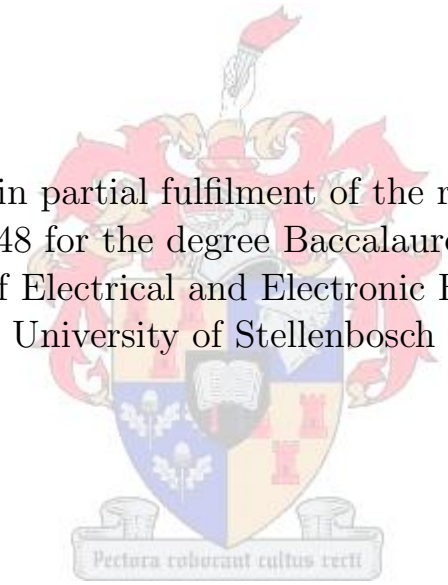
Monocular Vision Based SLAM Using Kinematic State Estimation

by

Aidan Russel Landsberg



Report submitted in partial fulfilment of the requirements of the module Project(E) 448 for the degree Baccalaureus in Engineering in the Department of Electrical and Electronic Engineering at the University of Stellenbosch

Department of Electrical and Electronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Supervisor: Dr. C.E. (Corné) Van Daalen

May 2015

Summary

Opsomming

Acknowledgements

I would like to express my sincere gratitude toward the following individuals for their role in making this project a success:

- My heavenly Father, for providing me with the intellectual capacity, guidance and support necessary to make a success of this project while remaining faithful and true in the toughest of times.

- My study leader, Dr. Corné Van Daalen, for his endless enthusiasm, guidance, patience, support, time and invaluable insight. As well as for personally setting aside the time to propose and supervise this project.

- My parents, for their endless support and motivation. As well as for making all the necessary sacrifices to provide me with the opportunity to complete this project successfully.

- Mr. Arno Barnard, for his advice regarding the embedded software design.

- My girlfriend Bianca La Gorcé, for supporting me endlessly.

- Benjamin Pannell, for being a great personal mentor and friend. As well as for providing me with insight regarding various software concepts.

- Luca Duesimi, for aiding in the design and construction of the stability platform.

- Warren Farmer, Kurt Coetzer and Lowku Leeuwenaar, for helping construct the stability platform.

Declaration

I, the undersigned, hereby declare that the work contained in this report is my own original work unless indicated otherwise.

Signature..... Date.....

Contents

List of Figures

List of Tables

Acronyms

2D	Two-dimensional
3D	Three-dimensional
CMOS	Complementary Metal-Oxide Semiconductor
EKF	Extended Kalman Filter
GPIO	General-purpose input/output
IMU	Inertial Measurement Unit
KF	Kalman Filter
I2C	Inter-Integrated Circuit Bus
ISR	Interrupt Service Routine
LIDAR	Light Detection and Ranging
MonoSLAM	Monocular Simultaneous Localisation and Mapping
PDF	Probability Density Function
PTAM	Parallel Tracking and Mapping
RV	Random Variable
SIS	Sequential Importance Sampling
SLAM	Simultaneous Localisation and Mapping
USB	Universal Serial Bus

List of Symbols

W	Inertial reference fame
C	Camera's free coordinate body frame
ΔT	Sampling instance
π	Constant denoting the ratio between a circles radius and its circumference
$\boldsymbol{\mu}$	Mean vector of a Gaussian random variable
$\boldsymbol{\Sigma}$	Covariance matrix of a Gaussian random variable
$\mathbf{I}$	Identity matrix
$\boldsymbol{\omega}$	Angular Rate (expressed in radians per second)
$\mathbf{R}^{CW}$	Rotation matrix projecting an entity from the body frame to the inertial frame
$\mathbf{C}$	Camera calibration matrix
f	Focal length of camera
k_u	Focal length normalisation constant
k_v	Focal length normalisation constant
u_0	Principal Point x-coordinate
v_0	Principal Point y-coordinate
r	Radial Distortion Parameter

Notation

Notation	Entities
x	Lower case italic text represents a scalar
x^W	Superscripts represent the coordinate frame (e.g. W or C)
x^T	Superscript T however represents the transpose
x_t	Subscripts bind a values to a specific instance (e.g. time or feature)
$\bar{x}$	Overscore text represent an estimate
$\|x\|$	A modulus denotes the absolute value
$\|\|x\|\|$	A double modulus denotes the norm
$\dot{x}$	Dot symbols represent derivatives
$\mathbf{x}$	Lowercase boldface text represents a vector
$\mathbf{X}$	Capital boldface text represents a Matrix
$\mathbf{X}'$	Accented capital boldface text represent Jacobians
	Processes
$quat(x)$	Function that converts a variable into a quaternion
$bel(x)$	Function that computes the Belief
$x \otimes x$	Represents a quaternion multiplication
$\frac{dx}{dt}$	Liebniz's notation to denote a standard derivative
$\frac{\partial x}{\partial t}$	Liebniz's notation to denote a partial derivative

1 Introduction

1.1 Robotic Localisation and Mapping

Robotics aims to equip machines with the capability of operating autonomously in the physical world to serve various practical purposes. These machines (robots) are designed to resemble human behaviour and action, so that they can substitute for humans in unknown and potentially hazardous environments ranging from planetary exploration to assembly lines [1, 2]. To achieve complete autonomous operation, it is essential that the robot is capable of observing it's surrounding environment, subsequently building a map thereof in order to locate itself within this map. The aforementioned processes are more commonly referred to as *map building* and *localisation* respectively. Ultimately, the physical world presents many unforeseen factors and circumstances. These factors contribute to *uncertainty* and generally emerge due to a robot's lack of critical information. Factors such as environments, sensors, robots, models and computation all lead to an increase in uncertainty and can prohibit accurate map building and subsequently, localisation. *Probabilistic robotics* however, models this uncertainty mathematically in order to provide fundamental probabilistic algorithms that can be used to obtain reasonably accurate and efficient solutions to map building and localisation.

Occupancy grid mapping is map building algorithm that uses probabilistic algorithms. Initially presented in a study by Elfes and Moravec [3], occupancy grid mapping seeks to calculate the probability that a given position in the environment is occupied by an obstacle. Occupancy grid mapping typically uses range sensors, such as sonar or laser range finders to calculate these probabilities and represent them in a *dense* spatial map. An example of an occupancy grid map is depicted in Figure 1. The disadvantages of using occupancy grid mapping however, is that the robot's pose (a robots position and orientation) is assumed to be known and that the environment is assumed to be static, limiting the practical applications thereof. A further overview of alternative map generating techniques is presented in a study by Thrun [4].

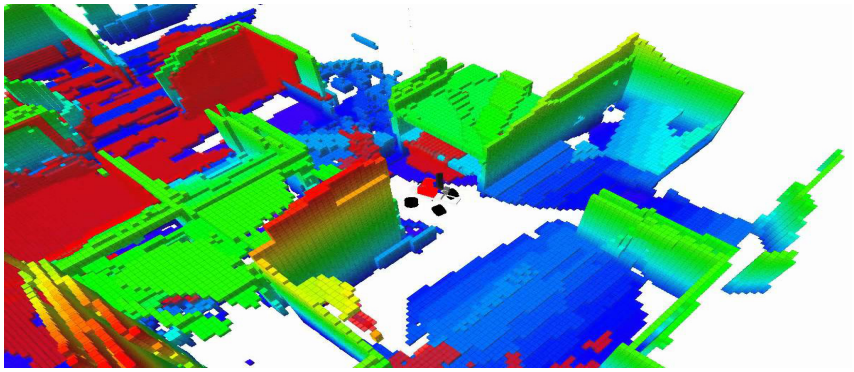


Figure 1: Example of a three-dimensional (3D) occupancy grid map. Adapted from [5].

Simultaneous localisation and mapping (SLAM) however, presents solutions that concurrently realises map building and localisation [6]. SLAM can be described as utilising both the sensor measurements and control inputs of the robot to construct a continuously expanding map of features in the surrounding environment, while concurrently estimating it's location with respect to these features. The sensor measurements provide information about the robot's environment and the control inputs provide the information about how

the robot is moving. A SLAM map typically represents the locations of it's features as a *sparse* set (as opposed to a dense occupancy grid map). An example of a SLAM map is depicted in Figure 2.

The relationship between map building and localisation remains essential to the SLAM problem. If the localisation technique is incorrect, subsequently obtained sensor information will be incorrect, resulting in map estimates that differ from the actual state of the environment. An incorrectly modelled environment will render the sensor information useless, as this information will not correspond with those expected by the constructed map. Ultimately, the resulting localisation approximation will drift over time and eventually become extremely inaccurate. Most modern realisations of SLAM rely on certain probabilistic methods, namely *recursive state estimation* to provide suitable estimates that minimise the uncertainty regarding the mapping and localisation relationship.

The most commonly used recursive state estimation techniques include both *optimal filtering* techniques [7] as well as *sequential importance sampling* (SIS) [8, 9]. The differences regarding the aforementioned methods can be described as follows: an optimal filtering technique only considers a single hypothesis upon modelling, whereas SIS multiple hypothesis to be considered. A further summary regarding the aforementioned probabilistic approaches is provided in a study by Chen [10].

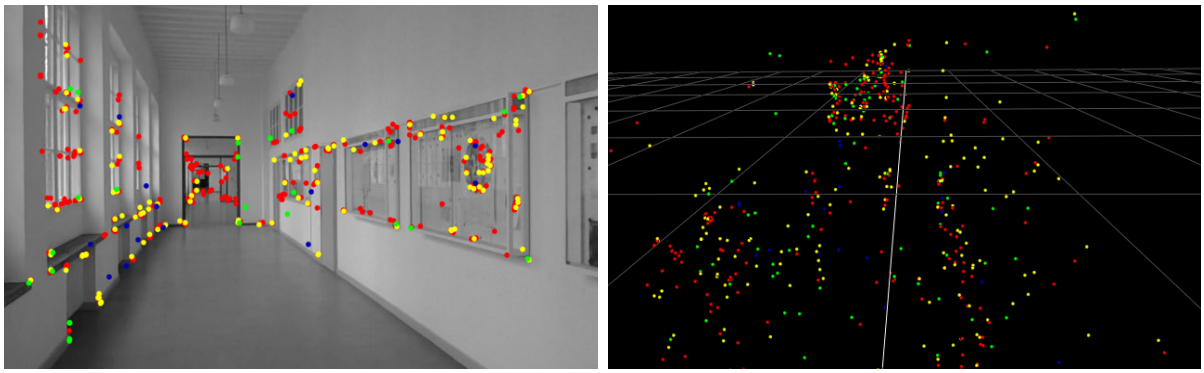


Figure 2: Left: Image frame indicating the feature points. Colours represent the size of a feature (warm colours being the smallest and cool colours being the largest). Right: Reconstructed SLAM three-dimensional (3D) map depicting the feature points with respect to a ground plane (shown as a grid). Adapted from [11].

The choice of SLAM implementation will primarily depend on the type of sensor(s) along with the time constraint imposed upon the application (e.g. online map generation vs. offline map generation). Furthermore parameters such as the resultant map's dimensionality as well as their sparsity representation (e.g. point clouds, occupancy grids or sparse sets) are considered upon a suitable SLAM implementation.

Figure 3 depicts the SLAM procedure and is further explained in Table 1.1. The robot stores an internal representation (or estimate) of the positions of the features, it's pose as well as the uncertainty associated with each of these entities. It should be noted that these uncertainties are not statistically independent of one another. At each frame, a *prediction* regarding the robot's pose, a *measurement* of the observed feature and an *update* of the internal representation is made.

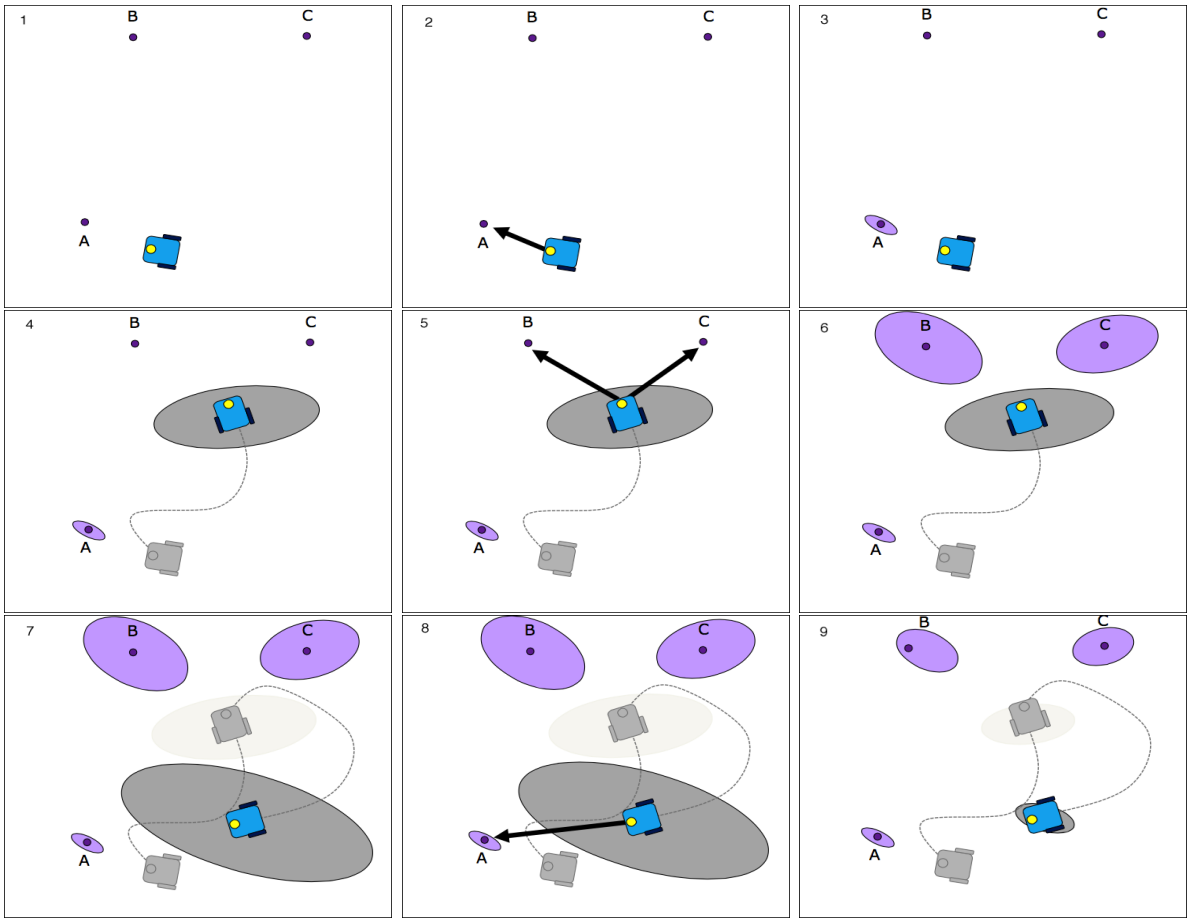


Figure 3: A basic representation of the SLAM procedure. Grey ellipses depict uncertainty regarding the robot's pose and purple ellipses depict uncertainty regarding the feature positions. Adapted from [12].

Table 1.1: SLAM individual steps.

Image	Process
$1-3$	Robot begins exploration, observes feature A and updates the internal representation accordingly.
4	Robot moves, subsequently increasing uncertainty regarding it's pose.
$5-6$	Robot observes features B and C and updates the internal representation. Note that the increase in pose uncertainty yields a greater uncertainty in feature position due to the relationship between the robot's pose and the feature estimates.
$7-8$	Robot moves, again increasing uncertainty regarding it's pose before re-observing old feature A. This is referred to as *loop closure*.
9	Because the feature locations and robot's pose estimates are not statistically independent, The uncertainty regarding the pose as well as that of all feature positions decreases.

A feature can also be referred to as a landmark and the aforementioned terms will from hereon in be used synonymously.

1.2 Problem Description

A vision-based autonomous vehicle operating within an unknown and restricted environment requires a constant update regarding it's *current location*. Many autonomous systems have limited knowledge regarding the surrounding environment but may possess a sensor capable of observing the environment - in this case a camera. It is therefore essential that a solution to this specific localisation problem incorporates the ability to use sensor measurements to build a map on the fly while concurrently locating itself within the map. In a restricted environment, it is likely that a robot will return to a previously observed region, making it essential to incorporate robust *repeatable* localisation where drift from ground truth can be corrected.

Vision-based SLAM implementations as depicted in Figure 3 provide the necessary functionality to achieve repeatable localisation. SLAM algorithms utilising single cameras [13, 14, 15] have provided elegant yet accurate solutions to the SLAM problem, reconstructing accurate SLAM maps and subsequently, localisation. One such algorithm presented by Davison et al. [13], termed MonoSLAM, allows for real-time repeatable localisation of a handheld camera moving within a restricted environment. MonoSLAM, along with the work proceeding it [13, 16, 17, 18], has achieved successful results in retrieving the trajectory of a robot, forming a persistent SLAM map and ultimately maintaining repeatable localisation. Although a map of features is not the desired outcome, it remains essential to solving the localisation problem.

There are however, inherent disadvantages of a MonoSLAM system. Firstly, the utilisation of a single camera prohibits the system from immediately obtaining an accurate depth estimate. A feature is required to be observed from several different viewpoints before an accurate depth estimate can be made. Secondly, the motion model, namely a *constant linear and angular velocity* model, constrains the movement of the system to smooth trajectories - as depicted in Figure 4. If erratic forces or disturbances act upon the system, the pose of the robot is generally lost, and in most cases irrecoverable. Lastly, because there is no sufficient knowledge of the motion of the robot, the practical applications of the system are vastly limited.

Information obtained from additional sensors could potentially improve the disadvantages presented by MonoSLAM. Implementations of single camera SLAM using information from an additional measurement sensor (e.g. a laser range finder), such as that presented in a study by Fu et al. [19], are not only too expensive but would be difficult to integrate with a single camera system. A more elegant solution that uses measurements to improve the motion estimates is required.

To extend the range of practical applications of MonoSLAM, a motion model capable of estimating the pose of a robot due to a variety of movements is essential. The current constant linear and angular velocity model limits the robot to smooth movements 4 and therefore needs to be adjusted or replaced. Using additional information can provide the current velocity motion model with better state estimates, but cannot allow movements that aren't "smooth". Davison et. al [13], confirms that additional information such as angular rates obtained from a *gyroscope* improve the state estimates, but cannot compensate for a change in model dynamics.

A *kinematic estimator* directly measures the derivatives (first and second order) of the position and orientation of the robot as opposed to calculating them from the control inputs and the system's physical model. Inertial measurement units (IMU) can directly measure these derivates and typically comprise of an accelerometer and a gyroscope (and sometimes a magnetometer) that record the accelerations and angular rates respectively.

Incorporating such a sub-system could allow a wider variety of practical applications as the physical model of the target system is irrelevant.

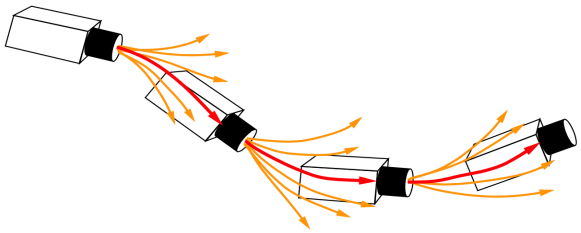


Figure 4: Visualisation of the smooth trajectories of the constant velocity model. Adapted from [13]

1.3 Project Objectives

This project seeks to utilise the aforementioned MonoSLAM system of Davison et al. and improve the localisation thereof by using *additional* sensor information. The improvement(s) of the system should allow the existing system to obtain better localisation and extend the range of applications upon which the system can be applied while maintaining the original performance standard: repeatable localisation at 30 Hz for approximately 100 features. The system should utilise a *single* camera as the measurement sensor and preferably support real-time operation. All processing regarding the SLAM algorithm can be done on a standard PC that communicates with sensors via a serial port. The project budget is R 1500.00.

The primary objectives of this study include:

- **Performing an overview on the current techniques used to realise SLAM**
  In order to understand how SLAM is implemented, an understanding of probability theory and state estimation is required. These concepts - specifically recursive state estimation and the Bayes Filter - need to be researched and analysed before choosing a suitable technique to implement.

- **Analysis of the kinematic estimator as an alternative motion model**
  A kinematic estimator is suggested as a alternative motion model. The kinematic estimator needs to be researched, mathematically derived and simulated. The results from the simulation should correspond to the mathematical derivation. The advantages and disadvantages of the kinematic estimator also need to be investigated.

- **Hardware Design of the System**
  The kinematic estimator requires additional measurements obtained from an IMU. The IMU is required to interface with the PC via a micro-controller. The micro-controller allows precise synchronisation between the images sampled by the camera and the IMU measurements. A functional diagram of the system's hardware components are depicted in Figure 5.

- **Comparison between proposed and original MonoSLAM implementations**
  If time allows, a set of tests are required to be derived and implemented to establish whether the proposed improvement(s) actually adhere to the problem description.

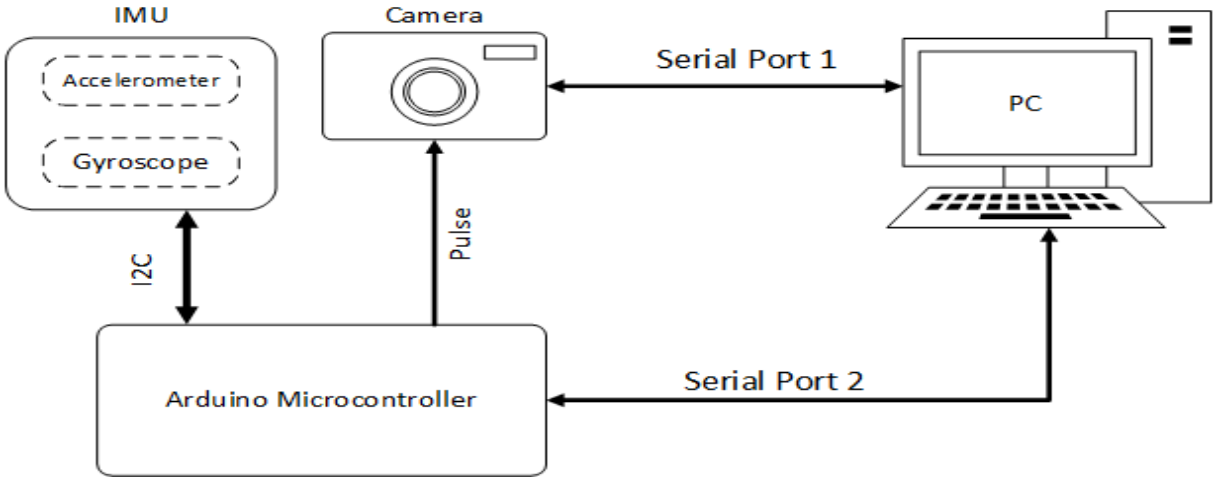


Figure 5: Functional diagram of the interaction between the system's hardware components

1.4 Project Outline

The remainder of this report is presented as follows:

Chapter 2: Probabilistic State Estimation Techniques
This chapter presents all of the research undertaken in order to effectively design and implement a suitable SLAM implementation. This chapter focusses of introducing the reader to the basic theory pertaining to probability theory and recursive state estimation and presenting the relevant mathematical expressions that model them. The chapter further introduces the steps necessary to derive the extended Kalman Filter from the Bayes Filter.

Chapter 3: System Design
This chapter provides a detailed overview of the solution to the MonoSLAM problem. The relevant variable representation and functional components are initially discussed. The chapter then seeks to derive the kinematic estimator necessary for the EKF's prediction step. An overview of the EKF's measurement update is also given.

Chapter 4: Implementation
This chapter sets out to predict the behaviour of the various sub-systems through simulation. The results from the analysis of the kinematic estimator, EKF and IMU are all analysed accordingly.

Chapter 5: Testing and Results
This chapter presents all the relevant tests carried out to determine the proposals validity. The results of these tests are subsequently analysed.

Chapter 6: Conclusions and Recommendations
This chapter provides a summary recommendations by the author that is applicable to future work. The final conclusion is formulated analysing the state of the final system compared to the objectives of the project.

2 Theory: Probabilistic State Estimation Techniques

2.1 Introduction

The following chapter will provide a brief, yet concise introduction to the fundamental algorithms that are necessary to implement SLAM. The fundamental concepts, particularly the various techniques associated with the implementation thereof will be addressed. The goal of this section is to introduce the fundamental concepts as well as the mathematical and probabilistic principles that form the basis of state estimation in the robotics field of study.

Initially, the Bayes Filter; that is the algorithm that forms the basis of all state estimation techniques presented in this report, will be introduced and formally discussed. Thereafter, the Gaussian Filter family - particularly the Kalman Filter as well as it's variants - are to be introduced, discussed and defined in terms of the context of this project. It is worthwhile to note that theory in this chapter resembles material from the book Probabilistic Robotics by Thrun et al. [20] and is adapted therefrom for convenience sake.

2.2 Recursive State Estimation

State variables define the mathematical state of a system's dynamics and describe the impact of the system's future behaviour in the absence of external factors. Although a robot's dynamics can be mathematically modelled, they have quantities that are not directly observable but can be obtained through sensor measurements. Sensors though, obtain limited data regarding certain quantities and most importantly, are affected and often corrupted by *noise*.

State estimation then, seeks to recover state variables using obtained sensor data, control inputs to the system as well as the system model. The value obtained is referred to as a *state estimate*. In the case of SLAM, the state estimates of a robot incorporate the robot's pose as well the position of each landmark in the environment.

Recursive state estimation however, doesn't require the system to keep a complete history of *all* measurements and control inputs, using only the current control inputs and measurements to update the previous state estimate. Probabilistic state estimation algorithms - to be investigated in this section - compute *belief* distributions regarding state variables where the belief reflects a robot's internal knowledge of it's state.

In the case of SLAM, the robot is required to know its location at each time instance t. The belief must thus be calculated at each time step. In order to achieve recursive state estimation, the state estimates of the robot should only incorporate the latest measurements. Additionally, the calculation of the belief should incorporate the previous estimates. Provided that the the system also obeys the Markov assumption - previous data and future data are independent provided that the current state $\mathbf{x}_t$ is known - the *Bayes Filter* provides such recursive state estimation.

2.3 Bayes Filter

The previously mentioned belief of a robot can be represented by a probability distribution that assigns a probability to each state outcome. The belief distribution is a posterior probability over the state variable that is conditioned over the measurements and control data [21]. Mathematically the belief with regard to a state variable $\mathbf{x}_t$ is as shown in the book Probabilistic Robotics by Thrun et al. [21]:

$$bel(\mathbf{x}_t) = p(\mathbf{x}_t \mid \mathbf{z}_{1,..,t},\ \mathbf{u}_{1,..,t}). \tag{2.1}$$

This describes, for a given time t, a joint density of the robot state as well as the landmark locations given all of the previously recorded observations $z_{1:t}$ and control inputs $u_{1:t}$. Table 2.2 presents a pseudo-algorithm of the Bayes Filter algorithm [20]:

Table 2.1: The Bayes Filter Algorithm

Input:	previous belief $bel(\mathbf{x}_{t-1})$, control input(s) $\mathbf{u}_t$, measurement(s) $\mathbf{z}_t$
Output:	current belief $bel(\mathbf{x}_t)$
for all $\mathbf{x}_t$:	
1.	$\overline{bel}(\mathbf{x}_t) = \int p(\mathbf{x}_t \mid \mathbf{u}_t,\ \mathbf{x}_{t-1}) bel(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1}$
2.	$bel(\mathbf{x}_t) = \eta p(\mathbf{z}_t \mid \mathbf{x}_t) \overline{bel}(\mathbf{x}_t)$
3.	end for.

The recursive nature of the algorithm can thus be seen from Table 2.1; whereby the belief $\overline{bel}(\mathbf{x}_t)$ at the current time t is obtained through initially calculating the belief at the previous time-step, $t-1$. The Bayes Filter contains two essential steps: *prediction* (line 1) and *measurement update* (line 2). The prediction step initially processes the control inputs before subsequently predicting the current belief based on the prior belief and the probability that a transition from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$ occurs. Thereafter, the measurement update improves the belief by adding information about the states, observed from new measurements.

The Bayes Filter can be implemented in many different ways. Upon choosing a suitable implementation, a trade off between the following properties needs to be made:

- Computational efficiency
- Accuracy of the approximation
- Ease of Implementation

The mathematical derivation of the Bayes Filter contains many assumptions and further technicalities. The techniques presented in this project require only a basic understanding of the Bayes Filter. A detailed analysis of the Bayes Filter can be obtained from the book Probabilistic Robotics by Thrun et al. [21].

2.4 Gaussian Filters

Amongst the many different implementations of the Bayes Filter are the *Gaussian filter* family. The basic idea behind a Gaussian filter is that beliefs can be represented as a multivariate Gaussian distributions, represented mathematically as follows [21]:

$$p(\mathbf{x}_t) = \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}|}} \exp\left\{ -\frac{1}{2}(\mathbf{x}_t - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_t - \boldsymbol{\mu}) \right\}, \tag{2.2}$$

where the density across the state variable $\mathbf{x}_t$ is characterised through two fundamental parameters: the mean $\boldsymbol{\mu}$ and the covariance $\boldsymbol{\Sigma}$. Such a parameterisation whereby a Gaussian is characterised through it's respective mean and covariance is called the *moments parameterisation* (as the mean and covariance represent the first and second order moments respectively). AThis parameterisation allows a number of recursive filter algorithms to be derived, two of which are examined in this project: the *Kalman Filter* (KF) and it's non-linear counterpart, the *Extended Kalman Filter* (EKF). It is important to realise that both of the aforementioned filters belong to the same sub-class of filters - namely the Kalman Filter Family - and therefore most of the fundamental concepts and functionality between them are identical. Each filter is discussed in further detail in the subsections that follow.

2.4.1 Kalman Filter

Probably the most fundamental of all Gaussian filter algorithms, is the *Kalman Filter*. The Kalman filter can be briefly described as an optimal estimator. It remains a popular technique for filtering and prediction of *linear* systems that contains Gaussian uncertainty. The Kalman filter seeks to describe a belief distribution of a state variable $\mathbf{x}_t$ as described in Equation 2.2. Subsequently, the state vector $\mathbf{x}_t$ is modelled by a single multivariate Gaussian distribution with a mean $\boldsymbol{\mu}_t$ and covariance $\boldsymbol{\Sigma}_t$, at each time instance t (while previous time-steps are denoted as $t-1$, $t-2$, etc.). The general implementation as described above though, is only valid provided that the following three properties hold true - as listed in [21]:

1. The state transition model probability **must** be a *linear* function with additive Gaussian (process) noise. The state transition model probability is shown according to Thrun et al.[21]:

$$g(\mathbf{u}_t, \mathbf{x}_{t-1}) : \mathbf{x}_t = \mathbf{A}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{x}_t + \mathbf{w}_t. \tag{2.3}$$

2. The observation model probability **must** be a *linear* function with additive Gaussian (sensor) noise. The observation model probability is shown according to Thrun et al.[21]:

$$h(\overline{\boldsymbol{\mu}}_t) : \mathbf{z}_t = \mathbf{C}_t\mathbf{x}_t + \mathbf{v}_t, \tag{2.4}$$

   where $\mathbf{w}_t$ and $\mathbf{v}_t$ represent process and sensor noise respectively.

3. The initial belief $bel(\mathbf{x}_0)$ must be normally distributed.

The input to the Kalman Filter is the belief at time $t-1$, represented by $\boldsymbol{\mu}_{t-1}$ and $\boldsymbol{\Sigma}_{t-1}$. The Kalman filter requires a control input $\mathbf{u}_t$ and a measurement $\mathbf{z}_t$ at time t

to update the belief. Like the Bayes Filter, the Kalman filter too is executed in two (sequential) steps: the *prediction step* and the *update step*.

Firstly at time t, the prediction step aims to calculate a predicted belief $\overline{bel}(\mathbf{x}_t)$ represented by $\overline{\boldsymbol{\mu}_t}$ and $\overline{\boldsymbol{\Sigma}_t}$. The predicted belief is obtained by incorporating the control input $\mathbf{u}_t$ and subsequently updating the mean vector and covariance matrix according to the (linear) state transition function. The update step aims to obtain the desired belief $bel(\mathbf{x}_t)$ from the predicted belief $\overline{bel}(\mathbf{x}_t)$ by incorporating the measurements $\mathbf{z}_t$. The Kalman Filter computes a Kalman gain, that intuitively suggests the influence of a measurement in the new state estimate. This gain is then used to update the mean estimate as well as the covariance matrix.

Each of the aforementioned steps are later discussed in more detail - with reference to implementations specific to this paper. Table 2.2 below, presents a pseudo-algorithm of the Kalman Filter [20]:

Table 2.2: The Kalman Filter Algorithm

Input:	previous mean $\boldsymbol{\mu}_{t-1}$ and covariance $\boldsymbol{\Sigma}_{t-1}$, control inputs $\mathbf{u}_t$, measurements $\mathbf{z}_t$
Output:	mean $\boldsymbol{\mu}_t$, covariance $\boldsymbol{\Sigma}_t$
	Prediction step
1.	$\bar{\boldsymbol{\mu}}_t = g(\mathbf{u}_t, \boldsymbol{\mu}_{t-1}) = \mathbf{A}_t \boldsymbol{\mu}_{t-1} + \mathbf{B}_t \boldsymbol{\mu}_t + \mathbf{w}_t$
2.	$\bar{\boldsymbol{\Sigma}}_t = \mathbf{A}_t \boldsymbol{\Sigma}_{t-1} \mathbf{A}_t^T + \mathbf{R}_{w,t}$
	Correction step
3.	$\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_t \mathbf{C}_t^T (\mathbf{C}_t \bar{\boldsymbol{\Sigma}}_t \mathbf{C}_t^T + \mathbf{R}_{v,t})^{-1}$
4.	$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t [\mathbf{z}_t - \mathbf{C}_t \bar{\boldsymbol{\mu}}_t]$
5.	$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{C}_t) \bar{\boldsymbol{\Sigma}}_t$

The Kalman Filter is generally considered an efficient algorithm for sparse sets. The *computational complexity* of a matrix inversion is bounded by an order of $O(d^{2.4})$ [21], where d represents the dimensions of the measurements vectro $\mathbf{z}_t$.

2.4.2 Extended Kalman Filter

Considering that most practical systems of interest yield non-linear behaviour, the Kalman filter in its purest form cannot be successfully implemented upon the vast majority of modern day systems. Non-linear transformations of Gaussian random variables (RV) result in a different RV while any linear transformation of a Gaussian random variable yields another **different** Gaussian variable. This violates an important condition of the Kalman Filter algorithm. This phenomenon can be illustrated through Figure 6:

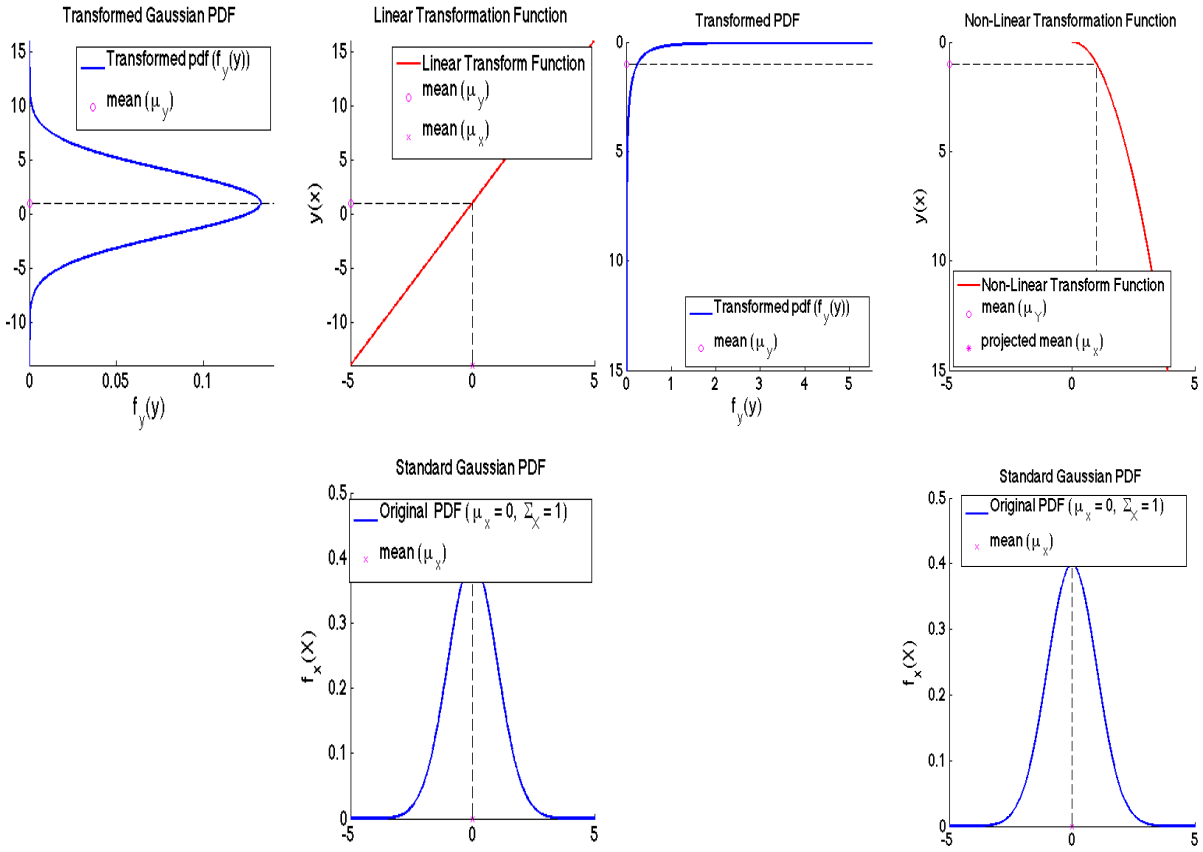


Figure 6: Left: Linear transformation of a Gaussian random variable. Right: Non-linear transformation of a Gaussian random variable.

The *Extended Kalman filter* (EKF), an extension of the general Kalman filter, aims to enable the modelling of non-linear systems through linearisation. As previously mentioned, the state transition function $g(\mathbf{u}_t, \mathbf{x}_{t-1})$, as well as the observation model $h(\mathbf{x}_{t-1})$ of most practical systems are typically both non-linear in nature. Considering the aforementioned statement; it is necessary to determine a method for approximating a non-linear function as a linear function, more commonly referred to as linearisation. The linearisation process of EKF aims to linearise these functions so that the fundamental operations of the Kalman Filter algorithm remain valid.

The linearisation process approximates an arbitrary non-linear function f by a linear function that is *tangent* to f at the mean value of the Gaussian, $\boldsymbol{\mu}$. If the Gaussian is then projected through this new linear approximation, the resultant transformation would yield a random variable that is Gaussian in nature (as in Figure 6). This technique is applied to both the state transition and observation functions. Many methods exist for linearisation of non-linear functions, but the EKF utilises the method of (first order)

Taylor expansion. The Taylor expansion creates a linear approximation of a non linear function, say f, by it's own value as well as that of it's gradient f'. The tangent of f can be depicted by it's partial derivative with respect to the state vector $\mathbf{x}_{t-1}$:

$$f'(\mathbf{x}_{t-1}, \mathbf{u}_t) := \frac{\partial f(\mathbf{x}_{t-1}, \mathbf{u}_t)}{\partial \mathbf{x}_{t-1}}. \tag{2.5}$$

The argument of the function f is chosen as the most likely point at the linearisation instance. For Gaussians, the most likely point is the mean μ_{t-1}. The linear approximation of the function f can then be achieved through the linear extrapolation evaluated at it's most likely point μ_{t-1}:

$$\begin{aligned} f(\mathbf{x}_{t-1}, \mathbf{u}_t) &\approx f(\mathbf{x}_{t-1}, \mathbf{u}_t) + f'(\mathbf{u}_t, \boldsymbol{\mu}_{t-1})(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) \\ &= f(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{F}'_t(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}), \end{aligned} \tag{2.6}$$

where $\mathbf{F}'_t = f'(\mathbf{u}_t, \mathbf{x}_{t-1})$ is the *Jacobian* matrix.

It is important to note that Jacobian matrix is determined at each linearisation instance (each individual time-step) as its parameters differ from one linearisation instance to the next.

It is very important to note that because only a first order Taylor expansion is used to *approximate* the linearisation, severe non-linearities will prohibit acceptable approximations of the Gaussian distribution upon transformations. If the linearisation point is chosen at a point close to the mean, the EKF will yield an acceptable approximation from the linearisation process.

There are other variants of the Kalman Filter that aren't discussed in this project, but have been carefully considered. It is assumed that the first order Taylor approximation provides a suitable approximation of the non-linearities that are expected in the system, namely the uncertainty regarding angle orientation errors. The *Unscented Kalman Filter* is assumed to be a less appropriate choice of filter to the EKF considering the the lack of severe system non-linearities. The objective of the project seeks to provide localisation for a set of approximately 100 landmarks. The *Information Filter* will only provide a better computational complexity than the EKF if the number of landmarks are much larger. This analysis suggests that the EKF provides a suitable yet simple implementation of the Bayes Filter.

Table 2.3 below, systematically and mathematically represents the steps associated with the EKF [20]:

Table 2.3: The Extended Kalman Filter Algorithm

Input:	previous mean $\boldsymbol{\mu}_{t-1}$ and covariance $\boldsymbol{\Sigma}_{t-1}$, control inputs $\mathbf{u}_t$, measurements $\mathbf{z}_t$
Output:	mean $\boldsymbol{\mu}_t$, covariance $\boldsymbol{\Sigma}_t$
	Prediction step
1.	$\bar{\boldsymbol{\mu}}_t = g(\mathbf{u}_t, \bar{\boldsymbol{\mu}}_{t-1})$
2.	$\bar{\boldsymbol{\Sigma}}_{\boldsymbol{t}} = \mathbf{G}'_t \bar{\boldsymbol{\Sigma}}_{\boldsymbol{t-1}} \mathbf{G}'^T_t + \mathbf{R}_{w,t}$
	Correction step
3.	$\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_{\boldsymbol{t}} \mathbf{H}'^T_t (\mathbf{H}'_t \bar{\boldsymbol{\Sigma}}_{\boldsymbol{t}} \mathbf{H}'^T_t + \mathbf{R}_{v,t})^{-1}$
4.	$\boldsymbol{\mu}_{\boldsymbol{t}} = \bar{\boldsymbol{\mu}} + \mathbf{K}_t[\mathbf{z}_t - h(\bar{\boldsymbol{\mu}})]$
5.	$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}'_t) \bar{\boldsymbol{\Sigma}}_{\boldsymbol{t}}$

3 System Design

3.1 Introduction

This chapter sets out to provide an overview of the MonoSLAM algorithm. Initially, a general overview of MonoSLAM will given, stating the problem, previous work done regarding this problem and the limitations imposed on this work. Thereafter, the adequate state representation will be shown along with the required modelling of the measurement sensors. Finally, a complete analysis of the *control update* and an overview of the *measurement update* will be given.

3.2 MonoSLAM

Originally presented in a study by Davison et al. [13], MonoSLAM provides suitable real-time and repeatable localisation using a single camera SLAM implementation. In MonoSLAM, the robot (typically a handheld camera) assumes a constant linear and angular velocity motion model, using the images from the camera to correct the state estimates. MonoSLAM is not the only successful implementation of single camera SLAM. Separate studies presented by Sola et al. [14] and Klein [15] respectively show alternative solutions to the single camera SLAM problem. MonoSLAM and the work proceeding it [13, 16, 17, 18] however, has achieved successful results in retrieving the trajectory of a robot, forming a persistent SLAM map and ultimately maintaining repeatable localisation. Davison et. al [13] also provides the source code of MonoSLAM, a *modular* system, under the GNU version 3 license so that possible improvements, such as those presented in this project, can be made. This modularity and apparent ease of implementation as opposed to the particle filter based implementations presented by the alternative methods suggest MonoSLAM as an appropriate choice.

As previously mentioned, a MonoSLAM system has many disadvantages. The lack of an initial accurate depth estimate, constrained movement and lack of sufficient knowledge regarding the robot's motion severely limit the systems practical applications. This project however, seeks to utilise a kinematic estimator as an alternative to the current velocity and angular velocity motion model. The kinematic estimator will measure the information provided from an inertial measurement unit that subsequently measures the angular rates and acceleration of the robot.

Furthermore, the remaining components of the proposed approach presented in this paper are identical to that of the MonoSLAM algorithm: that is, using an EKF vision based implementation where the update stage of the EKF depends of the measurements of image data from a single camera.

This particular vision based approach aims to use salient image *patches* as long term landmarks as presented in [22, 23]. These aforementioned patches are typically large in size (11×11 pixels) and are obtained through the image detection operator of Shi and Tomasi [24] from raw monochrome (greyscale) image data presented by the camera. The goal remains to repeatedly re-identify these image template patches over time after (potentially severe) camera movements. Invariably, basic 2D template matching algorithms are of little use, considering that any particular movements of the camera (even minimal) can severely alter the shape of a saved template patch. As a result, MonoSLAM provides the assumption that each patch lies on a locally planar surface and that the surface normal is parallel to the vector from the feature to the camera at the instance that it is initialised. Once the depth of this patch has been determined - this is done

through a small particle filter - the patch is stored to be used as a long term landmark. It is important to note that these patches are not replaced but rather stored to provide a template for matching against a newly obtained 2D image at a later stage. Because the patches are never updated and remain in memory, long term localisation is possible.

With regard to the management of the probabilistic map, it remains essential to the SLAM algorithm that decisions regarding the identification and deletion of landmarks be accurate and efficient. MonoSLAM's map-maintenance criterion demands that 12 reliable "good" features be visible within the camera's field of view in order to maintain accurate localisation. A good feature implies that a feature remains visible within the camera's field of view. A new feature is initialised using the image operator of Shi and Tomasi [24] upon a box of pixels (80 × 60 pixels) placed within an image. This box position is chosen at random with the constraints that it shouldn't overlap with any existing features and that according to the camera's linear and angular velocities, features cannot immediately disappear from the camera's field of view. If a visible feature is unsuccessfully matched more than 50% of the time, the landmark is required to be deleted. It must again be stressed that the aforementioned methods regarding vision based MonoSLAM measurements and map-management are described and implemented exactly as they are in [13].

Figure 7 depicts a functional diagram of the proposed system.

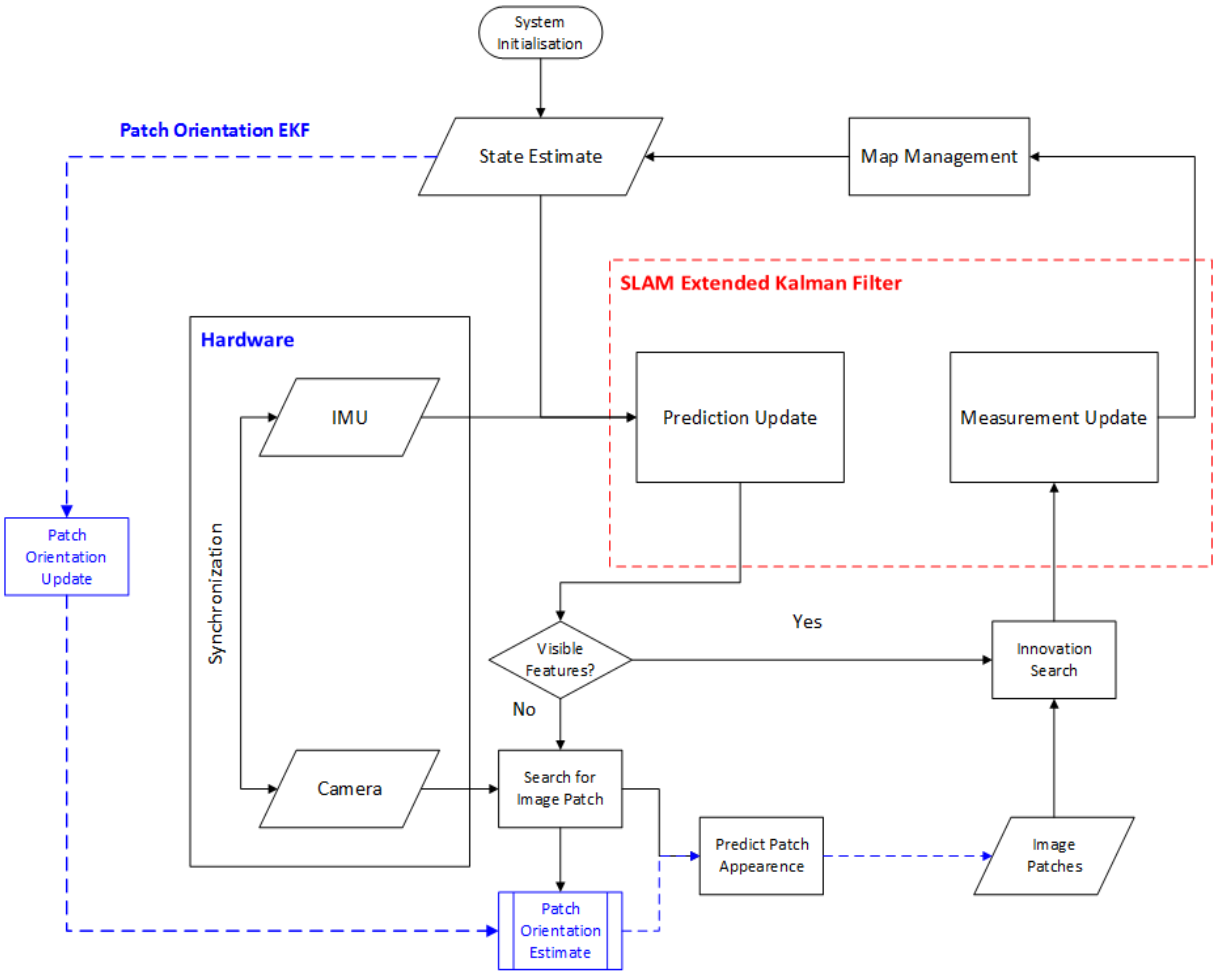


Figure 7: System functional diagram also depicting the relevant sub-systems. (The *prediction update* needs to be changed to control update)

3.3 State Representation

As previously mentioned, state variables represent the mathematical "state" of system. In order to calculate a belief distribution, the system must possess a model to predict future states. This model previously discussed as the state transition probability is commonly referred to as the *motion model*. All relevant states are embedded within the state vector $\mathbf{x}_t$. The state vector is defined at each time-step and comprises of the robot's *actual* pose and the *actual* landmark positions within the map.
Mathematically, the probabilistic map is typically represented as a state *estimate* comprised of a mean state vector $\boldsymbol{\mu}_t$ and a covariance matrix $\boldsymbol{\Sigma}_t$, the subscript t denotes a particular time-instance. The mean state vector, is a single column vector containing the estimates of the robot as well as the landmark positions, and the covariance matrix is a square matrix containing the covariances of each state with respect to every other state. These quantities can be mathematically shown according to the book Probabilistic Robotics by Thrun et al. [21]:

$$\boldsymbol{\mu}_t = \begin{pmatrix} \mathbf{x}_{v,t} \\ \mathbf{y}_{1,t} \\ \mathbf{y}_{2,t} \\ \vdots \\ \mathbf{y}_{n,t} \end{pmatrix}, \quad \boldsymbol{\Sigma}_t = \begin{pmatrix} \Sigma_{x,x} & \Sigma_{x,y_1} & \Sigma_{x,y_2} & \cdots & \Sigma_{x,y_N} \\ \Sigma_{y_1,x} & \Sigma_{y_1,y_1} & \Sigma_{y_1,y_2} & \cdots & \Sigma_{y_1,y_N} \\ \Sigma_{y_2,x} & \Sigma_{y_2,y_1} & \Sigma_{y_2,y_2} & \cdots & \Sigma_{y_2,y_N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Sigma_{y_n,x} & \Sigma_{y_n,y_1} & \Sigma_{y_n,y_2} & \cdots & \Sigma_{y_n,y_N} \end{pmatrix}. \tag{3.1}$$

These quantities then, allow us to approximate the uncertainty regarding the generated feature map as a N-dimensional single multi-variate Gaussian distribution, where N, as stated above, is the total number of state estimates within the state vector and n is the total number of landmarks within the map.

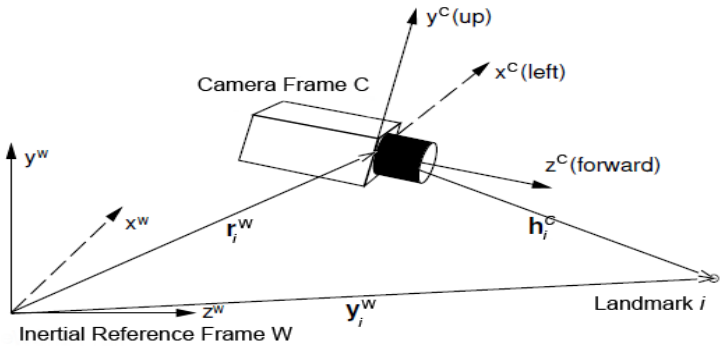


Figure 8: Graphical representation of the appropriate reference frames. Adapted from [22]

3.3.1 Position State Representation

The camera position state $\mathbf{x}_v$ represents all relevant information regarding the camera's position and orientation in a 3D space. The position state vector is comprised of the 3D position vector, $\mathbf{r}^W$, the unit orientation *quaternion* $\mathbf{q}^{WC}$ and the linear velocity vector, $\mathbf{V}^W$ representing the first derivatives of the position vector. The state camera vector -

comprising of 10 individual states - is mathematically described as follows:

$$\mathbf{x}_v = \begin{pmatrix} \mathbf{r}^W \\ \mathbf{q}^{WC} \\ \mathbf{v}^W \end{pmatrix}, \tag{3.2}$$

where $\mathbf{r}^W = (x \ y \ z)^T$ indicates the 3D cartesian position of the camera, $\mathbf{q}^{WC}$ the unit orientation *quaternion* indicating the camera orientation (represented in the body frame C) relative to the inertial reference frame W while $\mathbf{v}^W$ indicates the *linear* velocities of the camera relative to the inertial reference frame W. The reference frames are depicted in Figure 8.

A quaternion, as previously mentioned, represents the camera's orientation. Quaternions are chosen as opposed to Euler angles to prevent the scenario where one degree of freedom is lost due to two axes driven in a parallel configuration. This scenario is more commonly referred to as a *gimbal lock*. It should be noted that all quaternions represented in this paper are unit quaternions. This implies that the square root of the sum of all the squared elements is always equal to 1:

$$q_{0,t}^2 + q_{1,t}^2 + q_{2,t}^2 + q_{3,t}^2 = 1. \tag{3.3}$$

When considering a rotation that changes a robot's orientation, the process of computing the quaternion involves obtaining an angle-axis as well as a magnitude by which this axis is to be rotated. This process is described later in this chapter with reference to the state transition model.

Often, the modelling of dynamic systems require that additional parameters - apart from those describing the position and orientation of the robot - be included in the state vector along with the position state vector. This is illustrated in the description above, with the linear velocity vector, $\mathbf{V}^W$, forms the additional information required for system modelling. This is due to the control inputs, which are of such a nature that intermediary state (namely the linear velocity) is required to describe the control inputs effect on the actual position.

3.3.2 Feature Representation

As previously discussed, the aim is to describe a set of high-quality, well defined landmarks within the map. The map itself is to contain a 3D position of *each* observed landmark as well as a combined uncertainty. The feature estimates $\mathbf{y}_n$ - comprising of N landmarks - is mathematically described through three individual cartesian coordinates - x, y and z respectively:

$$\mathbf{y}_n = (x_n \ \ y_n \ \ z_n)^T, \tag{3.4}$$

where n corresponds to a specific, single landmark.

3.3.3 Measurement Sensor: Camera

The measurement sensor of the the robot in this instance is a single CMOS digital camera. The camera is modelled using the *pinhole camera model* depicted in Figure 9. The pinhole model provides a reasonable approximation of a 3D point in the world and approximates this position according to a 2D. This model incorporates certain assumptions - namely that most digital cameras use lenses rather than a pinhole that can result in distortion. The model can however be adapted to account for the assumption and better approximate the distortion.

The pinhole camera model can be described as a two-dimensional plane, containing the projections of the 3D point $P_i = (x_i \ y_i \ z_i)^T$. The process of representing a 3D coordinate in terms of a 2D coordinate $Q_i = (u_i \ v_i)^T$ is known as *perspective projection*. The 2D plane is commonly referred to as the *pinhole plane* which contains a infinitesimal hole at its centre - the *pinhole*. The camera possess its own 3D coordinate system with coordinate axes X_C, Y_C and Z_C (also referred to as the camera's *optical axis*). The pinhole is situated at the origin of this 3D coordinate system - this is also referred to as the *optical centre*, O. The *image plane* is located at a positive distance f from the optical centre O, parallel to the pinhole plane. This distance f is referred to as the camera's *focal length*.

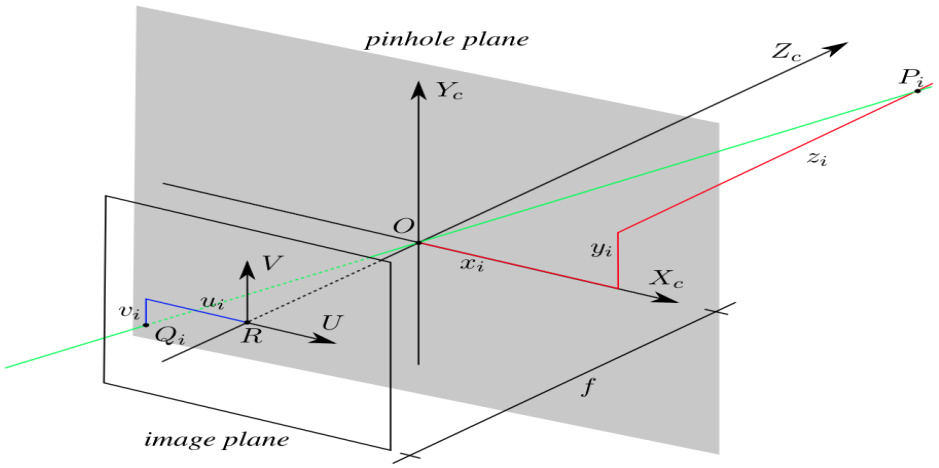


Figure 9: The pinhole camera model. The red lines show the displacement of a point P_i from the optical centre, the blues line show the displacement of a point Q_i from the image centre and the green line shows the projection from the point P_i point Q_i. Adapted from [25]

The undistorted projection of a point P_i in the image plane (u_i,v_i) can be shown as presented in a thesis by Albrecht [25]:

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} u_0 - f\dfrac{x_i}{z_i} \\ v_0 - f\dfrac{y_i}{z_i} \end{pmatrix}, \tag{3.5}$$

where u_0 and v_0 represents the *principal point* located at the image centre.

As previously mentioned, the pinhole camera model is an approximation of the CMOS machine vision camera that is actually being used. To account for the errors that such

an approximation may yield, it is proposed that the *projected* coordinates (u_i, v_i) be warped with a *radial distortion* as is done in [22], in order to obtain a new *distortion* coordinate (u_d, v_d) that will better resemble the one which the camera will provide. This radial distortion is mathematically shown as in a study presented by Swaminathan and Kayer [26]:

$$\begin{aligned} u_d - u_0 &= \frac{u - u_0}{\sqrt{1 + 2K_1 r^2}} \\ v_d - v_0 &= \frac{v - v_0}{\sqrt{1 + 2K_1 r^2}}, \\ r &= \sqrt{(u - u_0)^2 + (v - v_0)^2}, \end{aligned} \tag{3.6}$$

where K_1 and r represent the radial distortion parameters.

3.3.4 Control Input Equivalent

The following concept describes the dynamics to the system as a result of external influences. Upon considering the original MonoSLAM approach, it is evident that there are no *observable* control inputs. The approach presented in this paper though, aims to use measurements obtained through an IMU as an equivalent control input in order to derive a motion model that is potentially more accurate than the implementation of Davison et. al [13].

In most instances of robotics, it is essential to describe the dynamics involving a robot's movement. In the context of this paper, the robot/camera is free to move freely as per the user's control requests. Evidently, these requests exert external dynamics upon the system which are uncertain and stochastic at best.

In the approach presented by Davison et al., a constant velocity model is assumed and at each time-step, unknown linear and angular acceleration zero-mean, Gaussian processes are introduced that cause linear and angular velocity impulses. The model contains very little, if any information on the movement of the camera. It can be assumed that utilising additional information regarding the camera's movement will provide greater accuracy upon state estimation.

The inertial sensors, in the form of an IMU, is ideally mounted onto the camera. This allows the camera to be modelled as a rigid body upon which a kinematic estimation can be applied. The IMU directly measures the total accelerations $\mathbf{f}_t$ as well as the angular rates $\boldsymbol{\omega}_t$ with respect to the cameras rigid body frame C.

The control vector equivalent however, requires that the linear portion of the acceleration be obtained from the IMU measurement. It is known that the total acceleration measured by the IMU's accelerometer is adapted from a study presented by Servent et. al [27]:

$$\mathbf{f}_t = \mathbf{R}^{CW}(\mathbf{a}_t - \mathbf{g}), \tag{3.7}$$

with $\mathbf{a}_t$ is the linear acceleration vector, $\mathbf{g}$ is the gravity vector and $\mathbf{R}^{CW}$ is the rotation matrix that transforms the camera's body coordinate frame C, into the inertial reference frame W.
The rotation matrix is defined according to [28]:

$$\mathbf{R}^{CW} = \begin{pmatrix} q_{0,t}^2 + q_{x,t}^2 - q_{y,t}^2 - q_{z,t}^2 & 2(q_{x,t}q_{y,t} - q_{0,t}q_{z,t}) & 2(q_{x,t}q_{z,t} - q_{0,t}q_{y,t}) \\ 2(q_{x,t}q_{y,t} + q_{0,t}q_{z,t}) & q_{0,t}^2 - q_{x,t}^2 - q_{y,t}^2 - q_{z,t}^2 & 2(q_{y,t}q_{y,z} - q_{0,t}q_{x,t}) \\ 2(q_{x,t}q_{z,t} - q_{0,t}q_{y,t}) & 2(q_{y,t}q_{z,t} + q_{0,t}q_{x,t}) & q_{0,t}^2 - q_{x,t}^2 - q_{y,t}^2 + q_{z,t}^2 \end{pmatrix}. \tag{3.8}$$

The linear acceleration is a processed measurement from the directly obtained accelerometer measurement. The accelerometer measurements firstly subtract the gravity vector $\mathbf{g}$ before being rotated according to the correct orientation frame (from C to W).

Once obtained, these measurements form the *equivalent control vector* $\mathbf{u}_t$ that describes, at each time-step, the dynamics of the system as a result of external forces. The control vector equivalent is adapted as follows:

$$\mathbf{u}_t = \begin{pmatrix} \mathbf{a}_t \\ \boldsymbol{\omega}_t \end{pmatrix} = \begin{pmatrix} \ddot{x}_t & \ddot{y}_t & \ddot{z}_t & \omega_{x,t} & \omega_{y,t} & \omega_{z,t} \end{pmatrix}^T. \tag{3.9}$$

Because the IMU measures the actual movements through sensors, namely an ac-

celerometer and a gyroscope, it is important to note the effects of disturbances and process noise can be directly obtained through these measurements. Moreover, the uncertainty regarding the transition model, namely the process noise, is all incorporated within the noise measurements of the IMU. This noise can be modelled as a zero mean, Gaussian process $\mathbf{w}_t$ with a corresponding covariance matrix $\mathbf{R}_w$. The system noise can be then be mathematically described as follows:

$$\mathbf{w}_t = \begin{pmatrix} \mathbf{n}_{\mathbf{a},t} \\ \mathbf{n}_{\omega,t} \end{pmatrix} = \begin{pmatrix} n_{\ddot{x}_t} & n_{\ddot{y}_t} & n_{\ddot{z}_t} & n_{\omega_{x,t}} & n_{\omega_{y,t}} & n_{\omega_{z,t}} \end{pmatrix}^T, \tag{3.10}$$

with the aforementioned noise model is assumed to be a Gaussian random variable for each of the above elements.

Furthermore, the resultant IMU data to be used is to contain the measurements of the linear accelerations and angular rotations as well as the appropriate additive noise.

3.4 Control Update

This section presents the necessary steps to incorporate the control update of the EKF.

With reference to the probabilistic form of the solution to the SLAM problem, the prediction step requires a description in terms of a belief distribution. The description of the aforementioned state transition model can then, in terms of the probability distribution on the state transitions, take the following form:

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) = \frac{1}{\sqrt{|2\pi \mathbf{R}_w|}} \exp\left\{ \frac{1}{2} \left[\mathbf{x}_t - g(\boldsymbol{\mu}_t,\ \boldsymbol{\mu}_{t-1}) - \mathbf{G}_t^{x_t}(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})\right]^T \right. \\ \left. \mathbf{R}_w^{-1} \left[\mathbf{x}_t - g(\boldsymbol{\mu}_t,\ \boldsymbol{\mu}_{t-1}) - \mathbf{G}_t^{x_t}(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})\right] \right\}, \tag{3.11}$$

where $\mathbf{G}_t^{x_t}$ represents the Jacobian of the state transition motion and $\mathbf{R}_w$ is the process noise.

The state transition model is assumed to take the form of a Markov process, yielding that the current state $\mathbf{x}_t$ is only dependent upon the state immediately preceding it - $\mathbf{x}_{t-1}$ - as well as the input control $\mathbf{u}_t$. Additionally, it is important to note that the uncertainty regarding the state transition model is independent of the uncertainty regarding both the observation model as well as that of the probabilistic map itself.

3.4.1 State Transition Model

As previously discussed, the Extended Kalman FiIter requires a state transition (motion) model in order to obtain the mean estimate $\bar{\boldsymbol{\mu}}_t$, of current state of the system. In short, the motion model describes the transition from the previous state to the following state with regard to the robots kinematic motion as well as the control inputs. In order to derive the state transition model for the system at hand, it is vital that the certain characteristics of the system be understood. Firstly, the robot system - from here on in to be referred to as the **camera** - is comprised of a monocular camera and an attached IMU package. Secondly, the camera is to be considered as a six degree of freedom (DOF) rigid body. Briefly the six DOF describe the camera's three *translational* and three *rotational* degrees of freedom.

We therefore set out to define a kinematic estimation based motion model - using Newton's laws of motion - to describe the camera's movement through the environment as a result of initially unknown, external inputs to the system. Lastly, it should be stressed that embedded within the motion model should be the impacts of uncertainty through both internal and external factors.

It must be stressed that a derivation for the suitable motion model will obtained before a first order Taylor approximation is obtained as required by the EKF. Recalling, the states and control inputs:

$$\begin{aligned} \mathbf{x}_t &= \begin{pmatrix} \mathbf{r}_t^W & \mathbf{q}_t^{WC} & \mathbf{v}_t^W \end{pmatrix}^T \\ \mathbf{u}_t &= \begin{pmatrix} \mathbf{a}_t^C & \boldsymbol{\omega}_t^C \end{pmatrix}^T \end{aligned} . \tag{3.12}$$

Considering that the EKF is a recursive, numerical evaluation, it is necessary to convert the *continuous* linear differential equations that describe the state transition model into a discrete counterpart. Various methods of discretisation exist, though this specific implementation makes use of the forward difference (Eulers) method as the sampling period is assumed to be small enough. This method *approximates* the derivative for a state for a sampling period ΔT as follows:

$$\begin{aligned} \dot{\mathbf{x}}_{t-1} &= \lim_{\Delta T \to 0} \frac{\mathbf{x}_t - \mathbf{x}_{t-1}}{\Delta T} \\ \mathbf{x}_t &\approx \dot{\mathbf{x}}_{t-1} \Delta T + \mathbf{x}_{t-1} \end{aligned} . \tag{3.13}$$

Newton's second law of motion, describing the relationship between a body's mass and it's acceleration, is used to derive the linear motion mode after which the aforementioned method of discretisation is applied to obtain the discrete motion model. A full derivation of the linear state transition model is shown in Appendix C.

The state transition function of the kinematic estimator $g(\mathbf{u}_t, \mathbf{x}_{t-1})$ is defined at the current time t, is dependent on both the current control input equivalents $\mathbf{u}_t$ as well as the actual states $\mathbf{x}_{t-1}$:

$$\begin{aligned} \mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1}) = \begin{pmatrix} \mathbf{r}_t^W \\ \mathbf{q}_t^{WC} \\ \mathbf{v}_t^W \end{pmatrix} &= \begin{pmatrix} \mathbf{r}_{t-1}^W + \mathbf{v}_t^W \Delta T \\ \mathbf{q}_{t-1}^{WC} \otimes \text{quat}\big(\boldsymbol{\omega}_t^C \Delta T\big) \\ \mathbf{v}_t^W + \mathbf{a}_t^C \Delta T \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{r}_{t-1}^W + \dot{\mathbf{r}}^W \Delta T \\ \mathbf{q}_{t-1}^{WC} \otimes \text{quat}\big(\boldsymbol{\omega}_t^C \Delta T\big) \\ \dot{\mathbf{r}}_t^W + \mathbf{R}_t^{CW} \big(\ddot{\mathbf{r}}_t^C \Delta T\big) \end{pmatrix} \end{aligned} , \tag{3.14}$$

where ΔT is defined as the sample period between the previous time $t-1$ and t and $\text{quat}\big(\boldsymbol{\omega}_t^C \Delta T\big)$ denotes the process of obtaining a quartering of the rotation $\dot{\boldsymbol{\omega}}_t^C \Delta T$. Information regarding the change in linear acceleration is directly obtained from the control input equivalent data gathered by the IMU and can be numerically integrated it obtain linear velocity and ultimately, translational position.

All of the equations in the remainder of this subsection are adapted from the MonoSLAM documentation on models provided by Davison [28]

In order to compute the aforementioned quaternion, the rate at which the camera's rotational degrees of freedom are changing is required. The control input equivalent measurements from the gyroscope however, measure exactly this quantity - the angular rate. The angular rate is subsequently numerically integrated in order to obtain the angular position $\boldsymbol{\theta}_t$, before the quaternion is taken thereof. As previously mentioned, an angle-axis as well as a magnitude by which this axis is to be rotated is required to compute a quaternion.

The angle-axis $\boldsymbol{\gamma}$ is defined as follows:

$$\boldsymbol{\gamma} = \Big(\boldsymbol{\theta}, \|\boldsymbol{\theta}\|\Big) = \left(\begin{pmatrix} \theta_x \\ \theta_y \\ \theta_z \end{pmatrix}, \big\|\boldsymbol{\omega}_t^C \Delta T\big\| \right) = \left(\begin{pmatrix} \frac{\omega_{t,X}^C \Delta T}{\|\boldsymbol{\omega}_t^C \Delta T\|} \\ \frac{\omega_{t,Y}^C \Delta T}{\|\boldsymbol{\omega}_t^C \Delta T\|} \\ \frac{\omega_{t,Z}^C \Delta T}{\|\boldsymbol{\omega}_t^C \Delta T\|} \end{pmatrix}, \big\|\boldsymbol{\omega}_t^C \Delta T\big\| \right), \tag{3.15}$$

where $\omega_{t,\beta}$, $\beta \in \{X, Y, Z\}$ denotes the angular velocity about each respective coordinate axis. The result in Equation 3.15 is then represented as a unit quaternion denoting the same rotation:

$$\mathbf{q} = \begin{pmatrix} \cos\frac{\alpha}{2} & \frac{\theta_x}{\|\boldsymbol{\theta}\|}\sin\frac{\alpha}{2} & \frac{\theta_y}{\|\boldsymbol{\theta}\|}\sin\frac{\alpha}{2} & \frac{\theta_z}{\|\boldsymbol{\theta}\|}\sin\frac{\alpha}{2} \end{pmatrix}^T. \tag{3.16}$$

A quaternion multiplication between the orientation quaternion obtained at the previous time step and the angle-axis rotation quaternion in Equation 3.16 results in the camera's final orientation state at the the current time t.

3.4.2 Mean Estimate

The control update uses the previously defined state transition function to obtain a state estimate $\bar{\boldsymbol{\mu}}_t$ at the current time instance. This procedure is shown in Step 1 of Table 2.3. Recalling, the state transition function is dependent on both the current equivalent control inputs as well as the actual states to obtain a suitable state estimate:

$$\mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1}) = \begin{pmatrix} \mathbf{r}_{t-1}^W + \dot{\mathbf{r}}^W \Delta T \\ \mathbf{q}_{t-1}^{WC} \otimes \mathrm{quat}\big(\boldsymbol{\omega}_t^C \Delta T\big) \\ \dot{\mathbf{r}}_t^W + \mathbf{R}_t^{CW}\big(\ddot{\mathbf{r}}_t^C \Delta T\big) \end{pmatrix}. \tag{3.17}$$

It is worthwhile to note that the equivalent control inputs (which are the measurements from the IMU) incorporate the additive noise. This concept is shown in Equation 3.10.

3.4.3 Covariance Update

Before the control update can be concluded, the covariance matrix $\bar{\boldsymbol{\Sigma}}_{\boldsymbol{t}}$, corresponding to the previously determined mean vector $\bar{\boldsymbol{\mu}}_{\boldsymbol{t}}$ is required to be updated as a result of the linearisation process undertaken by the EKF. This procedure is denoted in Step 2 of Table 2.3. It can be noticed that the previously described Jacobian matrix of $\mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1})$ is thus required to realise this procedure. The Jacobian matrix of $\mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1})$, $\mathbf{G}_t^{x_t}$, can be mathematically defined as follows:

$$\mathbf{G}_t^{x_t} = \frac{\partial \mathbf{g}(\mathbf{u}_t, \mathbf{x}_{t-1})}{\partial \mathbf{x}_{t-1}} = \boldsymbol{\mu}_{t-1} = \begin{pmatrix} \frac{\partial \mathbf{r}_t^W}{\partial \mathbf{r}_{t-1}^W} & \mathbf{0} & \frac{\partial \mathbf{v}_t^W}{\partial \mathbf{r}_{t-1}^W} \\ \mathbf{0} & \frac{\partial \mathbf{q}_t^{WC}}{\partial \mathbf{q}_{t-1}^{WC}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{v}_t^W}{\partial \mathbf{v}_{t-1}^W} \end{pmatrix}, \tag{3.18}$$

with the non-zero elements of the Jacobian further trivially defined as follows (and according to the model defined in Equation 3.17),

$$\frac{\partial \mathbf{r}_t^W}{\partial \mathbf{r}_{t-1}^W} = \frac{\partial \mathbf{v}_t^W}{\partial \mathbf{v}_{t-1}^W} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \mathbf{I}, \tag{3.19}$$

and

$$\frac{\partial \mathbf{r}_t^W}{\partial \mathbf{v}_{t-1}^W} = \Delta T \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \Delta T \ \mathbf{I}, \tag{3.20}$$

The specially defined Jacobian with a partial derivative that is taken with respect to a quaternion however, requires a more intricate solution. The process of defining a new quaternion from the measured angular rate $\boldsymbol{\omega}_t^C$ - as denoted in Equation 3.17 can be more formally defined as follows:

$$\mathbf{q}_t^C = \text{quat}(\boldsymbol{\omega}_t^C \Delta T). \tag{3.21}$$

Hereafter, the final non zero element of the Jacobian can be defined in terms of the aforementioned new quaternion:

$$\frac{\partial \mathbf{q}_t^{WC}}{\partial \mathbf{q}_{t-1}^{WC}} = \begin{pmatrix} q_{1,t}^C & -q_{2,t}^C & -q_{3,t}^C & -q_{4,t}^C \\ q_{2,t}^C & q_{1,t}^C & -q_{4,t}^C & q_{3,t}^C \\ q_{3,t}^C & q_{4,t}^C & q_{1,t}^C & -q_{2,t}^C \\ q_{4,t}^C & -q_{3,t}^C & q_{2,t}^C & q_{1,t}^C \end{pmatrix}. \tag{3.22}$$

In order to complete the covariance update, the process noise covariance is still required to be updated. Upon revising, the process noise is captured by the control inputs and modelled as a zero-mean Gaussian process. The noise covariance $\mathbf{R}_w$ then, can be formally defined as follows:

$$\mathbf{R}_w = \begin{pmatrix} n_{\ddot{x},t} & 0 & 0 & 0 & 0 & 0 \\ 0 & n_{\ddot{y},t} & 0 & 0 & 0 & 0 \\ 0 & 0 & n_{\ddot{z},t} & 0 & 0 & 0 \\ 0 & 0 & 0 & n_{\omega_x,t} & 0 & 0 \\ 0 & 0 & 0 & 0 & n_{\omega y,t} & 0 \\ 0 & 0 & 0 & 0 & 0 & n_{\omega z,t} \end{pmatrix}^2, \tag{3.23}$$

with the perturbation levels $n_{w,t}$ as defined in equation 3.10.

3.5 Measurement Update

With reference again to the probabilistic form of the solution to the SLAM problem, the measurement step too, requires a description in terms of a belief distribution. The observation model however, models the uncertainty regarding a measurement taken at any given time instance $\mathbf{z}_t$, given that the locations of both the robot as well as the location of the landmarks are known. This uncertainty can be described in the following form:

$$p(\mathbf{z}_t \mid \mathbf{x}_t) = \frac{1}{\sqrt{|2\pi\mathbf{R}_v|}} \exp\left\{\frac{1}{2}\left[\mathbf{z}_t - \mathbf{h}(\bar{\boldsymbol{\mu}}_t) - \mathbf{H}_t^{x_t}(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t)\right]^T \mathbf{R}_v^{-1}\left[\mathbf{z}_t - \mathbf{h}(\bar{\boldsymbol{\mu}}_t) - \mathbf{H}_t^{x_t}(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t)\right]\right\}, \tag{3.24}$$

where $\mathbf{H}_t$ represents the Jacobian of the observation model and $\mathbf{R}_v$ is the sensor noise.

It can be (reasonably) assumed that the uncertainty regarding the measurements are conditionally independent given the uncertainty regarding the robot and landmark locations if indeed they are completely defined. Also, the correction step seeks to obtain the difference between the actual measurements $\mathbf{z}_k$ and the predicted measurements. These predicted measurements are to be obtained through an observation model that we from hereon in refer to as the measurement function, denoted as $\mathbf{h}(\bar{\boldsymbol{\mu}})$.

3.5.1 Measurement Model

The correction step of the Extended Kalman filter aims to ultimately correct the previously estimated robot pose and landmark position through exterior sensor measurements. With regard to the implementation proposed in this paper, these measurements are obtained through the use of a camera. The measurement process generally involves a measurement estimate that incorporates uncertainty.

With reference to figure 8, a feature's cartesian position can be described through a cartesian vector $\mathbf{h}_i^W(\bar{\boldsymbol{\mu}})$, where the feature's cartesian **point** is shown in relation to the camera's centre:

$$\mathbf{h}_i^W(\bar{\boldsymbol{\mu}}) = \mathbf{R}^{CW}\left(\mathbf{y}_i^W - \mathbf{r}^W\right) = \left(\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} - \mathbf{r}^W\right) \tag{3.25}$$

the subscript i corresponds a directional vector $\mathbf{h}^C$ from it's cartesian position $\mathbf{r}^W$ to the cartesian position of a given landmark $\mathbf{y}^W$.

A camera however, cannot directly measure a cartesian vector. Instead, a camera measurement (based on the model presented) obtains a vector $\mathbf{h}_i$ that is a function of $\mathbf{h}_i^W$. This vector describes a given feature's horizontal and vertical image positions (u, v). For an undistorted image, the vector $\mathbf{h}_i$, more commonly referred to as the measurement function is defined according to Equation 3.5:

$$\mathbf{h}_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} u_0 - fk_u \dfrac{h^R_{i,x}}{h^R_{i,z}} \\ v_0 - fk_v \dfrac{h^R_{i,y}}{h^R_{i,z}} \end{pmatrix} \tag{3.26}$$

where u_0 and v_0 represent the principal point and fk_u and fk_v are the camera calibration parameters described in Chapter 3.3.3.

It is evident from the model presented in equation 3.26 cannot be directly inverted to obtain a feature's position. The projection of a feature onto the camera's image plane removes any information required to directly obtain the depth of the feature.

3.5.2 Feature Matching

The following section discusses the measurement of a feature *fully* initialised within the SLAM map. The measurement process seeks to initially estimate the cartesian position of a given feature $\mathbf{y}_i$ within the SLAM map. Thereafter, the feature can be compared via a matching sequence. Generally, feature matching is conducted using a normalised cross-correlation search, where a 2D template of the 3D feature is scanned is across the entire image (at each pixel location) until a peak is obtained. MonoSLAM however, seeks to utilise an *active* approach for matching, minimising the the search field and improving efficiency.

The EKF inherently contains information that may be utilised in order to prohibit a full cross-correlation search. The measurement function $\mathbf{h}(\bar{\mu})$ for instance, provides an estimate for a given features location, namely $\mathbf{u}_d = (u_d, v_d)$. Knowledge of this location therefore allows an active search region to be described within the vicinity of this location. The location estimate of the feature is not the only information regarding the feature that is available as a result of the EKF. Additionally, the uncertainty regarding a given feature's location is stored within the state vector covariance matrix $\mathbf{P}_{nN}$. This information can be used to determine the size of the active search region surrounding the location estimate; where the size of the search region is directly proportional to the uncertainty of it's location. If the feature cannot be matched within the aforementioned search region, it cannot contribute to the correction of the robot's pose estimate and is therefore deleted form the SLAM map. The aforementioned process of defining the active search region can be mathematically defined through the *innovation covariance matrix* $\mathbf{S}_i$:

$$\mathbf{S}_i = \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{x}_v} \Sigma_{\mathbf{x}_v,\mathbf{x}_v} \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{x}_v}^T + \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{x}_v} \Sigma_{\mathbf{x}_v y_i} \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{y}_i}^T + \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{y}_i} \Sigma_{y_i \mathbf{x}_v} \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{x}_v}^T + \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{y}_i} \Sigma_{y_i y_i} \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{y}_i}^T + \mathbf{R}_v \tag{3.27}$$

The symmetric 2×2 matrix $\mathbf{S}_i$ represents a 2D Gaussian PDF around the estimated image coordinate. The innovation covariance matrix can then be used to determine an active region the a given feature should lie within. Typically, the active search region is defined to confine within 3 standard deviations (3σ) of the mean.

Furthermore, the innovation matrix provides a measure of the amount of content expected within an eventual actual measurement $\mathbf{z}_i$. In the event that many potential measurements are available, features containing a higher $\mathbf{S}_i$ present the EKF with more information regarding the camera's position. Candidates for feature estimates are thus chosen according to those that present the most information regarding the position estimate. Feature searches per sampling instance are generally limited (usually about 12

features) due to computational constrains.
Finally, as described in [13], an active search will always reduce the area of the template matching search region at the potential *additional* cost of calculating the reduced search region.

3.5.3 Feature Initialisation

The inherent disadvantage of a monocular camera, as previously mentioned, is the inability to immediately provide an estimate for the depth of a feature. As a result, a given feature is required to be observed at various viewpoints before its depth can be approximated through a multiple view triangulation. Instead, Davison et al. presents an alternative approach whereby a feature is initialised to lie along an infinite 3D line. This line, originating from the position at which the camera is estimated, extends indefinitely in the direction of the feature. The depth of the feature lies somewhere along the aforementioned line. This depth can be modelled as a uniformly distributed set of discrete depth hypothesis. Briefly, the feature's depth can be interpreted as a 1D probability density, represented only by particle distribution instead. The feature is can then *partially* initialised in the SLAM map as follows:

$$\mathbf{y}_{pi} = \begin{pmatrix} \mathbf{r}_i^W \\ \hat{\mathbf{h}}_i^W \end{pmatrix} \tag{3.28}$$

where $\mathbf{r}_i^W$ represents the origin of the line and $\hat{\mathbf{h}}_i^W$ is a unit vector representing its direction. The uncertainty describing the aforementioned entities are Gaussian in nature.

After a feature has been partially initialised, it can be assumed that the feature is re-observed and that each additional observation improves the depth estimate. The particle filter based depth estimation process itself is to a large extent complex, and is explained in more detain in [13]. Intuitively, the depth estimation process can be explained as follows: each particle in the particle set is projected into the image and subsequently matched across each observation. The resulting observations transform the initially uniformly distributed depth probability into one that better resembles a Gaussian density. Once the depth covariance is below a certain threshold, the depth is approximated with a Guassian probability density. Thereafter a feature becomes *fully* initialised, assigned with a standard 3D Gaussian representation.

3.5.4 Map Management

Map management forms an integral role in the realisation of the MonoSLAM algorithm. A real-time algorithm, as proposed in this paper, relies on efficient and accurate decisions regarding features within the SLAM map. As a result, a strict protocol is followed in [13] in order to realise a successful real-time algorithm.

4 Implementation

4.1 Introduction

This chapter sets out to test all the relevant sub-systems incorporated within this project. Initially, the necessary configuration and procedures are given in order to set up the system. Thereafter, the hardware components are individually tested and analysed to ensure that the expected performance is provided and whether redesign is necessary. Lastly, the core of this project, the kinematic estimator, is simulated and analysed as a possible alternative to the current velocity and angular velocity model.

4.2 Hardware Configuration

An overview of the system is depicted in Figure 5. It is evident that the system contains three hardware sub-systems:

- IMU.
- CMOS Camera.
- Arduino micro-controller.

Each of of the sub-systems requires a unique setup configuration in order to represent a *functional* unit of the whole system.

4.2.1 Inertial Measurement Unit

It can be recalled that inertial measurements are required in order to realise the kinematic estimator described in section 3.4.2. A 6DOF IMU board is thus required to realise these measurements. A 6DOF IMU requires measurements from an accelerometer and a gyroscope. Important properties to be considered upon choosing suitable IMU are as follows:

- Communication with master device (in this case a micro-controller).
- Suitable range that the sensors are capable of measuring.
- Noise on sensors.
- Communication speed.
- Low pass filter to prevent aliasing.
- Sample code.

The 6DOF IMU board to be used in this implementation is the SparkFun 6 DOF IMU as depicted in Figure 10. This board is comprised of a ITG3200 MEMS 3-axis gyroscope and a ADXL345 3-axis linear accelerometer. This component is chosen as it contains both the necessary sensors on the same chip while supporting I2C communication, low pass filtering, suitable range, sample code and a maximum sample rate of 1 kHz. Alternative products generally contain an additional 3 DOF and greater ranges that are overlooked considering that these products are generally twice the price of the chosen component.

The IMU is configured with the following properties:

1. 3.3 Volt input.
2. I2C interface (this device serves as the *slave*).
3. $\pm 2000°$ per second range for the gyroscope.
4. $\pm 4g$ range and 10-bit resolution for the accelerometer.
5. 100 Hz sample rate with 50 Hz Low Pass/Anti Aliasing Filter.

The accelerometer and gyroscope are configured in such a way that the sampling frequency is at least twice the bandwidth of the data (Nyquist frequency) to prevent Aliasing.

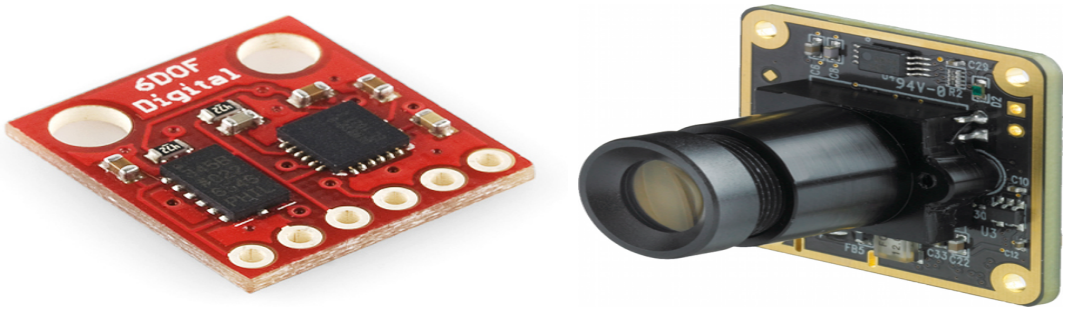


Figure 10:
Left: SparkFun 6DOF IMU. Photo by SparkFun - http://creativecommons.org/licenses/by-nc-sa/3.0/. Right: The Imaging Source 22BUC03-ML CMOS board camera. Adapted from [29]

4.2.2 CMOS Machine Vision Camera

The measurement sensor is required to be a single camera. The problem description only requires the camera to have a 30 Hz frame rate. The camera chosen for this project is a CMOS camera by The Imaging Source as depicted in Figure 10. The DFM 22BUC03-ML model is a machine vision board contains 4 GPIO pins that allow the shutter to be triggered. The camera was used for a previous project and incorporated in this project due to cost constrains.

The camera is configured with the following properties:

- Trigger mode, receiving a pulse to sample a frame (typically 30 Hz).
- Monochrome image capture (for efficiency).

The *intrinsic* parameters of the camera (e.g. principal point, distortion coefficients etc.) need to be determined through camera calibration made available by OpenCV [30]. The results of this procedure - depicted in Figure **??** - are shown as follows:

The trigger pulse is set by the micro-controller, this will be discussed next.

4.2.3 Micro-controller

A micro-controller is required for two reasons. Firstly, the values received from the IMU are analog and need to be converted to digital. Secondly, the camera is configured to receive a pulse to capture a frame. This pulse needs to be sent by the Arduino every sampling instance and simultaneously sample the IMU data so that the image data and the IMU data remain *synchronised*.

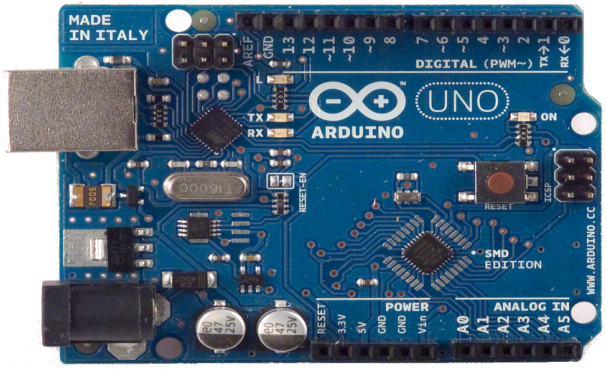


Figure 11: Arduino UNO SMD by Arduino - http://creativecommons.org/licenses/by-sa/3.0/.

This project doesn't require a large micro-controller. The specifications for the micro-controller are shown as follows:

- Two A2D ports for I2C communication with the IMU.
- 3.3 Volt supply voltage for the IMU.
- One digital output pin to trigger a pulse to the camera.
- Support community.

The Arduino micro-controllers provide a suitable, low cost solution. The Arduino Uno SMD Rev3 in particular is a scaled down version of the original Uno, providing the necessary functionality along with a helpful developer community with many forums and available code segments for the IMU used in this project.

4.3 Design Choices

4.3.1 Serial Communication

Table 4.1: Single-Byte I2C Write Cycle

MR	ST	ADR + W		R_ADR		DATA		SP
SL			ACK		ACK		ACK	

Table 4.2: Single-Byte I2C Read Cycle

MR	ST	ADR + W		R_ADR		S	R_ADR			NACK	SP
SL			ACK		ACK			ACK	DATA		

4.3.2 Hardware Integration

4.4 Software Configuration

5 Analysis: Testing & Results

A Summary of Work done

B Achieved ECSA Exit Level Outcomes

C Theoretical Concepts

C.1 State Space Model

As previously discussed, the EKF requires a state transition model in order to estimate the current state of the system. In short, the motion model describes the transition from the previous state to the following state with regard to the robot's kinematic motion as well as the control inputs. The *ideal* motion model in this particular instance can be described through a **linear** differential equation of the following form:

$$\dot{\mathbf{x}}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t + \mathbf{w}_t, \tag{C.1}$$

where the state matrix $\mathbf{A}$, describes the manner in which state evolves from the previous time-step to the current time-step without the influence of noise and controls, the input matrix $\mathbf{B}$, describes how the control vector $\mathbf{u}_t$ evolves from the previous time-step to the current time-step and $\mathbf{w}_t$ is a **zero-mean** Gaussian process representing the process noise with a covariance matrix $\mathbf{R}_w$.

Considering that the Extended Kalman Filter is a recursive, numerical evaluation, it is necessary to convert the previously defined continuous model into its discrete counterpart. Various methods of discretisation exist, though this specific implementation makes use of the forward difference/Eulers method. This method *approximates* the derivative for a state for a sampling period ΔT as follows:

$$\begin{aligned} \dot{\mathbf{x}}_k &= \lim_{\Delta T \to 0} \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\Delta T} \\ \Delta T \dot{\mathbf{x}}_k &\approx \mathbf{x}_{k+1} - \mathbf{x}_k, \end{aligned} \tag{C.2}$$

The state estimate of the discrete counterpart at the following sampling instance, namely $k+1$, is then presented as follows (given a small enough sampling instance ΔT):

$$\mathbf{x}_{k+1} = \big(\mathbf{I} + \mathbf{A}\Delta T\big)\mathbf{x}_k + \mathbf{B}\mathbf{u}_k\Delta T + \mathbf{w}_k\Delta T, \tag{C.3}$$

where $\big(\mathbf{I} + \mathbf{A}\Delta T\big) = \mathbf{A}_d$ is the discrete state matrix, $\mathbf{B}\Delta T = \mathbf{B}_d$ is the discrete input matrix and $\mathbf{w}_k\Delta T = \mathbf{w}_{d,k}$ is the discrete input process noise.

Ultimately, the form of the final difference equation describing the system at each individual sampling instance is given as follows:

$$\mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k + \mathbf{w}_{d,k}, \tag{C.4}$$

C.2 State Transition: Linear Model

In order to derive the motion model for the system at hand, it is vital that the certain characteristics of the system be understood. Firstly, the robot system is comprised of a monocular camera and an attached IMU package. Secondly, the camera is to be considered as a 6 DOF rigid body. Briefly the six DOF describe the camera's three *translational* and three *rotational* degrees of freedom.
We therefore set out to define a kinematic motion model - using Newton's laws of motion - to describe the cameras movement through the environment as a result of initially

unknown, external inputs to the system. Lastly, it should be stressed that embedded within the motion model, should be the impacts of uncertainty through both internal and external factors. It must also be stressed that initially, a stochastic, linear discrete-time model is adopted to approximate the motion model. Using the kinematic equations of linear and angular motion, it is aimed to ultimately and complete the previously defined state space model. We begin by describing all relevant states and control inputs:

$$\begin{aligned}\mathbf{x}[k] &= \begin{bmatrix} x_k & y_k & z_k & \dot{x}_k & \dot{y}_k & \dot{z}_k & q_{0,k} & q_{1,k} & q_{2,k} & q_{3,k} \end{bmatrix}^T \\ \mathbf{u}[k] &= \begin{bmatrix} \ddot{x}_k & \ddot{y}_k & \ddot{z}_k & \dot{q}_{0,k} & \dot{q}_{1,k} & \dot{q}_{2,k} & \dot{q}_{3,k} \end{bmatrix}^T \end{aligned} \tag{C.5}$$

and extend the discrete-time difference equation describing the system to incorporate the motion model,

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k + \mathbf{w}_{d,k}, \\ \mathbf{A}_d &= \begin{bmatrix} 1 & 0 & 0 & \Delta T & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta T & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta T & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = (\mathbf{I} + \mathbf{A}\Delta T), \\ \mathbf{B}_d &= \begin{bmatrix} \Delta T & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \Delta T & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \Delta T & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Delta T & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \Delta T & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \Delta T & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \Delta T \end{bmatrix} = \mathbf{B}\Delta T, \end{aligned} \tag{C.6}$$

$$\mathbf{w}_{d,k} = \mathcal{N}(0, \mathbf{R}_w) = \begin{pmatrix} \mathbf{n}_{\mathbf{a}_t,k} \\ \mathbf{n}_{\omega_t,k} \end{pmatrix} = \mathbf{w}_{d,k}\Delta T,$$

it can be observed from the model above that the motion model adheres to the forward method of discretisation derived in equation C.5. The motion model also adheres to the Markov process assumption, in that it can be completely described through only its transition from the previous state as well as the control inputs.

D Figures & Diagrams

D.1 Schematics & Circuit Diagrams

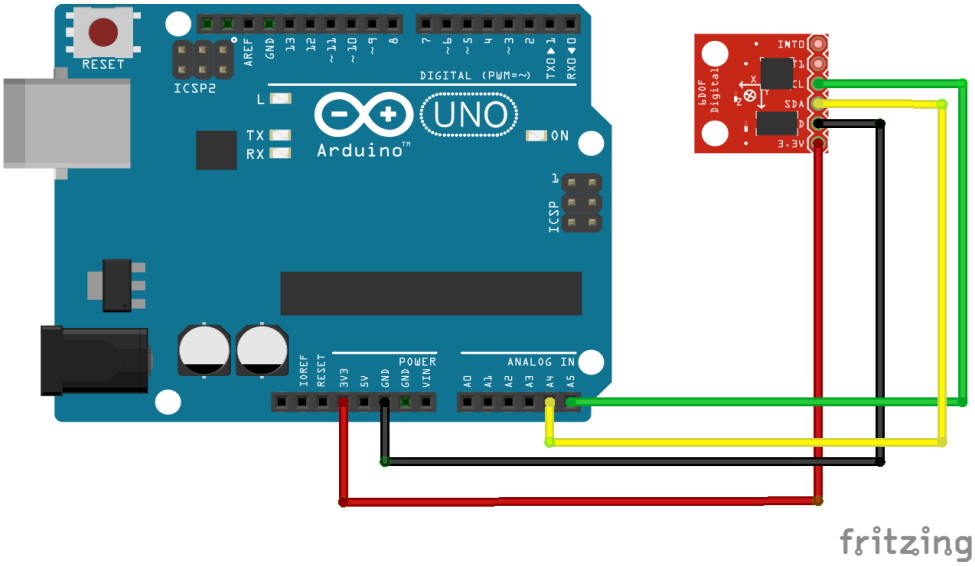


Figure 12: Circuit Diagram of the Inertial Measurement Unit (IMU).

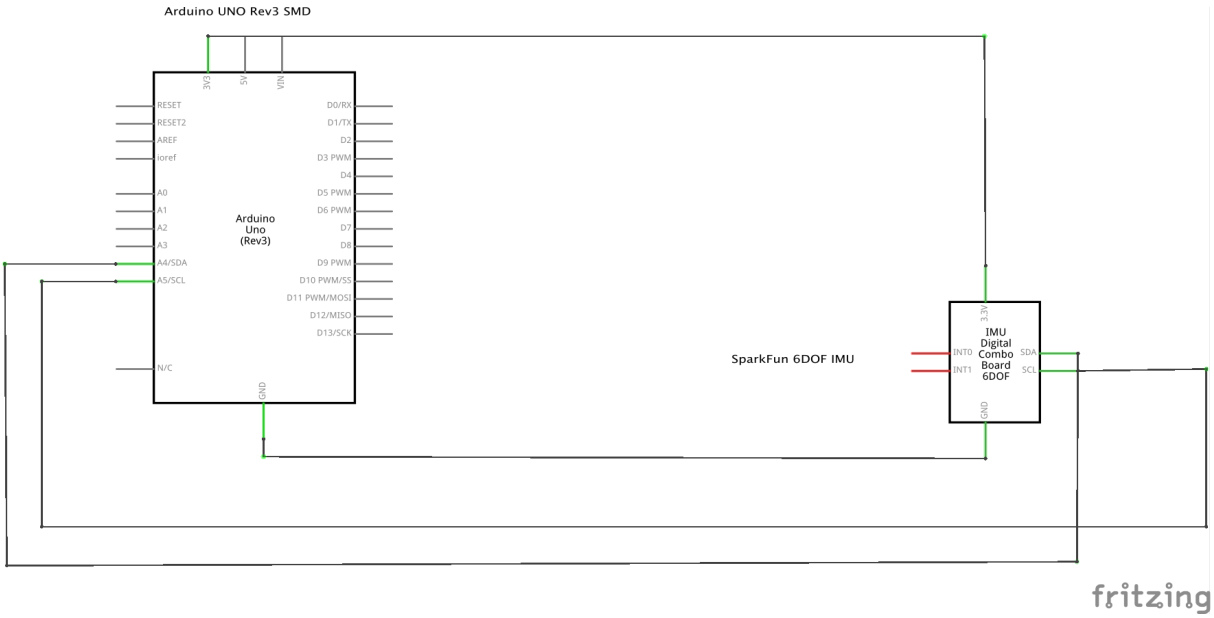


Figure 13: Schematic of the Inertial Measurement Unit (IMU).

These images were created using Fritzing (http://fritzing.org) under the CC Attribution http://creativecommons.org/licenses/by-sa/3.0/. The authors request that they only be mentioned.

References

[1] M. H. Hebert, *Intelligent Unmanned Ground Vehicles: Autonomous Navigation Research at Carnegie Mellon*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.

[2] T. Brogårdh, "Present and future robot control development—an industrial perspective," *Annual Reviews in Control*, vol. 31, no. 1, pp. 69–79, 2007.

[3] H. P. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, pp. 116–121, 1985.

[4] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intellegence in the New Milenium* (G. Lakemeyer and B. Nebel, eds.), Morgan Kaufmann, 2002.

[5] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[6] M. W. M. Gamini Dissanayake and Paul Newman and Steven Clark and Hugh F. Durrant-whyte and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 229–241, 2001.

[7] S. localization and mapping: part I, "Durrant-whyte, hugh and bailey, tim," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 2, pp. 99–110, 2006.

[8] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *AAAI/IAAI*, pp. 593–598, 2002.

[9] F. 2.0, "Montemerlo, michael and thrun, sebastian," *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*, pp. 63–90, 2007.

[10] Z. Chen, "Bayesian filtering: From kalman filters to particle filters and beyond," in *Statistics*, vol. 182, pp. 1–69, 2003.

[11] R. H. G. S. M. K. Sebastian Hilsenbeck, Andreas Moller and E. Steinbach, "Scale-preserving long-term visual odometry for indoor navigation," in *International Conference on Indoor Positioning and Indoor Navigation*, vol. 13, 2012.

[12] M. Chli, "Slam: Simultaneous localization and mapping." http://margaritachli.com, July 2011.

[13] A. Davison, I. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, pp. 1052–1067, June 2007.

[14] J. Sola, "Consistency of the monocular ekf-slam algorithm for three different landmark parametrizations," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3513–3518, IEEE, 2010.

[15] G. K. D. Murray, "Parallel tracking and mapping on a camera phone," in *Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality*, (Orlando), October 2009.

[16] M. V. Peter Gemeiner, Andrew Davison, "Improving localization robustness in monocular SLAM using a high-speed camera," in *Proceedings of Robotics: Science and Systems IV*, (Zurich, Switzerland), June 2008.

[17] H. Strasdat, J. M. M. Montiel, and A. Davison, "Scale drift-aware large scale monocular slam," in *Proceedings of Robotics: Science and Systems*, (Zaragoza, Spain), June 2010.

[18] J. Civera, A. J. Davison, and J. M. M. Montiel, "Unified inverse depth parametrization for monocular slam," in *In Proceedings of Robotics: Science and Systems*, 2006.

[19] S. Fu, H. ying Liu, L. fang Gao, and Y.-X. Gai, "Slam for mobile robots using laser range finder and monocular vision," in *Mechatronics and Machine Vision in Practice, 2007. M2VIP 2007. 14th International Conference on*, pp. 91–96, 2007.

[20] S. T. W. B. D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, pp. 9–53. The MIT Press, 2005.

[21] S. T. W. B. D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[22] A. J. D. I. D. N. D. Molton, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1052–1067, June 2007.

[23] A. J. Davison and D. W. Murray, "Mobile robot localisation using active vision," in *ECCV (2)* (H. Burkhardt and B. Neumann, eds.), vol. 1407, pp. 809–825, Springer, 1998.

[24] J. Shi and C. Tomasi, "Good features to track," pp. 593–600, 1994.

[25] S. Albrecht, "An analysis of visual mono-slam," Master's thesis, Universit´at Osnabr´uck, October 2009.

[26] R. Swaminathan and S. K. Nayar, "Nonmetric calibration of wide-angle lenses and polycameras," in *IEEE Transactions on Pattern Analysis and Machine Intellegence*, vol. 22, pp. 1172–1178, October 2000.

[27] F. Servant, P. Houlier, and E. Marchand, "Improving monocular plane-based slam with inertial measures," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, (Taipei, Taiwan,), pp. 3810–3815, 2010.

[28] A. J. Davison, *Models and State Representation in Scene: Generalised Software for Real-Time SLAM*, 2007.

[29] "The imaging source," May 2015.

[30] G. Bradski, "The opencv library," *Dr. Dobb's Journal of Software Tools*, 2000.