellipse, (e) ideal corner, (f) eigenvalues represent sphere

As depicted by Fig. 5.3 (f) the eigenvalues for a corner are represented ideally by a sphere and thus have equal values. Due to the lack of time it was not possible to determine an analytic method to judge which form is represented by which eigenvalues with respect to a single threshold determining the degree of similarity. This has been deferred to future investigations.

Chapter 6

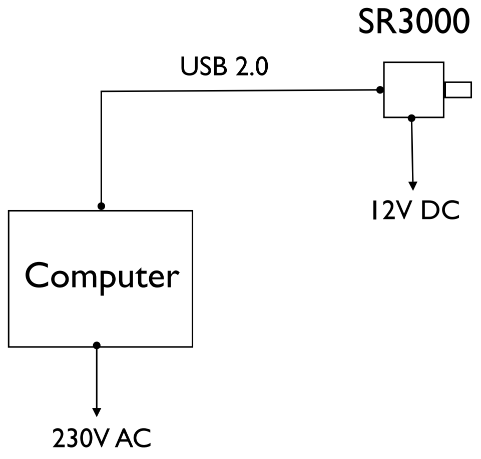
Implementation

6.1 Assembly

The Assembly gives a short overview about the used hardware and software. Firstly the used main hardware components are shown, secondly the relevant software is listed.

6.1.1 Hardware

The main hardware components consist of a standard computer (macbook, 2GHz Intel Core Duo, 2GB RAM), an USB connector cable, the SwissRanger SR3000 ToF camera, an ACDC converter including a +12 DC power connector. Following figures 6.1 depict the schematically and real hardware setup including auxiliary equipment.



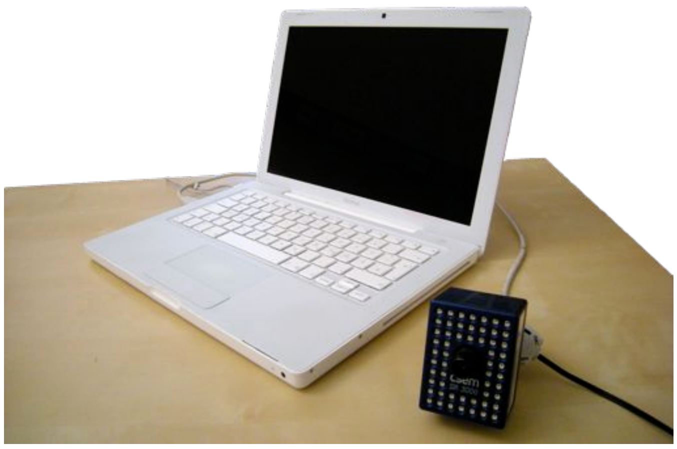


Figure 6.1: Schematically overview left real setup right.

6.1.2 Software

The software components can be grouped in three different categories: (1) Operation System related, (2) camera related, (3) MonoSLAM related. The installed operation consists of a 32 bit Linux, where Ubuntu 8.04 LTS was chosen as the preferred distribution. In oder to have Linux as a native operation system on the macbook, a separate partition was created to ensure that Linux and Mac OSX can run separately. The reason to use Linux and not for example Mac OSX comes with the fact that further used software as the camera API and the MonoSLAM Framework have been proven to run on Linux already. The camera related software consists of the SwissRanger SR3000, USB driver and Application Programming Interface (API), the Shared Memory Server and Client. The camera API is part of the camera delivery, whereas the Shared Memory Class, Server and Client are additional software components provided within this thesis work. The MonoSLAM framework consists of a powerful C++ library and an application implementing a vision based SLAM approach based on a monocular camera as sensor. Additionally software modifications has been applied in oder to utilize the SR3000 as a sensor. Following figure 6.2 provides an overview of the used software components:

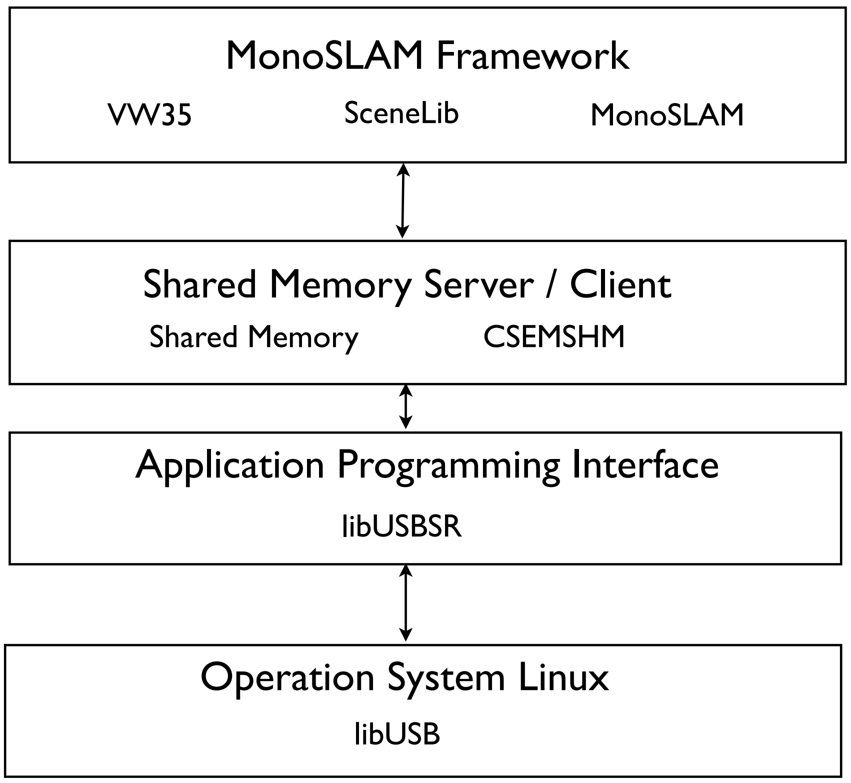


Figure 6.2: Software building blocks and their interfaces.

6.2 Camera Interface

Within this section the camera specific interface, the implementation of the Shared Memory Class and the Server and Client application are explained in more detail.

6.2.1 SR3000 API

The camera Application Programming Interface consists of several functions to setup the camera. The amplitude and distance data is acquired and processes the images distance further. All functions which are attached to the camera software are combined in the interface library libusbSR. This interface can be used to connect C,C++ code to the camera. Table 6.1 provides a list and a short description of important library functions. A complete and detailed description can be found in [25].

SR Open & Close	Opens and closes a connection to the camera via the USB device and assigns a unique camera handler to it.
SR Set Buffer	Sets a buffer that allocates free space on the computer. The camera image (first place amplitude second distance) is copied from the sensor into the buffer. However, the buffer size needs to be defined by the user
SR Acquire	Acquires the image (amplitude and distance). The distance is given in spherical coordinates and corrected according to the focal length, principal point and FPN offsets in the calibration file (cf. Sec. 3.2.5) Further a median filter (3x3) can be applied if needed
SR Get Image	The helper function returns the index to the first position of the image in the buffer. A offset parameter allows it to jump directly to the starting point of the amplitude or distance values.
SR Coordinate Transformation	Transforms the spherical coordinates into cartesian (x,y,z). Three versions exist for the data type: (1) double, (2) float, (3) unsigned integer. For each coordinate a separate buffer needs to be be allocated by the user.
SR Set Amplitude Threshold	It sets the maximum allowed amplitude value. Pixels with a lower value are filtered and set to '0'
SR Set Integration Time	Sets the integration time by intervalls from 0 to 255 (cf. Sec. 3.2.2)
SR Set Auto Illumination	If set changes the integration time automatically based on the perceived intensity. An integration time interval, an illumination threshold and target intensity value needs to provided. The given threshold defines the percentage of the number of amplitudes that are over a certain intensity value. The integration time is slowly adapted until the threshold has reached the target intensity value. For example: start at 10% over intensity 50 will stop at 10% over intensity 100.
SR Set Modulation Frequency	Set the modulation frequency of the camera LEDs. This will change the non-ambiguous distance range. $30MHz \rightarrow 5m$, $20MHz \rightarrow 7.5m, 19MHz \rightarrow 7.9m$ (cf. Sec. 2.5.5)

Table 6.1: SwissRanger SR-3000 interface library selected functions.

6.2.2 Shared Memory and Semaphore

The Shared Memory (SHM) Class is based on C++ code. It interconnects the camera API with a shared memory segment hold by the operation system. Shared memory can be used for interprocess communication, where different processes, respectively programs communicate via the same memory segment of the computers random access memory (RAM). In addition to the shared memory a control mechanism has been added that ensure that only one program can access the shared memory at time. A common solution for this problem is the semaphore (SEM) concept. In a practical implementation the semaphore function checks whether an access is possible or not. It waits until the lock has been cleared before the next in the shared chain is accessing the resource. Semaphores are commonly used in the synchronization of multiple processes or applications that want to access the same resource. For the relevance of this implementation semaphores and shared memory are considered from a practical point of view. (We refer to relevant computer science literature for more information.) For this implementation a shared memory segment and semaphore access have been chosen for several reasons:

- The additional layer between the camera API and the MonoSLAM framework provide more independence with respect to the camera functions. Within the MonoSLAM classes no additional camera specific functions need to be implemented. Only the shared memory class which encapsulate the API from the higher software layers.
- Separation of the camera and MonoSLAM applications. Its is possible that more than one application can easily access and acquired the same image frames.
- Fast and easy accessible data in comparison to Unix socket or the common TCP/IP network communication. Additional effort is needed to be undertaken to implement a proper communication sequence and assure the needed performance for grabbing the image. With respect to this context shared memory access offers a straightforward and easy methode.
- Reliable and basic operation system support. Unix System V has on default a built in Application Programming Interface for shared memory and semaphore management.

The downside of this approach lies in the inherence of an extra layer between the camera and the application software. It introduces additional complexity and an additional delay for image acquisition. Further a more simpler solution for access control could be implemented, due to the fact that maximal two processes have access to the shared memory. However, the overhead is rather small and the introduction delay can be considered as marginal. In the following, Tab. 6.2 lists the most relevant shared memory and semaphore functions.

Shared Memory Constructor	Creating a the shared memory when initialized with the allocated memory size. To guarantee unique access a SHM key is generated. Based on that key the operation system assignes a SHM ID to the established segment. This is archived with the system function *shmget*. The ID is important for further access to the shared memory. The constructor checks if the SHM key and the SHM ID already exists. If yes, it retrieves the existing SHM ID.
SHM Attach	Attach to the shared memory with respect to the SHM ID. The system function *shmat* returns the starting address of the created memory segment. This address is used to perform read and write operation on the shared memory.
SHM Detach	Detaching from the shared memory with respect to the starting address. By use of the system function *shmdt* the operation system releases the mapped SHM segment from the address space of the calling process.
SHM Delete	Deletes the shared memory with respect to the SHM ID. The system function *shmctl* will mark the memory as to be deleted. It will be finally destroyed after the last process has detached from the memory.
SHM Recover	Recovers the shared memory with respect to the shared memory key. This function can be used in case the memory segment has been deleted. The same SHM ID is generated based on the SHM key. If the segment already exists only the memory ID is stored.
Semaphore Constructor	Creates the semaphore set based on a SEM key. The retrieved SEM ID is assigned by the operation system. This is archived with the system function *semget*. Further the semaphore is set to the unblock state. The constructor also checks if the key already exist. If yes, the existing SEM ID only is retrieved.
SEM Operation	Increments or decrements the semaphore variable to unblock or block the semaphore. The operation is defined by the values -1 and 1 and is performed by the system function *semop*. If a semaphore has been blocked, processes that try to decrement the semaphore will be put to sleep until it is incremented again.
SEM Delete	Deletes the semaphore with respect to the SEM ID. The semaphore is immediately removed.
SEM Recover	Recovering the semaphore with respect to the SEM key. If the semaphore has been deleted then the same SEM ID is recreated. In case the SEM ID already exists it is stored for further handling.

Table 6.2: Shared memory and semaphore class selected functions.

6.2.3 Shared Memory Server

A control program has been developed in C++, in order to acquired camera images and to store them within a shared memory segment . It is based on the camera API of the SR3000 and the Shared Memory and Semaphore Class. Following tasks are performed by the Shared Memory Server application (cf. Fig. 6.4) .

- Creating, connecting, reconnecting, and deleting a shared memory segment by using the Shared Memory and Semaphore Class. Defines the memory segment data structure. Allows access to the shared memory through the semaphore operation.
- Initialising the camera by defining a camera handler, setting up camera buffers for image and coordinate transformation. Further settings have predefined values for integration time, the amplitude threshold and auto illumination flag.
- Acquire image frame through camera API function and copies the amplitude and distance values, Cartesian coordinates and the 8-bit grayscale image to the shared memory. Updating the new image flag and image counter variable of the shared memory data structure.
- Transforming spherical distances into Cartesian coordinates (x, y, z) with respect to the camera calibration (cf. Sec. 3.2.5). Convert the amplitude data into a 8-bit grayscale image.

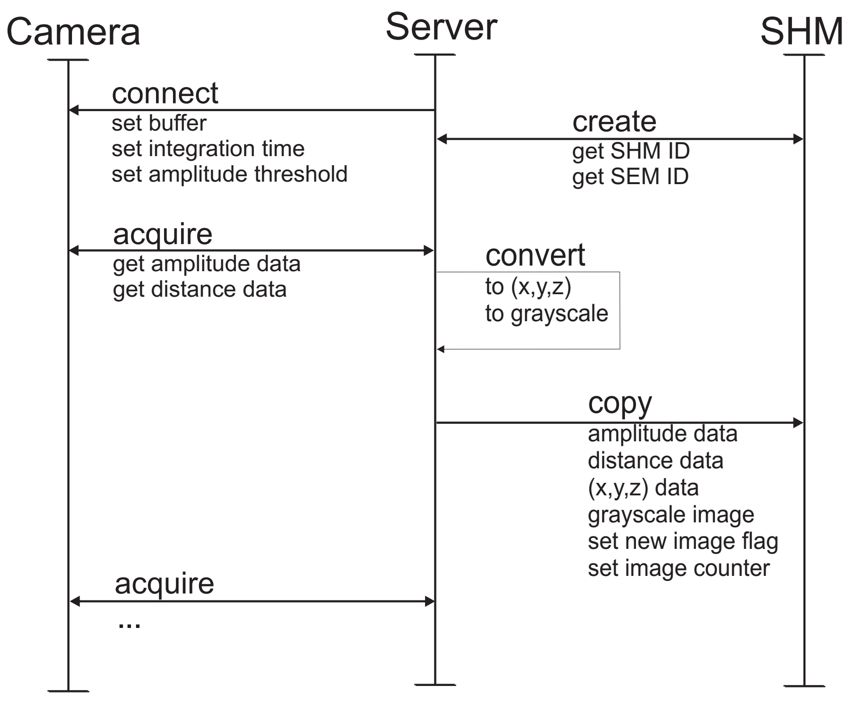


Figure 6.3: Communication flow from/to camera, server application and shared memory.

6.2.4 Shared Memory Client

The Shared Memory Client connects to the existing shared memory and query the image date of a single frame based on the Shared Memory and Semaphore library Class. With the stand alone client a snap shot of the the images values can be taken, in order to save the image values to a file for further processing. The following tasks are performed by the Shared Memory Client application (cf. Fig. 6.4).

- Query the shared memory and semaphore ID by using the Shared Memory and Semaphore Class. Access the shared memory through semaphore operation.
- Acquire image frame through the shared memory data structure and printing the data values to the output terminal. A selection of different input switches are available to obtain a single output for amplitude and distance values, Cartesian coordinates x, y, z and the 8-bit grayscale values. This can be configured either by a continuous mode or as a single shot.

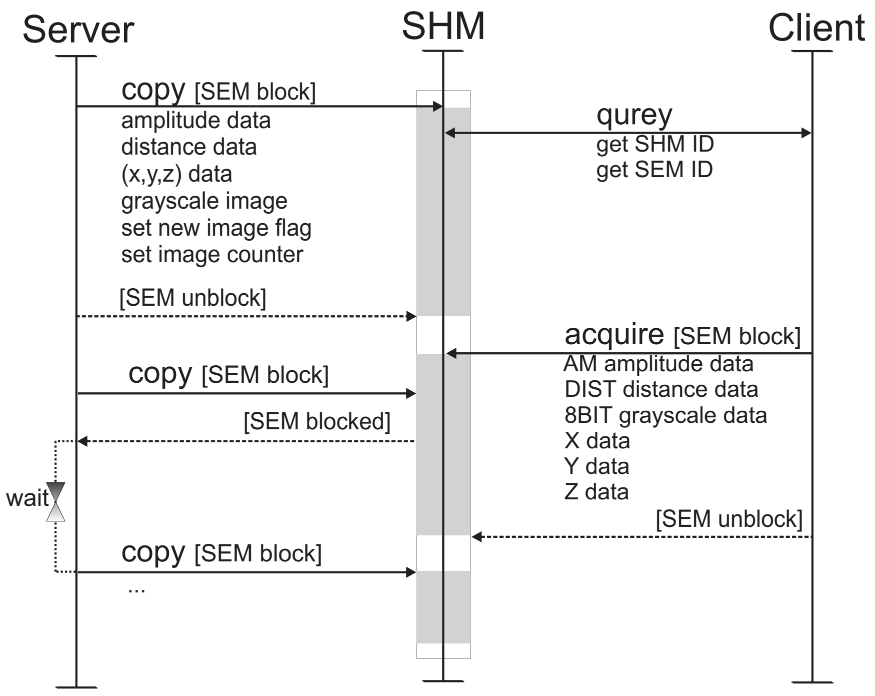


Figure 6.4: Communication flow from/to client, server application and shared memory. Gray bar indicates blocked semaphore. Example for the case client blocks access to shared memory via semaphore , while a server tries to connect. When the semaphore is unblocked the server process gets a wake up signal and reconnects.

6.3 MonoSLAM Framework

The MonoSLAM Framework SceneLib 1.0 is a free open-source C++ library, for real-time Simultaneous Localization and Mapping. It is designed and implemented by Andrew Davison et al. [12], and depends on the open source computer vision library VW35 developed at Oxford's Active Vision Lab by David Murray et al. [27] and the Open Graphics Libary (Open GL). The Scene platform provides general algorithms and methods to built SLAM applications. The details of the real-world implementations can be plugged into Scene by the definition of a model class. In the following sections a brief description of important Scene and VW35 libraries is provided. Based on that a sample application for sparse monocular SLAM is introduced. The modification of the VW35 and MonoSLAM application with respect to the camera sensor SR3000 is discussed.

6.3.1 VW35

VW libraries provide C++ objects and algorithms for building Computer Vision and other applications [3]. This includes numerical functions, simple image and image feature objects, hardware interface classes as well various widgets sets supporting GUI interfaces. E.g. matrix and vector classes, image processing and feature detection algorithms, FireWire support and the GUI libraries like OpenGL, GLUT and GLOW toolkit. In the following, relevant libraries (Fig. 6.5 and Tab. 6.3) are described with respect to an interface implementation of a shared memory segment. For a complete description of the full VW libraries refer to the source code documentation [2]

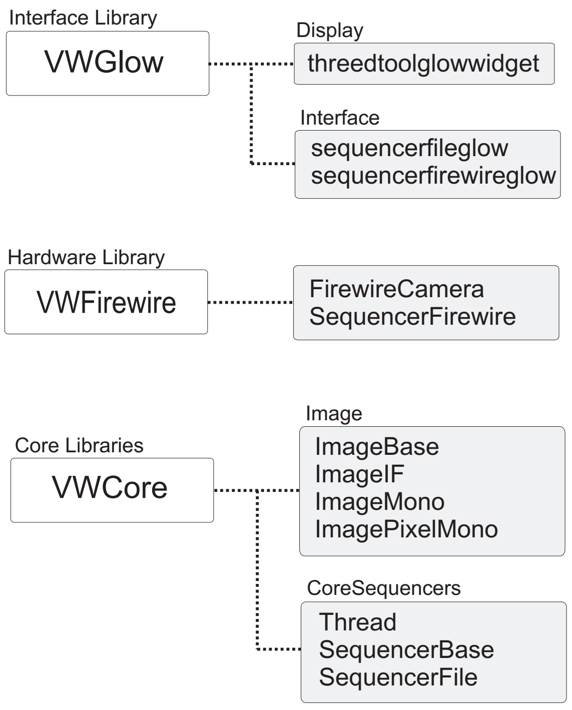


Figure 6.5: Overview of Libraries with a selection of related classes.

ThreeDToolGlowWidget	Class that create a control a Glow GUI window for 3D drawing. E.g. World coordinate system with camera pose and sparse feature map. It makes use of Open GL drawing function inherited from ThreeDToolGLCommon
SequencerFileGlow	Class that creates a Glow GUI control window for loading new image files based on the SequencerFile class
SequencerFirewireGlow	Class that creates a Glow GUI control window for loading new images based on the SequencerFirewire class
FirewireCamera	Class to provide control of a FireWire (IEEE 1394) digital camera. E.g. setup camera, start/stop ISO transmision, set image mode, get node number, Get ISO channel.
SequencerFirewire	Class that knows how to get images from FireWire cameras by making use of the FirewireCamera class. E.g. InitSequencer, CopyImage, GetBuffer, GetImageWidth, GetImageHeight
ImageBase	Base class that provides functions for constructing, storing, loading and operating on images. It is important as an abstract class to enforce consistency.
ImageIF	Abstract base class that ensure all derived image classes have specific functions in place. E.g. GetHeight, GetWidth.
ImageMono	Class for storing monochrome images, with pixels as single values.
ImagePixelMono	Class for storing luminance values only. Default type is 8-bit unsigned char.
Thread	Base class from which a multi-threaded application can be derrived. Used by SequencerBase to run as a single process independent from higher SLAM applications.
SequencerBase	Base class for all sequencers, provide basic mechanism for moving a single frame and continuously getting frames. In case a new frame arrives, a signal it is sent and can be retrieved by using a copy image function. A frame is generally assigned to one channel.
SequencerFile	Class that loads several ordered files. The file format is defined as the name_000000.xxx. It is possible to process a selection of image types. E.g. jpeg, png, gif, ppm, pgm, eps.

Table 6.3: List of selected VW35 libraries and classes related to a hardware interface implementation.

6.3.2 SceneLib

The SceneLib library provide generic classes and functions for performing Simultaneous Localization and Mapping (SLAM). Scene can be applied where it is wished to estimate the states of a generaly moving robot and many stationary measurable features via one or more sensors. They are fix mounted relative to the robot [10]. It manages basic feature information as feature's state, covariances and identifier via a feature model, implements the Extended Kalman Filter, a motion model describing the movement of a robot or sensor and provides an internal measurement model for making measurements of the robot state [40]. Further it enables it to manage a single EKF-SLAM map. E.g. creates and initializes new features, removes features, makes measurement of features. Creates and updates the robot feature states and covariances. Provides basic function to draw feature parameters as uncertainty, prediction and measurement. An important insight of the feature model is the split into partial and fully initialized feature [10]. This design has been applied to monocular Visual SLAM because of the uncertainty in the depth dimension. New features are initialized first using a particle distribution to estimate their depth before being transformed to full features in the EKF map. Following classes are considered as main Scene classes as compiled by Smith [41]. Refer to the source code documentation [40] for a complete description.

Scene_Single	Saves the entire system state and perform feature management.
Feature	Saves and manages a feature's state vector and covariances
Motion_Model	Basic class for all motion models. Describing robot or sensor movement. Accesses state and performs the state update. Stores no state itself.
Feature_Measurement_Model	Basic class for all feature and measurement models. Manages the feature's state and predicts a measurement. Splits into a complete feature in the map and a partially initializes the feature. It stores no state itself.
Sim_Or_Rob	Basic class that provides functionality for initialisation and the measurement of features, making internal measurements and controlling a vehicle.
Kalman	Friend class of Scene_Single that implements an Extended Kalman filter.

Table 6.4: List of selected Scene classes, where Scene build applications are derived from.

6.3.3 MonoSLAM

The MonoSLAM library is a particular implementation of a real-time Single Camera SLAM (MonoSLAM) based on the Scene library. It provides a set of classes and functions that initializes a 3D motion model for camera movements. It defines a pinhole camera model that can project image rays into the camera image and vice versa and provides a camera feature measurement model. It controls and runs MonoSLAM and provide an GUI independent interface for MonoSLAM applications such as MonoSLAMGlow. In the following main Scene classes are briefly described. For a complete and detailed description refer to the source code documentation [40].

MonoSLAM	Provides a main class that runs the MonoSLAM camera SLAM. It is independent of the GUI and application
MonoSLAMInterface	Provide an interface to the MonoSLAM core functions. Enable applications to control and receive feedback of the MonoSLAM functions. Inherited from MonoSLAM.
Robot	Class that models the Robot as a hand-held grayscale camera. Handles the image feature measurement. Inherits from the Scene class Sim_Or_Rob.
WideCamera	Provides a model of a pinhole camera with a cubic distortion model by Swaminathan and Nayar. Base class of WideCameraFeature.
Impulse ThreeD Motion Model	Class that provides a constant-velocity motion model. It assumes that statistics of its motion is on average Gaussian distributed for each time step.
ZeroOrder ThreeD Motion Model	Class that provide a zero-order motion model. The camera is assumed to stay in position. For each time step it moves from its location with a Gaussian profile.
Partially Initialised Point Feature	Provides the specific behaviour for standard image point features. Especially for initialization and matching. Inheriting from the Scene class PartiallyInitializedFeature and WideCameraFeature.
Fully Initialised Point Feature	A feature that performs point measurements from a wide-angle camera. It stores the feature state and the measurement state of the image location of the feature. Inherited from the Scene class Feature and WideCameraFeature.

Table 6.5: List of selected MonoSLAM classes, derived from SceneLib.

6.3.4 MonoSLAMGlow

MonoSLAMGlow is an application that implements a real-time Single Camera based on the MonoSLAM, Scene and VW35 libraries. It makes use of the Glow toolkit in order to build a GUI interface to visualize features position and uncertainties in a 3D map and augmented 2D camera images. Providing control over the camera capture process and enabling the tracking of features for the acquired scene. Fig. 6.6 considers the main functions of the MonoSLAMGlow application. Refer to the MonoSLAMGlow source code for a complete explanation.

SetUpMonoSLAM	Creating a MonoSLAM instance by use of the MonoSLAMinterface class and an initialization file. E.g. Number of particles, standard depth deviation, number of features to select, number of features to initialize at once, features to keep visible and time shift for each frame.
SetUpImageGrabber	Setup the sequencer that is related to the specific camera interface. E.g. FireWire.
SetUp3DDisplays	Setup function that draws and displays the 3D virtual view of the camera and 2D image.
SetUpButtons	Defines the layout and buttons. E.g. Displays 3D features, Displays 3D uncertainties, enables tracking, enables mapping, displays a 2D search region, initializes auto features.
ImageDraw3D	Creates a Feature Drawer based on the MonoSLAM class MonoSLAM FeatureDrawer to draw the feature and camera postion and uncertainties.
HandleNewFrameGlow	Handles a new frame that was received from the camera and copies it to a ImageMono object and initiates the update to the image display.
UpdateDisplayParameters	Updates the camera postition in the 3D view by querying the state vector from the camera motion model.
AutoInitialiseFeatureGlow	Initializes the features by the use of the MonoSLAM-Interface class and the Scene feature class.

Table 6.6: List of selected MonoSLAMGlow functions

6.3.5 VW35 Modification

Implementing the SwissRanger SR3000 as a camera sensor into the MonoSLAMFramework requires an enhancement of the VW35 hardware and interface libraries. Based on the VW-Firewire and VWGlow libraries a Shared Memory Library, a Shared Memory Sequencer and Sequencer Control Class have been developed. The Shared Memory Library consists of the Shared Memory and Semaphore Classes equal to Sec. 6.2.2. The Shared Memory Sequencer and Sequencer Glow GUI are derived from the related FireWire classes. Fig. 6.6 and Tab. 6.7 point out the relevant modifications.

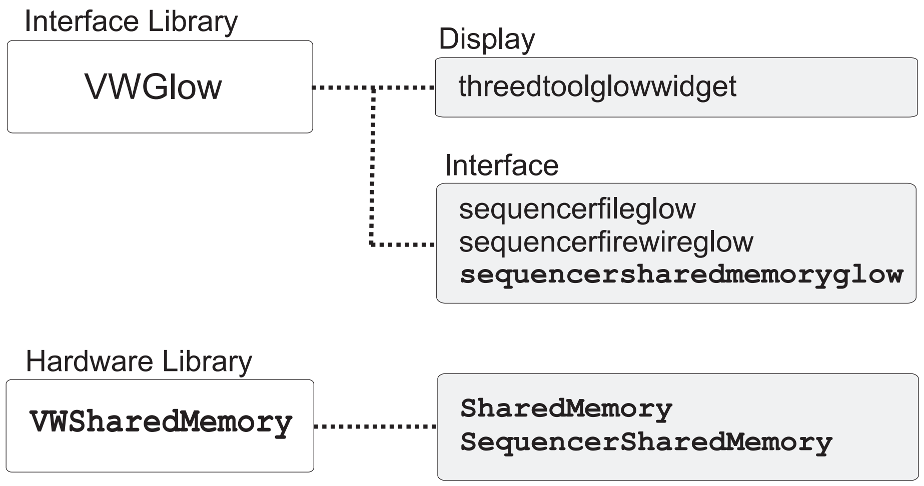


Figure 6.6: Overview of modified libraries and related classes.

SequencerSharedMemoryGlow	Create a Glow GUI control window for handling new images based on the SequencerSharedMemory class.
SequencerSharedMemory	Sequencer that grab images from the shared memory by use of the SharedMemory class E.g. Initialize Sequencer, set templet for memory structure. Copy 2D image - amplitude and gray scale values, copy 3D image (x, y, z coordinates) and 2D image
SharedMemory	Interface to a shared memory segment. Create, attach, delete shared memory. Manage access via semaphores (cf. Tab. 6.2)

Table 6.7: List of selected VW35 modifications

6.3.6 MonoSLAMGlow Modification

There are minor modifications to the MonoSLAMGlow code for the integration of the SR3000 camera. The HandelNewFrameGlow function has been adopted to copy the 3D image data into the application memory address range. Until now the gray scaled image data is used as measurement input into the monoslam interface. The 3D data is only visualized as point cloud in the 3D display with repect to the camera pose. Therefore the function SetUpButtons of the control panel and the MonoSLAM library class ThreeDDraw have been modified. Further processing of the x, y, z coordinates as measurement input has been considered for future enhancement.

Chapter 7

Results

7.1 Experimental Setup

Based on the hardware and software setup described in Chap. 6 an indoor office environment is chosen as the experimental setup scene. Beside the typical office objects e.g. monitor, books, desks, chairs, picture frames, additional targets haven been prepared to provide stable and unambiguous landmarks. The initial target frame is placed at the wall to set the 3D map coordinate system close to it. This means the camera depth is measured in relation to the wall, respectively to the target frame. A smooth and slow camera egomotion is carried out by hand and the Scene is observed along the opposing wall.

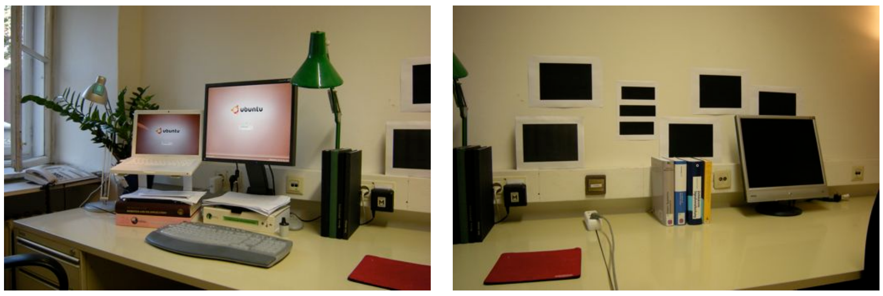


Figure 7.1: Overview of scene setup.

7.2 Test Procedures

This section provides a general overview of the performed test cases, to verify the server interface, the MonoSLAM interaction with the Swiss Ranger SR3000 camera, and as well the Fusion approach (cf. Sec. 5). The tests are grouped by the following three categories. Tab. 7.1, 7.2, 7.3 listing the performed procedures for each category.

1. Validation of interface functions
2. Validation of MonoSLAM functionality with SR3000
3. Validation of Fusion approach

7.2.1 Interface Functions

Subject	Description
Shared Memory Server	Creating, connecting, reconnect, deleting a shared memory segment, acquiring the image from the camera using the Shared Memory Server application. Testing principal functionality.
Camera Interface	Initializing the camera, setting predefined values for integration time, the amplitude threshold and auto illumination flag with the Shared Memory Server application. Testing camera control.
Shared Memory Client	Attaching, detaching from Shared Memory. Acquiring the image frame from the shared memory segment and printing the data values of amplitude and raw distance values, xyz-coordinates (each), and the 8-bit grayscale values to the output terminal with the Shared Memory Client application. Testing image grabbing and validating 2D and 3D image data.
Shared Memory Client Server Communication	Setup shared memory, with Shared Memory Server application and connect with Shared Memory Client application. Cartesian coordinates, and the 8-bit grayscale values to the output terminal. Testing asynchronous communication via semaphores.
VW Shared Memory	Creating, connecting, reconnecting, deleting a shared memory segment with the MonoSLAMGlow application. Testing principal functionality.
VW Sequencer Shared Memory	Attaching to and, detaching form the to Shared Memory. Acquiring the image frame from the shared memory segment and printing the data values of amplitude and distance values, Cartesian coordinates, and the 8-bit grayscale values to the output terminal with MonoSLAMGlow application. Testing image grabbing and display and validating 2D and 3D image data.

Table 7.1: List of interface test procedures (1).

7.2.2 MonoSLAM Functionality with SR3000

Subject	Description
2D display	Verifying and testing the displayed 3D map and 2D 8bit grayscale image frames. Evaluating different objects in a office environment.
Tracking	Testing feature initialization, feature detection and measurement and mapping as described in Sec. 4.3.
Parametrization	Testing scene with different camera parameters. Integration time, frequency, saturation threshold, amplitude threshold and auto-illumination.

Table 7.2: List of MonoSLAM functionality test procedures (2).

7.2.3 Fusion

Subject	Description
Structure Tensor	Acquire 2D and 3D image data and evaluating 3D corners with structure tensor. Detecting potential 2D and 3D corners.
3D Data Visualization	Evaluating 3D data points for different scenes.

Table 7.3: List of Fusion test procedures (3).

7.3 Evaluation

7.3.1 Shared Memory Server

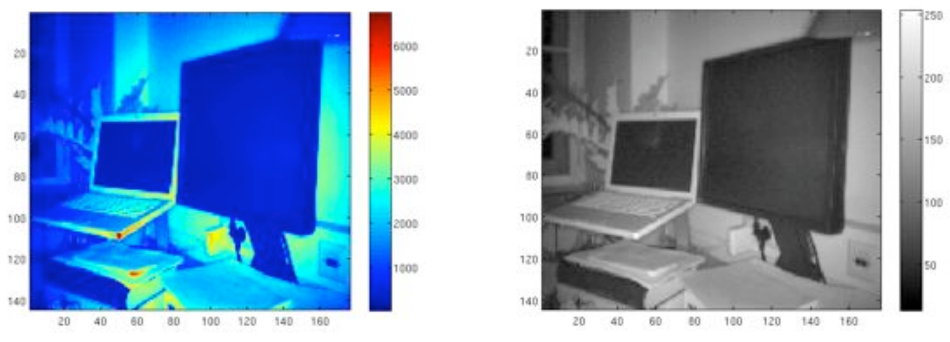
Creating, detaching, attaching and deleting to/from a shared memory segment is performed via a command line terminal. When starting the server application first the camera is initialized. This can be observed when the status LED of the camera turns from red to green. Subsequently images are acquired by the server. This can be verified by the terminal output, which displays the number of received frames and the calculated frames per second. Furthermore the camera status LED blinks faster, indicating successful image grabbing via the camera API. The shared memory segment is evaluated with a tool for Unix/Linux Inter-Process Communication (IPC). The tool (*ipcs*) provides information on the IPC facilities as shared memory ID, shared memory key, shared memory size in byte, and shared memory access rights. To stop the server the terminal control command is processed in the terminal window. The image frames stops immediately and the camera LED is flashing slower. Processing the query with *ipcs* illustrates that the shared memory still exists. Even when the camera is made currentless the server stops processing the query, but the shared memory is still active. In order to reconnect to the shared memory the server is restarted with the attachment option. The camera is initialized, the LED flashes faster and the terminal output displays again frame number and rate as before. To delete the shared memory the server stops with the terminal control command and afterwards is deleted with a delete option on the server application. Processing the query with ipcs show that the shared memory status provides no output for the previous defined SHM ID or SHM key.

7.3.2 Shared Memory Client Server Communication

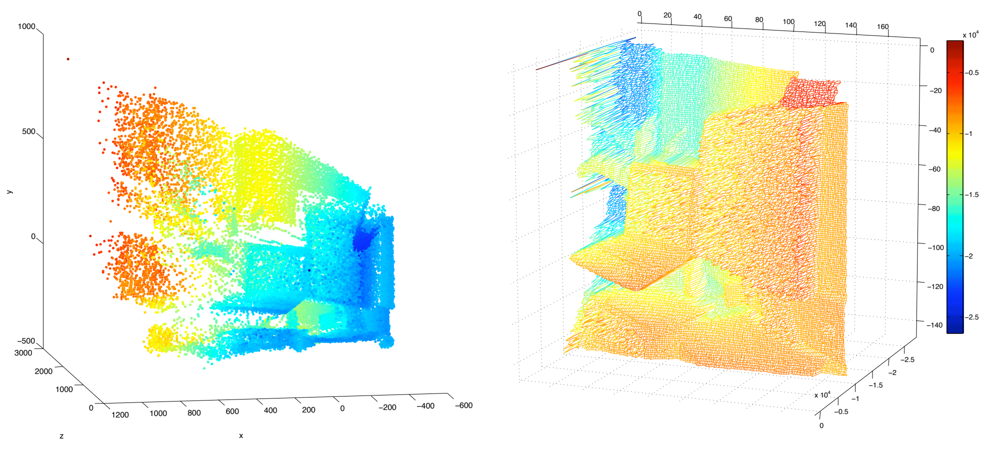
The communication between the server and client is controlled via semaphores. By starting the server one semaphore is created. The the *ipcs* tool provide the information on the SEM ID, SEM key and access rights. Testing the asynchronous communication the client starts within a second terminal in continuous output mode. As verified by standard monitoring tools, the CPU and memory usages of the computer remains stable (approx.30% for both cores). To delete the semaphore the server is stopped with the control command and afterwards deleted with a delete option of the server application. Processing the query with *ipcs* show that the semaphore status provides no output for previous defined semaphore ID.

7.3.3 Shared Memory Client

The Shared Memory Client attaches to the memory segment via a command line terminal and acquires immediately for each image frames the pixel values. Optional parameters defined, which data (amplitude, raw distance, x, y, z, and grayscale data) is printed as terminal output. For a number of test images each data output has been stored and visualized with MATLAB image processing toolbox for verification of correct data output.



(a) Amplitude (b) Gayscale



(c) xyz-coordinates (d) Raw-distance

Figure 7.2: Examples: (a) Amplitude image, [0-65536] with max. picture value 6752. (b) Grayscale image, normalized amplitude values, [0-255], max. picture value 254. (c) 3D plot with xyz-coordinates, [0-65536], units in millimeter. (d) Spherical raw-distance data over a mesh grid, [0-65536], max. picture value 26316 approx. 3m.

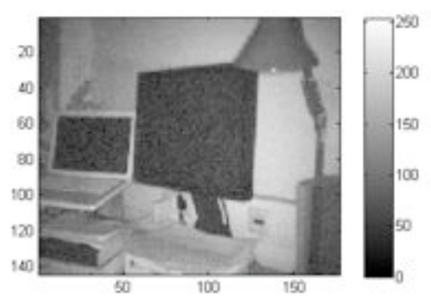
The plots in Fig. 7.2 of one image frame visualizes the different data values. The frame label in (a), (b) and the xy-plane in (c), (d) denote an image resolution of 176x144 pixel The colour bars indicate the maximum pixel value for each picture. The amplitude and distance are represented through a 16-bit value (unsigned integer) and therefore distributed along a wide range. By normalizing the amplitude values onto a maximal value of 8-bit (255) a grayscale representation is achieved.

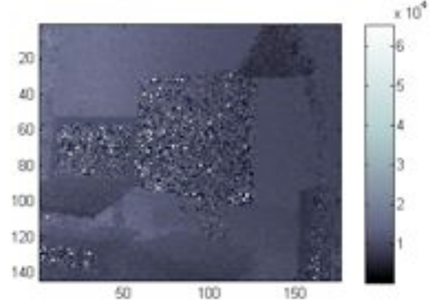
7.3.4 Camera Interface

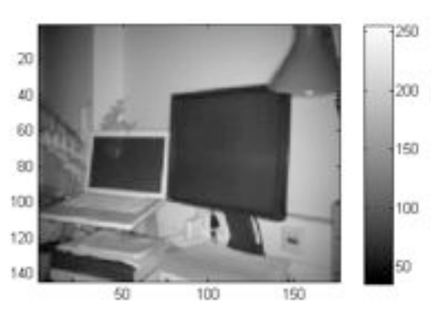
Following parameters values are set via the command line terminal as an optional parameter of a Shared Memory Server startup. For each parameter the data set of one image frame is provided by the Shared Memory Client command line application and visualized with MATLAB.

- integration time: 5, 20, 255
- amplitude threshold: 0, 100, 500
- modulation frequency: 19, 20, 21, 30
- saturation threshold: 0, 4000, 65532

Changing the integration time results in a higher exposure time as described in Sec. 3.2.2. The higher the value the longer the integration time of amplitude and distance data. For an integration time value of 5 the acquisition speed increases up to 46fps, with the cost of lower amplitude and distance values and an increased amplitude and distance noise. Increasing the integration time subsequently to 20 , 100, 255 lead to higher amplitude values and more precise data but to a decrease of the acquisition speed. Fig. 7.3 illustrates this effect on the grayscale and raw distance values. Note that a high value for the integration time introduces motion blur (cf. Sec. 3.3.5).


(a)


(b)


(c)

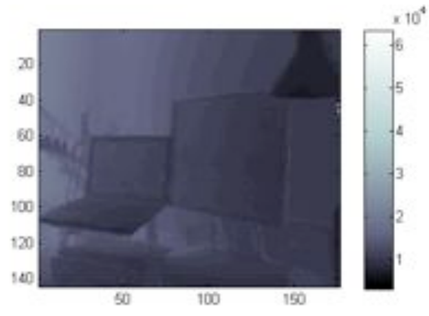

(d)

Figure 7.3: Example images for different integration times. (a) Grayscale image with integration time 0.5ms (46fps). (b) Raw distance plot as 2D intensity image (max. value 65532) depict clearly the effect of strong noise, due to low integration time. (c) Grayscale image with 51.2ms (4.5fps). (d) Raw distance plot as 2D intensity image (max. value 20392). For high integration time the noise is reduced significantly.

Modifying the amplitude threshold results in a cut off from low intensity values, which can be used to reduce amplitude and distance noise. However, the parameter value has to been selected with care, because all the values blow the threshold are mapped to zero and appear as a black pixel (lowerst intenstiy or maximal distance) in the image frame. Fig. 7.4 illustrates this effect for a threshold of 100 and 500.

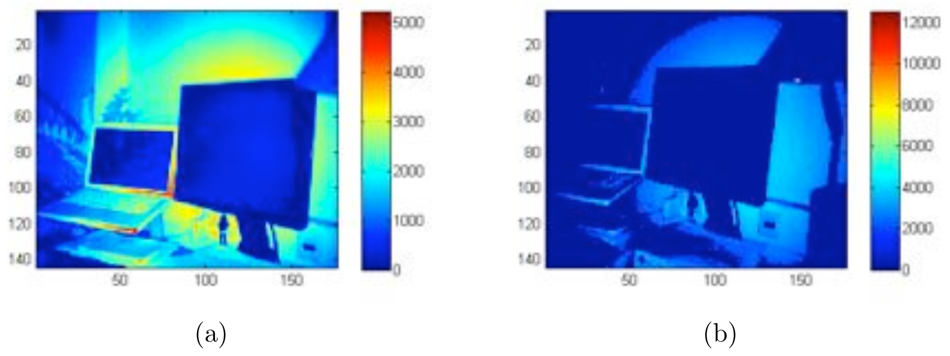


Figure 7.4: Example images for different amplitude thresholds. (a) Amplitude image threshold set to 100 with isolated pixel are set to zero (coloured dark blue). (b) Amplitude image threshold set to 500 with most of the pixel are below the threshold and therefore are coloured dark blue.

There are four explicite parameter values available to adjust the modulation frequency. As explained in Sec. 2.5.5 depends the non-ambiguity distance on the modulation frequency. Increasing the frequency results in a lower maximal distance, which corresponds to the distance values: 30MHz = 5.0m, 21 MHz = 7.1m, 20MHz = 7.5m and 19MHz = 7.9m. As denoted in Eq. 2.35 the measured distance is proportional to the non-ambiguois distance. This means that at a higher frequency, distance range values are more compressed, and thus provide a better resolution and distance accuracy as described in Sec. 3.2.3. Fig. 7.5 depicts the range scaling for different frequencies.

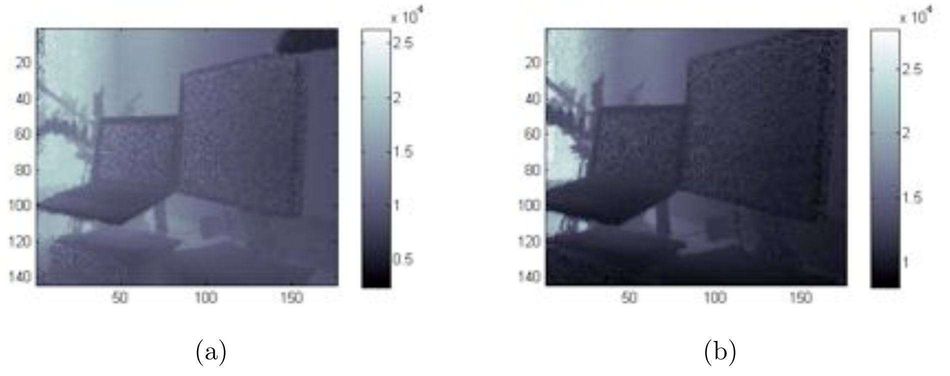


Figure 7.5: Example images for different modulation frequencies. (a) Raw distance plot with 20MHz. (b) Raw distance plot as 2D intensity image with 30MHz has a much more darker pixel with a wider value range.

Measurements obtained for saturation thresholds of 0, 4000 and 65532 had no effect on the outcome. Some threshold values provided senseless results. The changing of this parameter appeared useless, and therefore has not been considered for further analysis.

7.3.5 VW Shared Memory

The VW Shared Memory interface represents the integration of parts of the Shared Memory Client functionallity into the SceneLib framework. It is based on the same Shared Memory Class and perform attach, detach and acquire for each image frame via the the SR3000 API. Additionally it creates a shared shared memory segement on MonoSLAM startup and provides a function to delete the memory segment. As explained in Sec. 7.3.1 , successful creation is verified by the retrieved by IPC information. Also applying when the shared memory segment is deleted, which is stop via the Sequencer. The communication with the Shared Memory Server is similar to Sec. 7.3.3. In additional to the server create and delete command the VW client semaphore is verified by IPC information. For both applications (Shared Memory Server, MonoSLAM) it is verified by standard monitoring tools, for the used computer hardware (cf. Sec. 6.1.1) the CPU and memory usages of the computer remains stable (approx. 50% for both cores).

7.3.6 VW Sequencer Shared Memory

The Sequencer Shared Memory is started automatically as a new thread from the MonoSLAM application. It conrolls the image grabbing (start, stop, next, countinous) as well the loading of the amplitude, grayscale and xyz coordinate values. The functionallity of the sequencer is inherent verified by starting the MonoSLAM application. Fig. 7.6 displays a grayscale image and xyz data point cloud in the 3D map loaded from the shared memory by the sequencer. It is worth noting that the amplitude and raw distance values are not currently not used to updating the MonoSLAM algorithm (cf. Sec. 4.2). Instead the equivalent Cartesian coordiantes and normalized grayscale values are used.

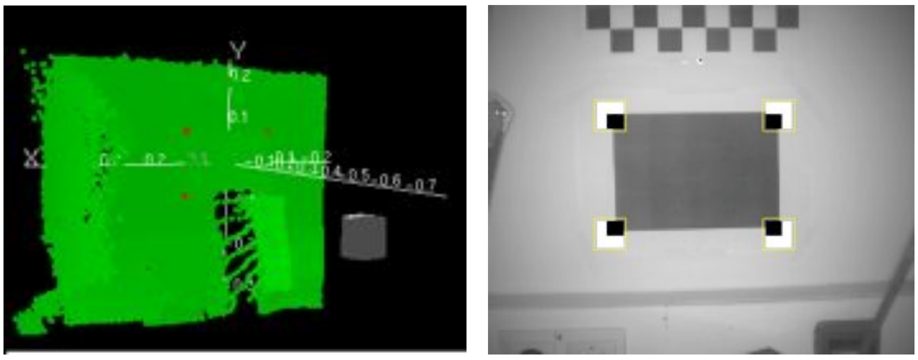


Figure 7.6: Loaded grayscale image and xyz coordinates on MonoSLAM startup.

7.3.7 2D and 3D Image Analyse.

Different test sequences are recorded in an indoor environment as illustrated in 7.1 to evaluate the displayed 2D and 3D image data. The MonoSLAM software has been enabled to save for every grabbed image the amplitude and xyz data values, as well as the grayscale image in a pgm format. This provides the possibiltity of further investigation via image proccessing toolkits e.g. MATLAB[1] or ImageMagick[2]. Fig.7.7 provides an outcome of recorded grayscale images.

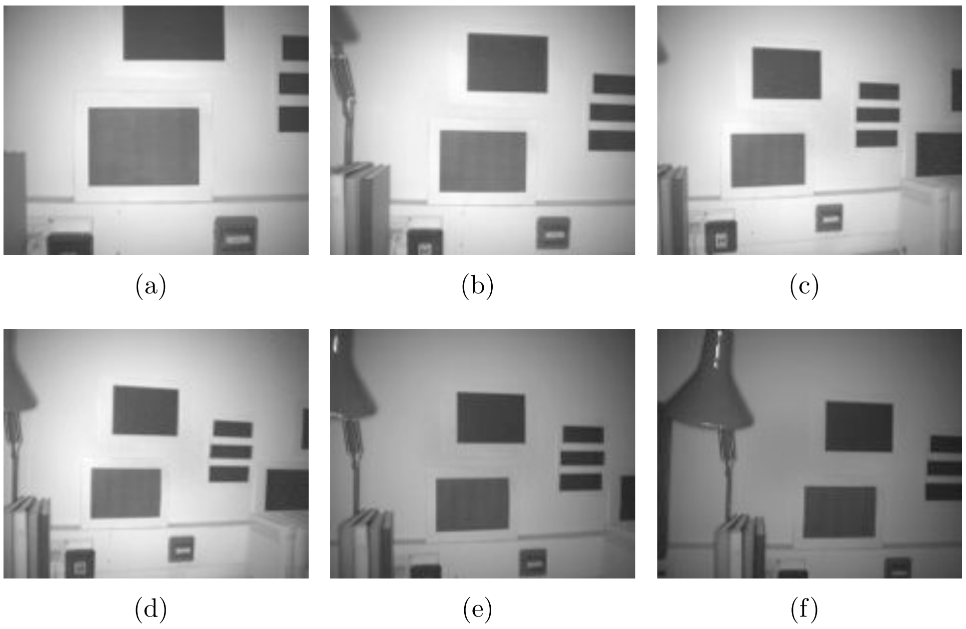
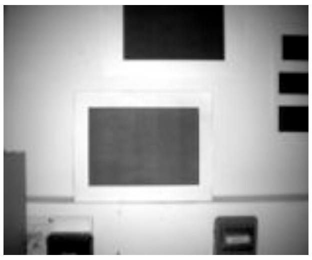


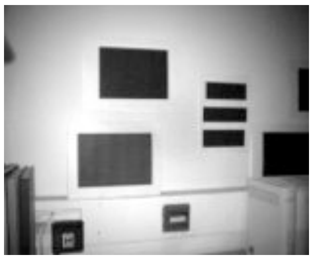
Figure 7.7: Selected images of a recorded scene. Default integation time value 35, modulation frequency 20 MHz.

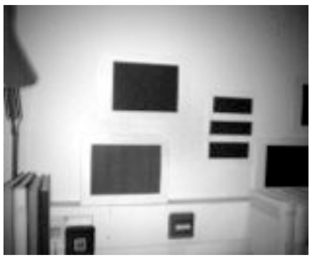
The picture depicts samples of the recorded scene which are evaluated below.The images in (a)-(d) show a small field of view compared to the scene setup in Fig. 7.1 taken by a digital photo camera. The illumination is not uniformly distributed along the image. The reason for that is the active lighting of the senor as described in Sec. 3.1.2. Regions towards the edges appear darker than in the centre of the image similarly to the illumination of a torch light. Pictures (c) and (d) appear with low contrast. This can be explained by the method used by the Shared Memory Server to convert the amplitude data into the grayscale values (cf. Sec. 6.2.4). All values are normalized onto the maximum intensity value for each image. In picture in (e) and (f) the image appears to be darker because maximum intenstiy corresponds to a total reflection on the lampshade centre. As observed in the images (a) and (b) the contrast increases as soon as the lampshade reflection disappears. To improve the image contrast the intenstiy values are strechted to cover the whole range of values. With black-out at most 2% of the pixels and white-out at most 1% of the pixels. Fig. 7.8 depicts the images after manipulating with ImageMagick.

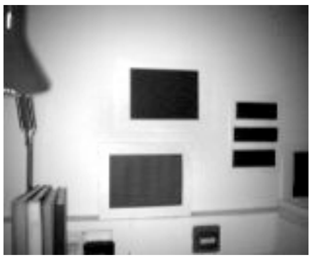
[1]MATLAB is a numerical computing environment and programming language created by MathWorks.
[2]ImageMagick® is a software suite to create, edit, and compose images.


(a)

(b)


(c)


(d)


(e)

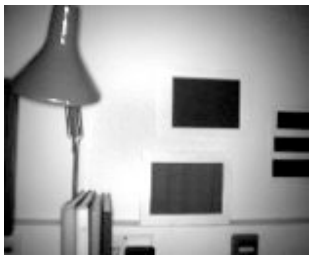
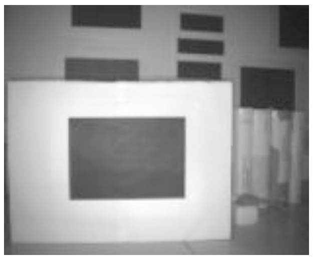

(f)

Figure 7.8: Selected images of recorded scene after normalization.

A similar effect to Fig. 7.7 occurs when an object is placed in front of the camera. The closer object absorbs most of the light from the LEDs whereas the background scene receives less illumination power. Fig. 7.9 provides an example for this effect. In this case a contrast modification does not improve the image as in Fig. 7.8.


(a) before normalization.

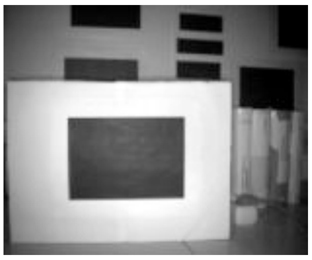
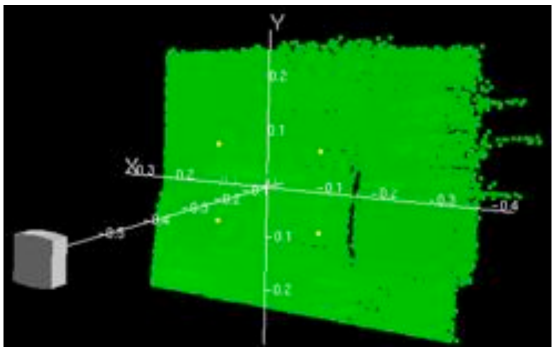

(b) after normalization

Figure 7.9: Image with target shadowing background sence illumination.

In the following the 3D image data of the corrosponding scene and selected images have been analysed. As illustrated in Fig. 7.10 one relevant effect that has been observed is the amplitude mixed pixel (cf. Sec. 3.3.2) and distance noise (cf. Sec. 3.2.3). The dark rectangles introduce strong photon shot noise, due to their low reflectivity and thus low backscattered optical energy (cf. Sec. 3.3.5). An example is highlighted at the dotted rectangles in Fig. 7.10. Scattering has been found at the change over from the white to black borders. It occurs in particular in the black rectangles surrounded by the white frames. This effect can be explained due to the difference in target reflectivity (cf. Sec. 3.2.4) of white and black objects. Because of the differences in the received optical energy the "black" pixels are more influenced by the scattered light of their neighboring "white" pixels. This leads finally to a difference in distance measurments. It is worth noting that this effect changes dynamically with camera motion and the observed environment. Therefore it is very difficult to determine each effect as introduced in Sec. 3.3 without a comprehensive analysis.



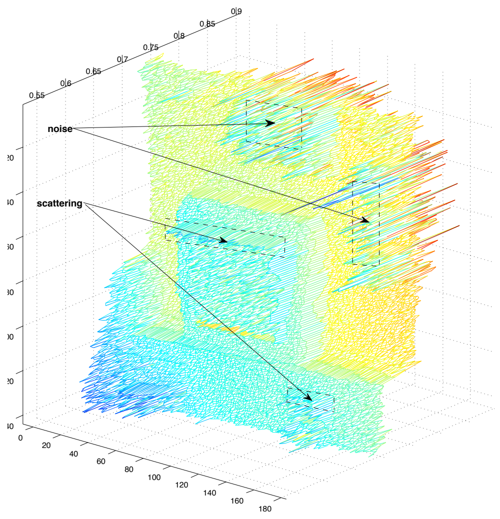


Figure 7.10: Point cloud and distance mesh grid example cooresponds to Fig. 7.7 (a)

7.3.8 Feature Tracking

Feature detection, initialization and active measurement has been tested according to the procedure in Sec. 4.3. Within this work it was only possible to integrate 2D information as measurement information into the MonoSLAM algorithm. Future investigation should evalute the use of 3D information as a measurement update within the MonoSLAM framework. Fig. 7.11 illustrates the results of a recorded scene where tracking has been enabled.

Fig. (a)-(c) depicted the detection, initialization and aktive measurement of features. The number of detected features is restricted to close objects. A small field of view constrains the active measurement of features within a larger range of camera motion. To compensate that effect and to increase the feature detection rate, explicit targets have been aranged closed to the initial target. Further distance to the objects and a camera horinzontal viewing angle have been increased during motion, to keep features visible. Fig. (c)-(f) illustrates that the postion is lost if no new feature has been initialized, which restricts the motion range of the camera. Potential natural landmarks e.g. power or LAN sockets on cable shaft at the wall, are too small and have far less contrast quality for detection. This effect has been observed with likewise increasing distance. The illumination contrast decreases and lower amplitude values add additional noise Sec. 3.2.3.

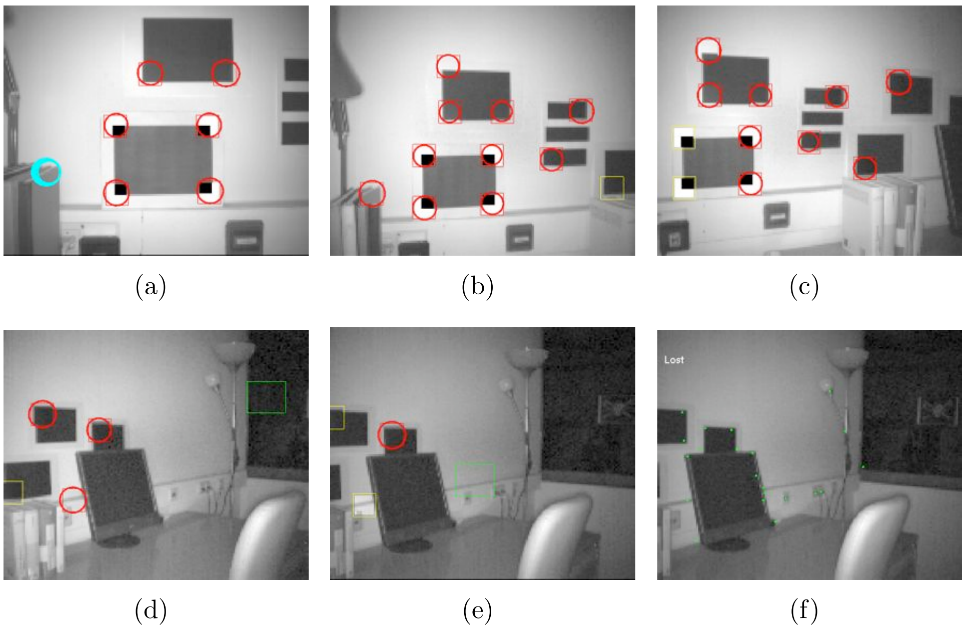
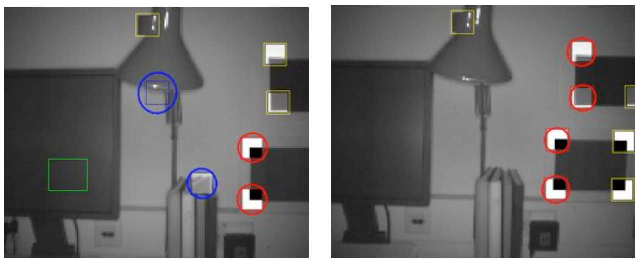


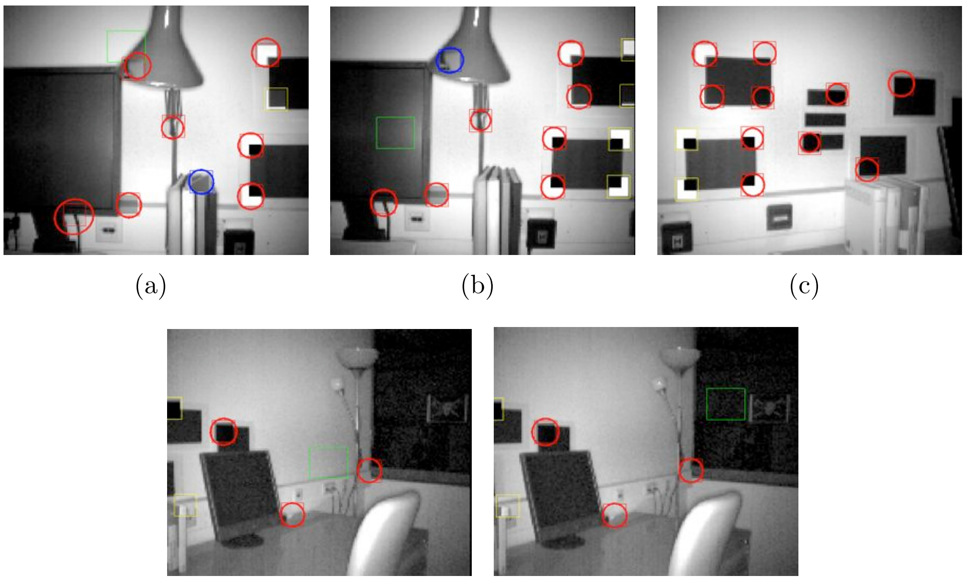
Figure 7.11: Selected images of the recorded scene with tracking enabled. (a) Feature intialization and measurement after startup. (b) Increase distance and 45° viewing angle. (c) Not all feature could be active measured due to restricted field of view. (d) Reduced feature set at 50° viewing angle. (e) One aktive measured feature at 60° viewing angle. (f) Tracking lost due to inceased motion uncertainy and reduced measured features.

A nother effect with respect to the illumination is the measurement error when the image contrast suddenly changes as illustrated in Fig. 7.12. As described in Sec. 7.3.7 does the picture contrast decreases when a high reflected object enters visibility. This illumination change can cause observation errors due to image patches cross-correlation search(cf. Sec. 4.2.6). In oder to improve the feature detection rate the grayscaled images have been modified according to Sec. 7.3.7. It could be shown that this correction provides a postive effect on feature detection and stabilizes the active feature measurement. Fig. 7.13 shows the tracking results before and after the image processing.



(a) (b)

Figure 7.12: Selected images of reorded scene with feature measurement errors. (a) Blue feature with detected measurement errors. (b) Subsequent frames, features are deleted.



(a) (b) (c)

(d) (e)

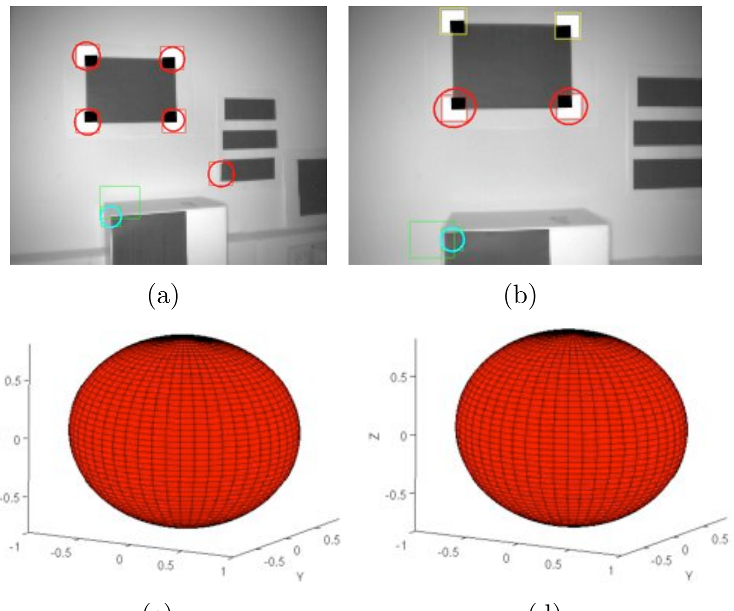
Figure 7.13: Selected images of recorded scene after contrast enhancement with tacking enabled. (a) One feature detected with measurement errors but additional features are detected. (b) Subsequent frames, one feature is deleted, other features remain the same. (c) One feature more detected compared to Fig. 7.11. (d) Compared to 7.11 one more feature detected. (e) Compared to Fig. 7.11 is tracking still possible.

7.3.9 Parametrization

To determine the approximate optimal set of configuration parameter for the operation with the MonoSLAM algorithm. Tests with different integration times, modulation frequencies and amplitude thresholds have been conducted. Based on the experimental scene as depicted in Fig. 7.1 an integration time value of 35, a modulation frequency of 30MHz, and amplitude threshold value of 5 have provided practicable findings. An integration time value of 35 corresponds to a frame rate of 20fps. It been has shown to be a well balanced value between the maximum possible frame rate to improve the MonoSLAM measurement update and the maximum integration time, while still providing a low signal to noise ratio. Changing the modulation frequency to 30 MHz increases the the depth accuracy and thus the 3D meassurement data. The slightly increased amplitude threshold limit the low amplitudes which can cause strong noise. The effect on the 3D data is rather moderate. Autoillumination has shown to be in principal a good feature due to is ability to provide good illumination while moving through the scene. However, its major downside is the variable frame rate from 5fps up to 46fps, that is not applicable for the MonoSLAM algorithm which requires a nearly constant frame rate. Furthermore a low frame rate can introduce motion blur (cf. Sec. 3.3.5) which would lead to wrong measurement data for feature detection and measurement.

7.3.10 3D Corners

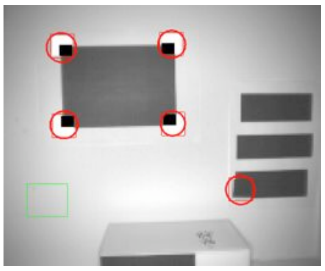
To identify 3D corners the Structure Tensor concept as described in Sec. 5.3.1 has been evaluated. Fig. 7.14, 7.15 illustrates the selected images with detectable corners and their corresponding structure tensor visualization. An image patch with the size of 11x11 matrix has been extracted with the corresponding grayscale values. For this data matrix the structure tensor matrix was calculated within MATLAB. The eigen decompostion of this matrix calculates the eigenvalues and eigenvectors which have been drawn in the following. Fig. 7.14 illustrating the detection of 3D corners for two different viewing directions. As shown in Fig. 7.14 (a) a green detection box overlaps a real corner in its centre point. Within Fig. 7.14 (b) the detection box cover mostly the black plane, with some small overlaps on the edge. For comparison detected features in the plane have been analysed.

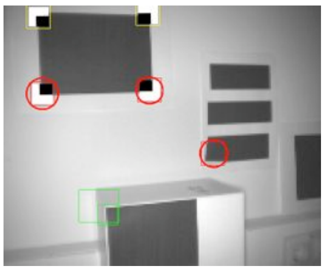


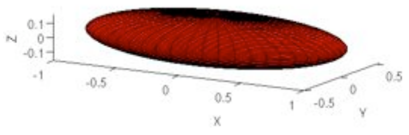
(a) (b) (c) (d)

Figure 7.14: Selected images with 3D corners

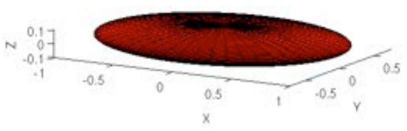
Fig. 7.15 illustrates the detection of two features. One at the wall and the second at an object infront of the wall. As shown in Fig. 7.15 (a) the active measured feature patch on the right lies on a wall. The corresponding eigenvalue visualization (c) shows a ellipsis that is similar to the step-plan representation in Sec. 5.3.1. Within Fig. 7.15 (b) the detection box covers half of the detected corner. The corresponding eigenvalue visualization (d) represents ellipsis similar to (b). This is surprising, and one might expect a sphere instead and thus a 3D corner detected. However, the extracted path seems to be more similar to a step-plane case.


(a)


(b)



(c)



(d)

Figure 7.15: Selected images with 2D corners.

7.4 Findings

The integration of a 3D camera within a Visual SLAM framework has been successfully shown and the impact of the sensor characteristics on the MonoSLAM algorithm has been analyzed. The investigation reveals that the illumination and contrast of the acquired 2D image depends greatly on the chosen integration time, the distance, the scene and to some extent on the image processing. It has been verified that low and fluctuating illumination and low contrast gain has a negative effect on the feature detection rate and can also increase measurement errors. According to conducted tests this limits the ability of the camera to distances in the range from approximately 1-3 metres. The restrictions depend mainly on the chosen integration time of the sensor. A high integration time provides a good illumination but can cause a frame rate that is too slow for the MonoSLAM algorithm. On the other hand a low integration time introduces noise, that can handicap the feature detection. The auto illumination function, which in general provides a good automatic control of the illumination, but varies strongly in its frame rate is therefore incompatible with the algorithm. The major disadvantage of the camera is its small field of view. The results show that the MonoSLAM algorithm only performs well in an environment with close good features to track. In case new features are not detectable the algorithm becomes unstable and loses the camera and feature position. With a small field of view the feature tracking has proven to be difficult with respect to the camera handling. The motion speed has been slowed down with motion stops in order to keep the feature visible longer. This contradicts the underlying motion model, which supposedly has a constant velocity with additive Gaussian noise and cause unexpected instabilities with the lost in map and camera position.

The visualization results of the 3D data show the strong influence of interference effects such as scattering, mixed pixels and also the increasing noise for low amplitude values, this is due to a longer distance or less reflected light. An increase of the modulation frequency results in a better distance resolution and range accuracy but reduces the maximal measurable distance. Even under these limitations the obtained distance is considered to be applicable for the feature initialization in the MonoSLAM algorithm. Future investigation shall prove the implementation of this concept as described in Sec. 5.2 and provide performance comparison with the current particle initialization. Due to the transformation of the unknown distance to the measured distance with a small range of uncertainty, it is expected that the feature is initialized faster as with a full distance range.

In this work the structure tensor concept (cf. Sec 5.3.1) has been tested to obtain 3D corners. Results show that corners can be identified by its visualization of eigenvalues and eigenvectors but accuracy of the match is dependent on the detected image region with respect to the sensors distance noise. It has been shown that corners in the centre point of the detected region are more distinct than corners towards located towards the edges of the detection box. Future investigations shall analyze a optimal detected image region for evaluating 3D corners.

A straight forward improvement of the raw data conversion from 16-bit to 8-bit data, with standard image processing concepts such as histogram equalization and gamma correction, can enhance the quality of the 2D image for feature detection. This can either be a feature of the camera itself, or done in software before using the image as measurement input to Visual SLAM. Improving the camera optic to extend the field of view and the increase of the pixel array provide the most contribution to the feature detection rate and thus the usability for Visual SLAM. Increasing the camera frame rate further without affecting the integration time would provide a better performance of the MonoSLAM algorithm.

Chapter 8

Summary and Further Work

8.1 Summary

The theory chapter built a set of fundamental basic concepts, beginning at the very principal of random variables and distributions up to the definition of Bayesian probability, which is an important concept in probabilistic robotics. Based on that the belief representation denoted as posterior probability has been derived to estimate the robot state and position. Furthermore the robot has been modelled as interacting with its environment and is described by structured dynamic systems. Robots beliefs about the environmental state, the received measurement data and robot controls have been introduced as important characteristics of this model. Following the concept of Bayesian filtering, the Kalman Filter and its successor the Extended Kalman Filter (EKF) explains how the robot can calculate the belief about its state effectively with this algorithm. Based on that it has been explained that robot motion and sensor perception always needs to incorporate errors, therefore uncertainty associated with them is addressed in a motion and measurement model by probabilistic terms. In that context feature extraction is explained as useful method for object recognition and tracking. Finally basic solutions to localization and mapping, respectively Simultaneous Localization and Mapping Problem (SLAM) based on the EKF algorithm have been explained.

The chapter Time-of-Flight introduces the concept of range image cameras that provides a 2D image and additional 3D information due to distance measurements. The SwissRanger SR3000 is described as an implementation for this kind of sensor. Beginning with the sensors measurement principle based on modulated light that is sent out by a signal emitter, partially reflected by an object and mapped onto a sensor array by optics. Following the influence of the integration time and the determination of the distance accuracy, where quantum shot noise is a limiting factor, has been described. Further it has been highlighted that different targets reflectivity, respectively different demodulated amplitudes have different effects on the distance measurement. Finally the main disturbing effects are explained, such as scattering, mixed pixels and multiple reflection.

The chapter Vision SLAM provide an brief overview of solutions to camera based Simultaneous Localization and Mapping problems. In particular the MonoSLAM algorithm of Davison [11] has been explained in more detail. It is based on a single camera as sensor; using an EKF based algorithm to estimation the camera and map feature positions. It proposes a technique of active feature measurements and mapping, the use of a general motion model for smooth camera movement, and solutions for monocular feature initialization and feature orientation estimation.

The chapter Fusion proposes an appoarch that uses the Time-of-Filght camera generated distance and 3D data within the MonoSLAM algorithm, instead of using a 2D image to reconstruct the depth and 3D structure information.

The chapter Implementation has provided an overview of the used hardware and software. The MonoSLAM Framework SceneLib 1.0 for real-time SLAM and mapping and its basis the computer vision library VW35. The Scene platform provides general algorithms and methods to build SLAM applications. A brief description of important Scene and VW35 libraries has been listed. Finally the modification of the VW35 and MonoSLAM application with respect to the camera sensor SR3000 has been explained.

The chapter Results has provided a description of the performed testcases to verify the developed interface, the MonoSLAM interaction with the Swiss Ranger SR3000 camera, as well the Fusion approach. In the following the influence of the sensor properties onto the localization and mapping process has been evaluated and how they benefits from the measured distance information. The Results have showed that illumination and contrast of the acquired 2D image depend greatly on the chosen exposure time, the object distance, and the recorded scene and to some extent on the used image processing. Low and fluctuating illumination and low contrast gain has a negative effect on the feature detection rate and can also increase measurement errors. The major disadvantage of the camera is its small field of view. The results have shown that the MonoSLAM algorithm only performs well in an environment with close and good features to track. In case new features are not detectable the algorithm gets instable and loses the camera and feature position. This limits the ability of the camera to distances in the range from approx. 1-3 metres. The results of the visualization of the 3D data have shown that the strong influence of interference such as scattering, mixed pixels and as well the increasing noise for low amplitude values. An increase of the modulation frequency results in a better distance resolution and range accuracy but reduces the maximal measurable distance. In this work the concept of structure tensors has been tested to obtain 3D corners. Results have shown that corners can be identified by its visualization of eigenvalues and eigenvectors but accuracy of the match depends on detected image region with respect to the sensors distance noise. Even under these limitations the obtained distance is considered to be applicable for the feature initialization in the MonoSLAM algorithm. On conceptual basis an enhanced feature initialization method that use measured distance for depth calculations has been discussed.

8.2 Further Work

In the following, some possible improvements and considerations for future investigations will be given.

Future investigation shall prove if the measured depth information of the ToF camera can be incorporated with the existing MonoSLAM algorithm to improve the feature initialization. The particle filter initialization introduced by Davison (cf. Sec. 4.2.5) assumes that the feature lies along a semi-infinity line from the camera to the viewing direction of the feature. To determine them a set of discrete depth hypothesis are uniformly distributed along that line, and begin after several image frames to converge to a Gaussian distribution with the mean representing the most likely feature distance. The direct measured depth of the ToF camera would allow shrinking the range of the particle filter hypotheses and thus allowing the initialization duration to decrease significantly.

A further improvement of the MonoSLAM algorithm is to update the feature position directly with the measured depth, during the tracking of active measured features. Thus, the feature measurement model can be directly inverted to update the feature position in the 3D map. Given the image of a perspective-projective camera only, inverting is not possible due to the unknown distance.

The visualization of 3D corners with its eigenvalue and eigenvector decomposition has been presented. To enable the MonoSLAM algorithm to detect 3D corners a cornerness measure need to calculated. Therefore an operater needs to be selected to determine a threshold for the 3D corner detection. A survey on 3D operators is presented by Rohr [34] that could be used for further investigations.

Distance measurements are strongly influenced by interference effects such as scattering, mixed pixels and backscattered light with low amplitudes. This information can be partially improved by filtering invalid data points. As presented by May et al. [22] the application of Gaussian blurring and jump edge filtering can remove those interference effects and smoothing the 3D image. This would improve the measurement data for the 3D corner detection.

A different approach is to implement another type of Time-Of-Flight camera sensor. Recent developments provide active 3D cameras that measure the distance by implementing a fast image shutter in front of a CCD sensor and blocking the incoming light. The collected light by each pixel is inversely proportional to the depth of the specific pixel [17]. The advantage of this solution is the usage of a standard CCD sensor as well as standard camera optics. This certainly would provide a bigger field of view, a better image quality with sufficient frame rates for good 2D feature detection because additional to the depth information the 2D image would be based on regular environment illumination. A new standard RGB 1.3 M-Pixel webcam with potential low price and stated distance depth resolution of 1-2 cm is announced by 3DV Systems [1].

List of Abbreviations

2D	two-dimensional
3D	three-dimensional
GUI	Graphical User Interface
LASER	Light Amplification by Stimulated Emission of Radiation
PC	Personal Computer
NASA	National Aeronautics and Space Administration
DARPA	Defense Advanced Research Projects Agency
IEEE	Institute of Electrical and Electronics Engineers
SLAM	Simultaneous Localization and Mapping
RIM	Range Image
ToF	Time-of-Flightl
API	Application Programming Interface
USB	Universal Serial Bus
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
RBPF	Rao-Blackwellised Particle Filter
CCD	Charge Coupled Device
CMOS	Complementary Metal Oxide Semiconductor
LED	Light Emitting Diode
ACDC	Analog Digital Converter
FPS	Frames Per Second
RAM	Random access Memory
IP	Internet Protocol
TCP	Transmission Control Protocol
SHM	Shared Memory
SEM	Semaphore
OpenGL	Open Graphics Library
GLUT	OpenGL Utility Toolkit
GLOW	OpenGL Object-oriented Windowing toolkit
SIFT	Scale Invariant Feature Transform
CSV	Computer Stereo Vision
SFM	Structure from Motion
H	Horizontal

Bibliography

[1] 3DV Systems, 2nd Carmel St. Industrial Park Building 1 P.O.Box 249 Yokneam, 20692 Israel. *www.3dvsystems.com/.*

[2] Active Vision Lab University of Oxford. *VW : Source-code documentation 0.35.*

[3] Active Vision Lab University of Oxford. *The VW Libraries*, 2003.

[4] S. Arseneau and J. Cooperstock. An asymmetrical diffusion framework for junction analysis. In *British Machine Vision Conference*, 2, pages 689–698, September 2006.

[5] Karl Johan Åström and Richard M. Murray. *Feedback Systems An Introduction for Scientists and Engineers.* Princeton University Press, 2008.

[6] D. Chekhlov, M. Pupilli, W. Mayol, and A. Calway. Robust real-time visual slam using scale prediction and exemplar based feature description. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–7, June 2007.

[7] A.J. Davison and N. Kita. 3d simultaneous localisation and map-building using active vision for a robot moving on undulating terrain. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 1:I–384–I–391 vol.1, 2001.

[8] A.J. Davison and D.W. Murray. Simultaneous localization and map-building using active vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):865–880, Jul 2002.

[9] Andrew J. Davison. Slam with a single camera. In *SLAM/CML Workshop Tutorial.* IEEE International Conference on Robotics and Automation, ICRA, May 2002.

[10] Andrew J. Davison. *Models and State Representation in Scene: Generalised Software for Real-Time SLAM.* Robotics Research Group Department of Engineering Science Universtiy of Oxford, May 2008.

[11] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[12] Andrew J. Davison and Paul Smith. *http://www.doc.ic.ac.uk/ ajd/index.html.* Department of Computing Imperal College London, 180 Queen's Gate SW7 2AZ, UK.

[13] C. Drocourt, L. Delahoche, B. Marhic, and A. Clerentin. Simultaneous localization and map construction method using omnidirectional stereoscopic information. *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 1:894–899 vol.1, 2002.

[14] H Durrant-Whyte and T Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics and Automation Magazine*, 13(2):99–110, 2006.

[15] P. Gemeiner, W. Ponweiser, P. Einramhof, and M. Vincze. Real-time slam with a high-speed cmos camera. *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*, pages 297–302, Sept. 2007.

[16] S.A. Guomundsson, H. Aanaes, and R. Larsen. Environmental effects on measurement uncertainties of time-of-flight cameras. *Signals, Circuits and Systems, 2007. ISSCS 2007. International Symposium on*, 1:1–4, July 2007.

[17] Ronen Gvili, Amir Kaplan, Dr. Eyal Ofek, and Dr. Giora Yahav. Depth key. In *SPIE Electronic Imaging 2003 Conference*, volume 5006, pages 564–574, 2003.

[18] Timo Kahlmann. *Range Imaging Metrology: Investigation, Calibration and Development.* PhD thesis, ETH Zurich, 2007.

[19] Ho-Duck Kim, Dae-Wook Kim, and Kwee-Bo Sim. Simultaneous localization and map building using vision camera and electrical compass. *SICE-ICASE, 2006. International Joint Conference*, pages 5915–5918, Oct. 2006.

[20] Robert Lange. *3D Time-of-flight distance measurement with custom solid-state image sensor in CMOS/CCD-technology.* PhD thesis, Department of Electrical Engineering and Computer Science at University of Siegen, 2000.

[21] Robert Lange, Peter Seitz, Alice Biber, and Rudolf Schwarte. Time-of-flight range imaging with a custom solid state image sensor. In Hans J. Tiziani and Pramod K. Rastogi, editors, *Laser Metrology and Inspection*, volume 3823, pages 180–191. SPIE, 1999.

[22] Stefan May, David Droeschel, Dirk Holz, and Christoph Wiesen. 3d pose estimation and mapping with time-of-flight cameras. In *Workshop on 3D Mapping.* IEEE International Conference on Intelligent Robots and Systems (IROS 2008), Frauenhofer Institute for Intelligent Analysis and Information Systems (IAIS), September 2008.

[23] Mesa-Imaging AG. *SR-3000 Data Sheet.*

[24] Mesa-Imaging AG, Technologieparkstrasse 1 8005 Zuerich Switzerland. *www.mesa-imaging.ch.*

[25] Mesa Imaging AG. *Swissranger 1.0.8.x Miniature 3D time of flight camera*, 2007.

[26] M.J. Milford and G.F. Wyeth. Single camera vision-only slam on a suburban road network. *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3684–3689, May 2008.

[27] David Murray and Ian D. Reid. *http://www.robots.ox.ac.uk/ActiveVision/*. University of Oxford Active Vision Group.

[28] T. Oggier, M. Lehmann, R. Kaufmann, M. Schweizer, M. Richter, P. Metzler, G. Lang, F. Lustenberger, and N. Blanc. An all-solid-state optical range camera for 3d real-time imaging with sub-centimeter depth resolution (swissranger). In L. Mazuray, P. J. Rogers, and R. Wartmann, editors, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 5249 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 534–545, February 2004.

[29] Thierry Oggier. *SwissRanger SR-3000 Manual V1.04*. Mesa Imaging AG, MESA Imaging AG Technoparkstrasse 1 CH 8005-Zürich www.mesa-imaging.ch, March 2007.

[30] Thierry Oggier, Bernhard Büttgen, and Felix Lustenberger. Swissranger sr3000 and first experiences based on miniaturized 3d-tof cameras. Technical report, Swiss Center for Electronics and Microtechnology. CSEM, 2005.

[31] Athanasios Papoulis and S. Unnikrishna Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill Higher Education, 2002.

[32] Peyton Z. Peebles Jr. *Probability, Random Variables, And Random Signal Principles*. McGraw-Hill Book Company, 1987.

[33] Mark Pupilli and Andrew Calway. Real-time visual slam with resilience to erratic motion. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:1244–1249, 2006.

[34] K. Rohr. On 3d differential operators for detecting point landmarks. *Image and Vision Computing*, 15(3):219–233, 1997. Cited By (since 1996): 33.

[35] Thorsten Schmitt. *Vison-based Probabilistic State Estimation for Cooperating Autonomous*. PhD thesis, Technische Universität München, 2004.

[36] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR94)*, pages 593–600, 1994.

[37] Roland Siegwart and Illah R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. The MIT Press, 2004.

[38] R. Sim, P. Elinas, M. Griffin, A. Shyr, and J.J. Little. Design and analysis of a framework for real-time vision-based slam using rao-blackwellised particle filters. *Computer and Robot Vision, 2006. The 3rd Canadian Conference on*, pages 21–21, June 2006.

[39] Robert Sim and James J. Little. Autonomous vision-based exploration and mapping using hybrid maps and rao-blackwellised particle filters. *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2082–2089, Oct. 2006.

[40] Paul Smith. *Scenelib Documentation 0.9.* Robotics Research Group Department of Engineering Science Universtiy of Oxford.

[41] Paul Smith. *Single-camera SLAM using the SceneLib library.* Robotics Research Group Department of Engineering Science Universtiy of Oxford, Nov 2005.

[42] M. Svedman, L. Goncalves, N. Karlsson, M. Munich, and P. Pirjanian. Structure from stereo vision using unsynchronized cameras for simultaneous localization and mapping. *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3069–3074, Aug. 2005.

[43] S. Takezawa and G. Dissanayake. Simultaneous localisation and mapping problems in indoor environments with stereovision. *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*, pages 6 pp.–, Nov. 2005.

[44] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics.* The MIT Press, 2005.

[45] J. Weingarten and R. Siegwart. Ekf-based 3d slam for structured environment reconstruction. *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3834–3839, Aug. 2005.

[46] G. Zunino and H.I. Christensen. Simultaneous localization and mapping in domestic environments. *Multisensor Fusion and Integration for Intelligent Systems, 2001. MFI 2001. International Conference on*, pages 67–72, 2001.