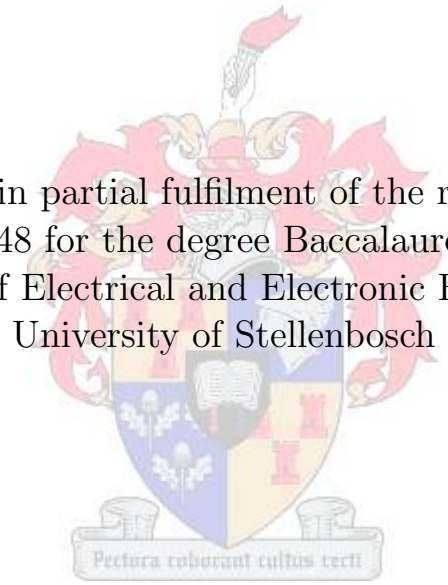
Monocular Vision Based SLAM Using Kinematic State Estimation

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

Contents

List of Figures

List of Tables

1 Introduction

1.1 Robotic Localisation and Mapping

Map building, the idea of constructing a map of a previously unknown environment remains a favourable field of research amongst the robotics community. Briefly, robotics incorporates the observation of surrounding environments as well as the manipulation thereof, through the utilisation of electro-mechanical devices (robots). Its application in modern day society ranges from planetary exploration to assembly lines. Robotic systems - like real world systems - are situated in the physical world. 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. The following five factors invariably lead to an increase in uncertainty: environments, sensors, robots, models and computation.

Probabilistic robotics is an alternative approach to robotics that acknowledges uncertainty; a factor that until only recently, has been generally ignored in the field of robotics. The key idea of probabilistic robotics is to model uncertainty mathematically, using the calculus of probability theory. Probabilistic robotics aims to utilise these uncertainty models in order to provide probabilistic algorithms that will ultimately allow meaningful comparisons between **reliably obtained** sensor data and an appropriate model. Such algorithms have since, provided the fundamental basis for the advancement of many conceptual implementations in the *mobile robotics* field.

One such concept, namely *autonomous robot navigation*, is of great importance to the aforementioned concept of map building. Autonomous navigation has seen significant advances over the past decade or so, yielding a number of successful implementations - none more so than Simultaneous Localisation and Mapping (SLAM). The extent of the aforementioned advancements can be observed in an overview of map generating techniques presented in [1]. The SLAM problem can be briefly described as utilising a robot's sensor information in order to construct a map of the surrounding environment, while concurrently approximating it's location within this map - all autonomously. The relationship regarding the map construction as well as the localisation within the map remains essential to the SLAM problem. If the localisation technique is evidently erroneous, subsequently obtained sensor information will be incorrect, resulting in map approximations that differ from the actual state of the environment. An incorrectly modelled environment however, will render sensor information gathered by the robot useless; as this information will not correlate with those expected by the constructed map. Ultimately, the resulting localisation approximation will drift over time and eventually become erroneous and useless.

Many modern realisations of SLAM rely on underlying probabilistic methods, namely that of *recursive state estimation*, for successful implementation. Methods pertaining to recursive state estimation vary according to the application of the system. The most commonly used probabilistic methods include *optimal filtering* techniques such as the Kalman filter (as well as all it's variants), as well as *sequential importance sampling* (SIS) techniques such as the *particle filter*. A summary regarding these aforementioned probabilistic approaches can be found in [2]. Briefly, the differences regarding the aforementioned methods can be described as follows: a Kalman filter in a mono-modal approach, meaning that only a single hypothesis is considered upon modelling, whereas the particle filter allows multiple hypothesis to be considered. Other fundamental parameters and specifications also determine which specific implementation of the SLAM problem is most applicable. Oftentimes, these parameters mainly include 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 (e.g. two-dimensional (2D) vs. three-dimensional (3D)) as well as their sparsity representation (e.g. point clouds, occupancy grids or sparse sets) are considered upon a suitable SLAM implementation. Generally, the main distinguishment between the various realisations of SLAM is the choice of sensor. Ultimately the choice of sensor results in the two popular approaches that are to be further explained: *range finder approaches* and *vision based approaches*:

1.1.1 Range Finder Approaches

Range finder approaches utilise a beam or a wave to determine the distance to an obstacle in the environment. Generally, the distance to an obstacle from the range finder can be obtained through measuring the time it takes for a pulse generated in a beam/wave to return to the transmitter, after it has reflected off the obstacle. Two-dimensional range finders are only able to obtain these aforementioned distances in a given plane at an opening angle (e.g. range finders with an opening angle of 180° obtain all range measurements within a 180° field of view in the given plane). Three-dimensional range finders however, aren't limited to a single plane and can obtain range measurements in all planes within a field of view equivalent to the opening angle. The 3D point in space where the pulse is reflected can then be reconstructed given that the orientation of the range finder are known, resulting in a 3D point cloud. Range finder approaches include sonar, laser, infrared and ultrasonic. Laser range finders, more commonly referred to as LIDAR (Light Detection And Ranging) are generally considered to provide the best accuracy, especially with regard to depth measurements. The choice of range finder however, varies dependent on the application and specifications of the system at hand.
Range finders obtain large amounts of data regarding the range to obstacles in the environment, especially in the 3D map case. As a result, non-probabilistic methods (as opposed to those previously discussed) are implemented. One such method, namely *scan matching*, aims to merge overlapping point clouds into a single point cloud. This approach, like many range finder approaches however, generally present a great cost in both processing as well as finance.

1.1.2 Vision Based Approaches

Vision based approaches utilise the data captured from cameras. Camera's project the 3D features they observe in the world around them onto a 2D *image plane*. Once projected, the 2D image is then digitised into pixel coordinates where they can then be interpreted, analysed and manipulated in software. There has been a great improvement in the quality of data captured by cameras in the past decade, allowing very reliable projections of the environment. Additionally, cameras are compact, reliable and relatively low-cost with respect to alternative sensors used in SLAM implementations.

Camera systems can be configured in different ways to realise SLAM, utilising anything from a single camera to many calibrated cameras. Each configuration yields it's unique advantages (as well as disadvantages) with successful published implementations existing for each configuration. The two most popular implementations include that of a single camera system and that of a stereo pair. Each of the aforementioned implementations are briefly described below along with their respective pros and cons:

1. **Single Camera**
   Single camera systems (a.k.a monocular systems) rely on using the image data of a single camera only. Although there have been various successful implementations and variations using single camera systems [3, 4, 5], it remains an issue that the depth of a feature cannot be recovered from a single image. The utilisation of a single camera however, ensures a lower cost as well as a relatively simple SLAM realisation as apposed to that of alternative SLAM approaches.

2. **Stereo Calibrated Camera Pair**
   Stereo camera pairs utilise two identical cameras fixed at a given distance between them. These systems are able to effectively recover the depth of a feature in an image from a single observation, through matching a this feature across both images. The setup for such a system however, is somewhat complex as the individual cameras are required to be carefully calibrated. Additionally, stereo systems generally provide a greater computational and financial expense than that of a monocular system.

A meaningful comparison between the aforementioned algorithms is given by [6].

1.2 Project Proposal and Outline

The ultimate goal of any three dimensional (3D) SLAM approach, is to obtain a probabilistic 3D map of point features, representing at every time instance, the estimates of both the state of the robot as well as the positions of every feature observed. The previously mentioned features of interest are more commonly referred to as *landmarks* and the aforementioned terms will, from hereon in, be used synonymously. Most importantly though, the map is to contain the *uncertainty* associated with each of the aforementioned estimates.
The past decade however, has yielded many acceptable and impressive solutions as in [7, 8, 9], to the aforementioned requirements using a single camera, with the original proposal presented by Davison et al. [3], more commonly referred to as MonoSLAM (Monocular vision based SLAM). In MonoSLAM, the robot is set to **only** obtain sensor information through the utilisation of a singular camera system. There are however, inherent disadvantages of such a system: Firstly, the utilisation of a single camera prevents the system from immediately obtaining an accurate depth estimate. Landmarks are required to be examined from various viewpoints before an appropriate depth estimation can be concluded. Secondly, the motion model (namely a constant velocity model) constrains the movement of the system to smooth, constant trajectories. In the event that erratic, immediate disturbances act upon the system, the pose of the robot is generally lost, and in most cases irrecoverable. Lastly, because no sufficient knowledge regarding the movement of the robot exists - its practical application is vastly limited.
The proposed approach to be presented in the following paper, aims to improve the system to limit of these disadvantages, namely the latter two. This paper aims to extend the aforementioned system of Davison with the utilisation of *additional* sensor information regarding the robot's motion. An inertial measurement unit (IMU) will provide this additional information, recording the movement of the robot through space as a result of the control inputs to the system. The additional data allows the system to be modelled as a rigid, kinematic body upon which kinematic estimation can be applied. Inevitably, a kinematic estimator allows the system a greater knowledge of its movement - because it is being measured through inertial sensors.

The process regarding the construction of this map of features, namely that of recursive state estimation, is to be implemented through the use of an (Extended) Kalman filter. The map initially, completely void of any landmarks, is recursively updated according to the subsequent fusions of both the prediction and the measurement presented to the Kalman filter. As new (potentially interesting) features are observed, the state estimates of both the camera as well as the landmarks are both updated - augmenting the state vector with additional features (if indeed they are observed) while deleting any landmarks that are no longer of interest. In order to obtain the best possible result, the algorithm should strive to obtain accurate state estimation regarding the movement of the robot as well as a sparse set of high-quality landmarks. Each of the steps regarding the realisation of the previously mentioned map of features are to be defined, described and analysed in the sections that follow.

The intended contribution of this paper realises an improvement in not only the state estimation of the system, but localisation as a whole. Additionally the kinematic estimator embedded within the system should serve as a means to integrate the proposed system as part of a larger system regardless of that system's physical model. Moreover, the proposed system is required to operate in real time at a rate of at least the 30 Hz for 100 landmarks; as achieved by Davison in his original work presented in [3].

Before continuing, it is important to consider and understand the notation used in the sections that follow. Two separate coordinate systems are to be considered, namely the *fixed* inertial reference fame system W and the cameras free coordinate frame system, more commonly referred to as the body frame C. System variables defined within either of the aforementioned coordinate systems, are from here on in, to be designated a superscript to establish in which coordinate system it may be relevant (e.g. x^W). Derivatives of parameters are denoted through a prime symbol, second derivates are denoted through a double dot symbol and so forth; for instance the derivative of position x will be denoted as $\dot{x}$ and its second derivative denoted as $\ddot{x}$. Vectors will be printed in bold and non-italics to better distinguish them from scalers. An example can be shown regarding the variable x: $\mathbf{x}$ denotes a vector while its scalar counterpart would be represented as x.

TODO: Glossary, List of abbreviations and List of symbols

2 Recursive State Estimation

Probabilistic robotics yields a unique yet fundamental concept at its core, that is: estimating a state through sensor data. Quantities exist that are not directly observable, yet are still able to be obtained through sensor data. Sensors though, obtain limited data regarding certain quantities and most importantly, are affected and often corrupted by *noise*.
Recursive state estimation then, seeks to recover state variable from the obtained sensor data bearing in mind the previously mentioned limitations. Probabilistic state estimation algorithms - to be investigated in this section - compute *belief* distributions regarding state variables. The following section will provide a brief, yet concise introduction to these various algorithms. 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 paper, 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 paper. It is worthwhile to note that theory in this chapter resembles material from Probabilistic Robotics (2005) [10] and is adapted therefrom for convenience sake.

2.1 Bayes Filter

Upon considering probabilistic robotics, a key concept worth describing is the previously mentioned *belief* of a robot. Briefly, the belief represents the robot's understanding regarding the state of it's own dynamics as well as the dynamics of the surrounding environment. This critical concept proves a fundamental basis in probabilistic robotics. The belief can be represented as a conditional probability distribution whereby each possible scenario (state) is a signed a probability (density). Mathematically the belief with regard to a state variable x_t is denoted as follows:

$$bel(x_t) = p(x_t \mid z_{1:t},\ u_{1:t}), \tag{2.1}$$

A brief description would yield that the distribution above describes, for a given time instance 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 $u_{1:t}$.
Considering that the state of the robot is constantly updated at every time-step t and that each update is dependent upon the state at the previous time-step, it is essential that the algorithm required be recursive in nature. The *Bayes Filter* algorithm provides precisely such a procedure. The algorithm calculates the belief distribution stated in equation 2.1 from the observation and control data. Table 2.2 presents a pseudo-algorithmic interpretation of the Bayes Filter algorithm [10]:

Table 2.1: The Bayes Filter Algorithm

Input:	previous belief $bel(x_{t-1})$, control input/s u_t, measurement/s z_t
Output:	current belief $bel(x_t)$
for all x_t:	
1.	$\overline{bel}(x_t) = \int p(x_t \mid u_t,\ x_{t-1}) bel(x_{t-1}) dx_{t-1}$
2.	$bel(x_t) = \eta p(z_t \mid x_t) \overline{bel}(x_t)$
3.	end for.

The recursive nature of the algorithm can thus be seen from table 2.1; whereby the belief $\overline{bel}(x_t)$ 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 as well as the probability that the control inputs induce a transition form x_{t-1} to x_t. Thereafter, the measurement update seeks to determine the belief in the event that a measurement has been observed that may correct any errors presented in the previous prediction step. The mathematical derivation of the Bayes Filter contains many assumptions and further technicalities. The techniques presented in this paper though, require only a basic comprehension of it's implementation. Furthermore, detailed analysis of the Bayes Filter can be obtained from Probabilistic Robotics by Thrun et al. [11]

2.2 Gaussian Filters

As previously discussed, the Bayes Filter possesses many different derivations and variants thereof. Amongst these derivations, 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:

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

where the density across the state variable 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). Here from, a number of recursive filter algorithms can be derived, two of which are examined in this paper: 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 following segments of this section.

2.2.1 Kalman Filter

Probably the most fundamental (and popular) of all Gassian filter algorithms, is the *Kalman Filter*. The Kalman filter can be briefly described as the algorithm that enables the realisation of many practical recursive estimation systems, including the SLAM problem. It remains a popular, well studied technique for filtering and prediction of linear systems that contains uncertainty - typically uncertainties which are Gaussian in nature. 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 μ_t and covariance Σ_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 [Probabilistic Robotics]):

1. The state transition model probability - with function $g(\mathbf{u}_t,\ \boldsymbol{\mu}_{t-1})$ - **must** be a *linear* function with additive Gaussian (process) noise.

2. The observation model probability - with function $h(\bar{\boldsymbol{\mu}}_t)$ - **must** be a *linear* function with additive Gaussian (measurement/sensor) noise.

3. The initial belief $bel(x_0)$ must be normally distributed.

In the context of Simultaneous Localisation and Mapping (SLAM), a Kalman Filter algorithm seeks to determine the **optimal** trade off between the state of the robots pose and the position of the landmarks within the map - given process and measurement noise. Moreover, the solution to this specific implementation of the SLAM problem, takes a probabilistic form where the belief with regard to state variable $\mathbf{x}_t$ is denoted as shown in equation 2.1:

$$bel(\mathbf{x}_t) = p\big(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\big), \tag{2.3}$$

Like the Bayes Filter, the Kalman filter too is executed in two (sequential) steps: the *prediction step* and the *update step*. Firstly, the prediction step aims to estimate a state into which the system will be transitioned from the previous state estimate ($\boldsymbol{\mu}_{t-1}$, $\boldsymbol{\Sigma}_{t-1}$) as a result of a set of internal and/or external dynamics to the system. These dynamics are typically described through the state transition function $g(\mathbf{u}_t, \boldsymbol{\mu}_{t-1})$. Once an estimate is obtained for the transitioned state estimate ($\bar{\boldsymbol{\mu}}_t$, $\bar{\boldsymbol{\Sigma}}_t$), a measurement prediction is made in order to provide the expected measurements provided that the system were to find itself within the estimated transitioned state. These measurements are obtained through an observation model which has a function $h(\bar{\boldsymbol{\mu}}_t)$. Thereafter, an actual measurement, $\mathbf{z}_t$ is then obtained through the system sensors in order to determine the actual state of the system ($\boldsymbol{\mu}_t$, $\boldsymbol{\Sigma}_t$). Ultimately, the actual state of the system and the previously predicted state are then compared with one another in order to obtain the (optimally weighted) *Kalman gain*: This entity yields minimum mean-squared error in the estimate and ultimately provides the desired optimal trade-off between the measurement and the estimate. Intuitively, the Kalman gain controls how much the systems "trusts" the measurements over the estimate. 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-algorithmic representation the aforementioned steps and also incorporates a systematical and mathematical representation for the following system definition of the state transition and the observation models respectively [10]:

$$\begin{aligned} g(\mathbf{u}_t, \mu_{t-1}) : \mathbf{x}_t &= \mathbf{A}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{x}_t + \mathbf{w}_t \\ h(\bar{\mu}) : \mathbf{z}_t &= \mathbf{C}_t\mathbf{x}_t + \mathbf{v}_t \end{aligned} \tag{2.4}$$

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

Table 2.2: The Kalman Filter Algorithm

Input:	previous mean μ_{t-1} and covariance Σ_{t-1}, control inputs $\mathbf{u}_t$, measurements $\mathbf{z}_t$
Output:	mean μ_t, covariance Σ_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.	$\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_t\mathbf{C}_t^T(\mathbf{C}_t\bar{\boldsymbol{\Sigma}}_t\mathbf{C}_t^T + \mathbf{Q}_{v,t})^{-1}$
4.	$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t[\mathbf{z}_t - \mathbf{C}_t\bar{\boldsymbol{\mu}}_t]$
5.	$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{C}_t)\bar{\boldsymbol{\Sigma}}_t$

TODO: Other relevant information regarding the EKF such as complexity an other unique characteristics and specifications - perhaps observability.

2.2.2 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 tend to suppress the Gaussian nature of the distribution that is being modelled. Any linear transformation of a Gaussian random variable yields another **different** Gaussian variable. Non-linear transformations of a Gaussian random variable (RV) however, will always yield a non Gaussian RV. The Kalman filter algorithm thus, cannot be implemented system. This phenomenon can be illustrated through figure 1 presented below:

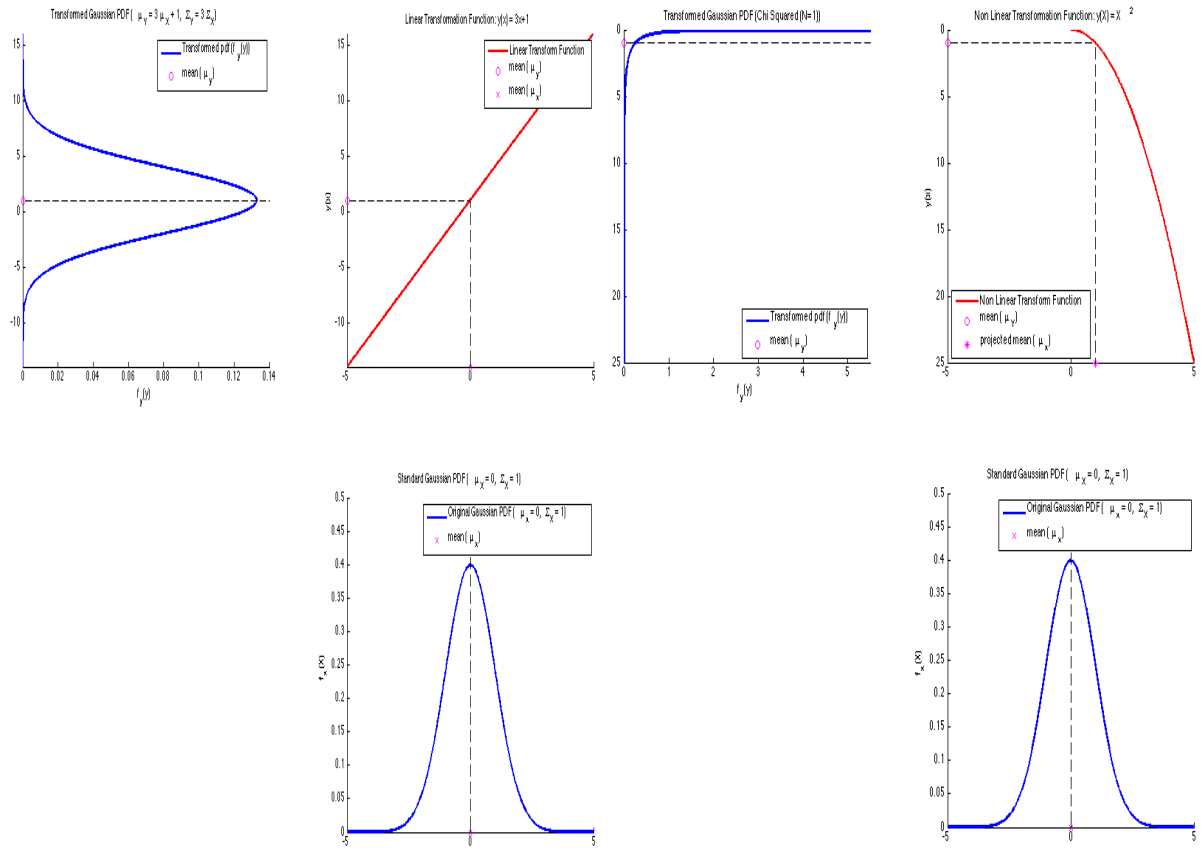


Figure 1: Graphical representation depicting both a linear (left) and a non linear (right) transformation of a Gaussian random variable.

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 $g(\mathbf{u}_t, \mu_{t-1})$, as well as the observation model $h(\bar{\mu}_t)$ 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 1). 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 $\mathbf{F}_t = f'(\mathbf{u}_t, \boldsymbol{\mu}_{t-1})$ 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.

Once linearisation is achieved, the EKF; which behaves otherwise identically in terms of operation to the general Kalman filter, can be implemented upon non-linear systems. 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. Provided that the linearisation point is in close proximity to the mean of the Guassian, the EKF will yield an acceptable approximation from the linearisation process. Other variants of the Kalman filter (such as the Unscented Kalman filter) that aren't discussed in this paper, are then required to be considered.

Table 2.3 below, systematically and mathematically represents the aforementioned steps [10]:

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.	$\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_{\boldsymbol{t}} \mathbf{H}_t^T (\mathbf{H}_t \bar{\boldsymbol{\Sigma}}_{\boldsymbol{t}} \mathbf{H}_t^T + \mathbf{Q}_{v,t})^{-1}$
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 Design: EKF Monocular Vision Based SLAM Using Kinematic State Estimation

The ultimate goal of the approach presented here, is to obtain a probabilistic three dimensional (3D) map of features, representing at every time instance, the estimates of both the state of the camera as well as the cartesian positions of every feature observed. This scenario is one approach of the previously discussed SLAM problem. Such a problem, is typically solved through the utilisation of the EKF. The goal of a single camera based SLAM algorithm (Monocular vision based SLAM), is to ultimately realise the objective of obtaining the aforementioned probabilistic map, through the utilisation of a single camera as achieved by Davison et al. [3]. Various **successful** SLAM algorithms exist that utilise sensors other than cameras (laser range finders, ultrasonic sensors etc.) [12, 13, 14]. It can be argued though, that the general approach presented in [3] - more commonly referred to as *MonoSLAM* - as well as the variants thereof [15, 8, 7, 16] can be improved through the utilisation of additional information regarding the motion of the robot. As previously discussed, the MonoSLAM system utilises a single camera as it's only sensor. Apart from the inability to immediately estimate depth, a MonoSLAM system possesses no knowledge regarding it's movement. The approach presented in this paper then, seeks to utilise exactly such information through the utilisation of an inertial measurement unit (IMU) as an extension to the original implementation. IMU's have been successfully implemented in SLAM based systems before, as implemented in [17]. With the addition of an IMU, information regarding the changes in movement and orientation of the camera (namely the linear accelerations and the angular rates) can be obtained and directly *observed*. Ultimately, the stochastic constant velocity motion model of the general MonoSLAM algorithm can then be replaced with a kinematic estimation based motion model - one that is initially assumed to be more accurate - as it contains a considerably larger amount of information directly associated with a robot's movement.

Furthermore, the remaining components of the proposed approach presented in this paper are identical to that of the general 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.
Moreover, this particular vision based approach aims to use salient image *patches* as long term landmarks as presented in [18, 19]. These aforementioned patches are typically large in size (11×11 pixels) and are obtained through the image detection operator of Shi and Tomasi [20] 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. Good features imply that feature remains visible given the relative position of the camera and the feature, as well as the camera position at which the feature is initially saved. A new feature is initialised using the image operator of Shi and Tomasi [20] 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 [3].

The following section sets out to define the necessary segments of the MonoSLAM algorithm in the context of this paper alone. Initially, the adequate state representation of the system at hand will be denoted, whereby the components of the state vector - namely the camera position and cartesian feature states - will be defined and discussed. Additionally, the affect of the proposed extension - namely the IMU measurements as the control inputs to the system - will be properly defined and described. Furthermore, this section will seek to use the aforementioned definitions to completely define the two sequential steps required to successfully implement the EKF, namely the *prediction* and *update* steps.

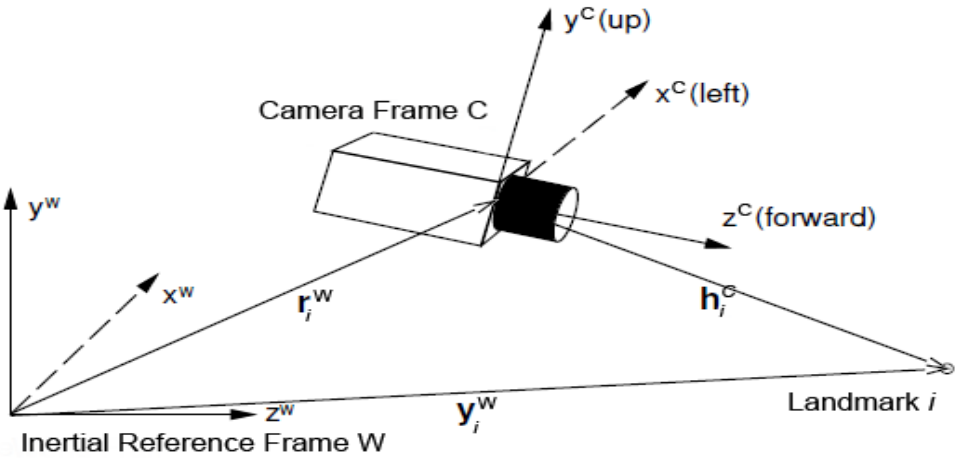


Figure 2: Cartesian Representation of the Reference Frames - REF:DAVISON

3.1 State Representation

Generally, a state can be defined as any facet that has the ability to impact the future. In the context of this particular paper, the states will comprise of all facets that impact the future of both the robot and the environment dynamics. As per the definition of the EKF, it is essential that the system possesses a model to estimate future states. This model is commonly referred to as the previously discussed state transition model. All relevant state estimates are embedded within the state vector $\mathbf{x}_t$, which is comprised of two parts, the camera state $\mathbf{x}_v$ and the landmark position estimates $\mathbf{y}_n$ respectively. The camera state provides the estimate for the robot's pose at each time-step and the landmark estimates provide the landmark's estimated position within the map.
Mathematically, the probabilistic map is typically represented through a mean state vector $\mathbf{x}_t$ and a covariance matrix $\mathbf{P}_{nN}$. The mean state vector, as previously mentioned, is a a single column vector containing the estimates of the camera as well as the landmark positions, and $\mathbf{P}_{nN}$ is a square matrix containing the covariances of each state with respect to every other state. These quantities can be mathematically shown as follows:

$$\mathbf{x}_t = \begin{pmatrix} \mathbf{x}_{v,t} \\ \mathbf{y}_{1,t} \\ \mathbf{y}_{2,t} \\ \vdots \\ \mathbf{y}_{n,t} \end{pmatrix}, \quad \mathbf{P}_{nN} = \begin{pmatrix} P_{x,x} & P_{x,y_1} & P_{x,y_2} & \cdots & P_{x,y_N} \\ P_{y_1,x} & P_{y_1,y_1} & P_{y_1,y_2} & \cdots & P_{y_1,y_N} \\ P_{y_2,x} & P_{y_2,y_1} & P_{y_2,y_2} & \cdots & P_{y_2,y_N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P_{y_n,x} & P_{y_n,y_1} & P_{y_n,y_2} & \cdots & P_{y_n,y_N} \end{pmatrix}, \tag{3.1}$$

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.

3.1.1 Camera Position State Representation

The following concept describes a suitable method to represent all relevant information regarding the camera's position and orientation in a 3D space. According to most implementations of robot localisation, there exists no contrast between the concepts of a camera state $\mathbf{x}_v$ and a camera position state $\mathbf{x}_p$. It is therefore important to note that a position state - containing the required information regarding a robots position - is merely an element of the camera state vector (where additional information required to describe the camera's position forms the remainder of the camera state 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 *quarternion* 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.

A quarternion, as previously mentioned, represents the camera's orientation. 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:

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

The process of computing the quarternion 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 - separate to 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 position state vector $\mathbf{x}_p$ comprising of the 3D position vector, $\mathbf{r}^W$ and the unit orientation *quaternion* $\mathbf{q}^{WC}$. 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 states (namely the linear velocity) is required to describe the control inputs effect on the actual position.

3.1.2 Cartesian 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.

3.1.3 Control Inputs

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 observable control inputs obtained through an inertial sensor in order to derive a motion model that is potentially more accurate than that in [3].

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. (2007), 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. Even though there have been proven successful implementations regarding the aforementioned approach (as well as other variants and extensions thereof [16, 8]), the model contains very little, if any information on the movement of the camera. It can be assumed that the utilisation of additional information regarding the camera's movement (through the proposed inertial sensors) will provide greater accuracy upon state estimation - especially with regard to

the transitioning thereof.

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 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 expressed mathematically as follows:

$$\mathbf{f}_t = \mathbf{R}^{CW}(\mathbf{a}_t - \mathbf{g}_t) \tag{3.5}$$

with $\mathbf{a}_t$ is the linear acceleration vector, $\mathbf{g}_t$ 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 can be mathematically defined as follows:

$$\mathbf{R}^{CW} = \begin{pmatrix} q_{0,t}^2 + q_{x,t}^2 - q_{y,t}^2 - q_{z,t}^2 & 2(q_{x,t}q_{y,t} - q_{0,t}q_{z,t}) & 2(q_{x,t}q_{z,t} - q_{0,t}q_{y,t}) \\ 2(q_{x,t}q_{y,t} + q_{0,t}q_{z,t}) & q_{0,t}^2 - q_{x,t}^2 - q_{y,t}^2 - q_{z,t}^2 & 2(q_{y,t}q_{y,z} - q_{0,t}q_{x,t}) \\ 2(q_{x,t}q_{z,t} - q_{0,t}q_{y,t}) & 2(q_{y,t}q_{z,t} + q_{0,t}q_{x,t}) & q_{0,t}^2 - q_{x,t}^2 - q_{y,t}^2 + q_{z,t}^2 \end{pmatrix} \tag{3.6}$$

Once obtained, these measurements (not to be confused with the EKF's measurements) form the 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 is mathematically described as follows:

$$\mathbf{u}_t = \begin{pmatrix} \mathbf{a}_t \\ \boldsymbol{\omega}_t \end{pmatrix} = \begin{bmatrix} \ddot{x}_t & \ddot{y}_t & \ddot{z}_t & \omega_{x,t} & \omega_{y,t} & \omega_{z,t} \end{bmatrix}^T \tag{3.7}$$

Because the IMU measurements gather the actual data through exteroceptive sensors, namely an accelerometer 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:

$$\mathbf{w}_t = \begin{pmatrix} \mathbf{n}_{\mathbf{a},t} \\ \mathbf{n}_{\omega,t} \end{pmatrix} = \begin{bmatrix} n_{\ddot{x}_t} & n_{\ddot{y}_t} & n_{\ddot{z}_t} & n_{\omega_{x,t}} & n_{\omega_{y,t}} & n_{\omega_{z,t}} \end{bmatrix}^T \tag{3.8}$$

with the aforementioned noise model yielding for each of the above elements, a Gaussian random variable.
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.

3.2 Prediction Step

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:

$$\begin{aligned} bel(\mathbf{x}_t) &= p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) \\ &= \frac{1}{\sqrt{|2\pi \mathbf{R}_w|}} \exp\left\{ \frac{1}{2} \big[\mathbf{x}_t - g(\boldsymbol{\mu}_t,\ \boldsymbol{\mu}_{t-1}) - \mathbf{G}_t(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})\big]^T \right. \\ &\left. \mathbf{R}_w^{-1} \big[\mathbf{x}_t - g(\boldsymbol{\mu}_t,\ \boldsymbol{\mu}_{t-1}) - \mathbf{G}_t(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})\big] \right\}, \end{aligned} \tag{3.9}$$

where G_t represents the Jacobian of the state transition motion.

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.

3.2.1 Mean Estimate: 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 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, non linear discrete-time model is adopted to approximate the model. We begin by recalling all relevant states and control inputs:

$$\begin{aligned} \mathbf{x}_t &= \begin{bmatrix} \mathbf{r}_t^W & \mathbf{q}_t^{WC} & \mathbf{v}_t^W \end{bmatrix}^T \\ &= \begin{bmatrix} x_t & y_t & z_t & q_{0,t} & q_{1,t} & q_{2,t} & q_{3,t} & \dot{x}_t & \dot{y}_t & \dot{z}_t \end{bmatrix}^T \\ \mathbf{u}_t &= \begin{bmatrix} \mathbf{a}_t^C & \boldsymbol{\omega}_t^C \end{bmatrix}^T \\ &= \begin{bmatrix} \ddot{x}_t & \ddot{y}_t & \ddot{z}_t & \omega_{x,t} & \omega_{y,t} & \omega_{z,t} \end{bmatrix}^T \end{aligned} \tag{3.10}$$

The state transition function $g_v(\mathbf{u}_t, \boldsymbol{\mu_{t-1}})$ defined at the current time t, is dependent on both the current control inputs $\mathbf{u}_t$ as well as the previous mean $\boldsymbol{\mu}_{t-1}$:

$$\begin{aligned} g_v(\mathbf{u}_t, \boldsymbol{\mu}_{t-1}) = \begin{pmatrix} \mathbf{r}_t^W \\ \mathbf{q}_t^{WC} \\ \mathbf{v}_t^W \end{pmatrix} &= \begin{pmatrix} \mathbf{r}_{t-1}^W + \mathbf{v}_t^W \Delta T \\ \mathbf{q}_{t-1}^{WC} \otimes \mathrm{quat}\big(\boldsymbol{\omega}_t^C \Delta T\big) \\ \mathbf{v}_t^W + \mathbf{a}_t^C \Delta T \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{r}_{t-1}^W + \dot{\mathbf{r}}^W \Delta T \\ \mathbf{q}_{t-1}^{WC} \otimes \mathrm{quat}\big(\boldsymbol{\omega}_t^C \Delta T\big) \\ \dot{\mathbf{r}}_t^W + \mathbf{R}_t^{CW}\big(\ddot{\mathbf{r}}_t^C \Delta T\big) \end{pmatrix} \\ &= \bar{\boldsymbol{\mu}}_t \end{aligned} \tag{3.11}$$

where ΔT is defined as the sample period between the previous time $t-1$ and t and $\mathrm{quat}\big(\boldsymbol{\omega}_t^C \Delta T\big)$ denotes the quartering corresponding to the rotation represented by $\dot{\boldsymbol{\omega}}_t^C \Delta T$. Information regarding the change in linear acceleration is directly obtained form the control input data gathered by the IMU and can be numerically integrated in order to illustrate it's effect on the systems linear velocity and ultimately, translational position.

In order to compute the aforementioned quaternion, the rate at which the camera's rotational degrees of freedom are changing is required. The control inputs 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 γ is defined as follows:

$$\gamma = \Big(\boldsymbol{\theta}, \|\boldsymbol{\theta}\|\Big) = \left(\begin{pmatrix} \theta_x \\ \theta_y \\ \theta_z \end{pmatrix}, \big\|\boldsymbol{\omega}_t^C \Delta T\big\| \right) = \left(\begin{pmatrix} \frac{\omega_{t,X}^C \Delta T}{\|\boldsymbol{\omega}_t^C \Delta T\|} \\ \frac{\omega_{t,Y}^C \Delta T}{\|\boldsymbol{\omega}_t^C \Delta T\|} \\ \frac{\omega_{t,Z}^C \Delta T}{\|\boldsymbol{\omega}_t^C \Delta T\|} \end{pmatrix}, \big\|\boldsymbol{\omega}_t^C \Delta T\big\| \right) \tag{3.12}$$

where $\omega_{t,\beta},\ \beta \in \{X, Y, Z\}$ denotes the angular velocity about each respective coordinate axis. The result in 3.12 is then represented as a unit quarternion denoting the same rotation:

$$\mathbf{q} = \begin{pmatrix} \cos\frac{\alpha}{2} & \frac{\theta_x}{\|\boldsymbol{\theta}\|}\sin\frac{\alpha}{2} & \frac{\theta_y}{\|\boldsymbol{\theta}\|}\sin\frac{\alpha}{2} & \frac{\theta_z}{\|\boldsymbol{\theta}\|}\sin\frac{\alpha}{2} \end{pmatrix}^T \tag{3.13}$$

A quaternion multiplication between the orientation quaternion obtained at the previous time step and the angle-axis rotation quaternion in 3.13 results in the camera's final orientation state at the the current time t.

3.2.2 Covariance Update

Before the prediction step can be concluded, the covariance matrix $\bar{\Sigma}_t$, 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 $g_v(\mathbf{u}_t, \boldsymbol{\mu_{t-1}})$ is thus required to realise this procedure. The Jacobian matrix of $g_v(\mathbf{u}_t, \boldsymbol{\mu_{t-1}})$, $\mathbf{G}_t$, can

be mathematically defined as follows:

$$\mathbf{G}_t = \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{t-1}} = \begin{pmatrix} \frac{\partial \mathbf{r}_t^W}{\partial \mathbf{r}_{t-1}^W} & \mathbf{0} & \frac{\partial \mathbf{v}_t^W}{\partial \mathbf{r}_{t-1}^W} \\ \mathbf{0} & \frac{\partial \mathbf{q}_t^{WC}}{\partial \mathbf{q}_{t-1}^{WC}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{v}_t^W}{\partial \mathbf{v}_{t-1}^W} \end{pmatrix} \tag{3.14}$$

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

$$\frac{\partial \mathbf{r}_t^W}{\partial \mathbf{r}_{t-1}^W} = \frac{\partial \mathbf{v}_t^W}{\partial \mathbf{v}_{t-1}^W} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \mathbf{I} \tag{3.15}$$

and

$$\frac{\partial \mathbf{r}_t^W}{\partial \mathbf{v}_{t-1}^W} = \Delta T \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \Delta T \ \mathbf{I} \tag{3.16}$$

The specially defined Jacobian with a partial derivate that is taken with respect to a quarternion 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 3.11 can be more formerly defined as follows:

$$\mathbf{q}_t^C = \text{quat}(\boldsymbol{\omega}_t^C \Delta T) \tag{3.17}$$

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

$$\frac{\partial \mathbf{q}_t^{WC}}{\partial \mathbf{q}_{t-1}^{WC}} = \begin{pmatrix} q_{1,t}^C & -q_{2,t}^C & -q_{3,t}^C & -q_{4,t}^C \\ q_{2,t}^C & q_{1,t}^C & -q_{4,t}^C & q_{3,t}^C \\ q_{3,t}^C & q_{4,t}^C & q_{1,t}^C & -q_{2,t}^C \\ q_{4,t}^C & -q_{3,t}^C & q_{2,t}^C & q_{1,t}^C \end{pmatrix} \tag{3.18}$$

In order to complete the covariance update, the process noise covariance is still required to be updated. The process noise covariance, namely $\mathbf{R}_w$ (not to be confused with the coordinate frame rotational matrix $\mathbf{R}^{CW}$)is described in section 3.1.3. 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:

$$\mathbf{R}_w = \begin{pmatrix} n_{\ddot{x},t} & 0 & 0 & 0 & 0 & 0 \\ 0 & n_{\ddot{y},t} & 0 & 0 & 0 & 0 \\ 0 & 0 & n_{\ddot{z},t} & 0 & 0 & 0 \\ 0 & 0 & 0 & n_{\omega_x,t} & 0 & 0 \\ 0 & 0 & 0 & 0 & n_{\omega y,t} & 0 \\ 0 & 0 & 0 & 0 & 0 & n_{\omega z,t} \end{pmatrix}^2 \tag{3.19}$$

with the perturbation levels $n_{w,t}$ as defined in equation 3.8. Sorry Dr. Van Daalen, I realise after our meeting today that the aforementioned matrix is required to be transformed....

3.3 Correction Step

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:

$$P(\mathbf{z}_t \mid \mathbf{x}_t) = \frac{1}{\sqrt{|2\pi\mathbf{R}_v|}} \exp\left\{\frac{1}{2}\big[\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_t) - \mathbf{H}_t(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t)\big]^T \mathbf{R}_v^{-1}\big[\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_t) - \mathbf{H}_t(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t)\big]\right\}. \tag{3.20}$$

where $\mathbf{H}_t$ represents the Jacobian of the observation model.

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 $\mathbf{h}(\bar{\boldsymbol{\mu}})$.

3.3.1 Vision Based Sensor Theory

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, v) be warped with a *radial distortion* as is done in [18], 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 follows:

$$\begin{aligned} u_d - u_0 &= \frac{u - u_0}{(1 + k_1^2 r^2 + k_2 r^4)} \\ v_d - v_0 &= \frac{v - v_0}{(1 + k_1^2 r^2 + k_2 r^4)}, \\ r &= \sqrt{(u - u_0)^2 + (v - v_0)^2}. \end{aligned} \tag{3.21}$$

3.3.2 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 2, 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:

$$\mathbf{h}_i^W(\bar{\boldsymbol{\mu}}) = \mathbf{R}^{CW}\left(\mathbf{y}_i^W - \mathbf{r}^W\right) = \left(\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} - \mathbf{r}^W\right) \tag{3.22}$$

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 as follows:

$$\mathbf{h}_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} u_0 - fk_u \dfrac{h_{i,x}^R}{h_{i,z}^R} \\ v_0 - fk_v \dfrac{h_{i,y}^R}{h_{i,z}^R} \end{pmatrix} \tag{3.23}$$

where u_0 and v_0 represent the principal point and fk_u and fk_v are the camera calibration parameters described in Appendix C.
It is evident from the model presented in equation 3.23 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.

3.3.3 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* $\mathbf{S}_i$:

$$\mathbf{S}_i = \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{x}_v} P_{xx} \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{x}_v}^T + \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{x}_v} P_{xy_i} \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{y}_i}^T + \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{y}_i} P_{y_i x} \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{x}_v}^T + \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{y}_i} P_{y_i y_i} \frac{\partial \mathbf{u}_{d,i}}{\partial \mathbf{y}_i}^T + \mathbf{R}_v \quad (3.24)$$

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 [3], 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.

3.3.4 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:

$$\mathbf{y}_{pi} = \begin{pmatrix} \mathbf{r}_i^W \\ \hat{\mathbf{h}}_i^W \end{pmatrix} \quad (3.25)$$

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 [3]. 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.

3.3.5 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 [3] in order to realise a successful real-time algorithm.

4 Experimental Configuration & Procedure

4.1 System Overview

4.2 Hardware Configuration

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 3.2.1. The 6DOF IMU board to be used in this implementation is the SparkFun 6 DOF IMU. This board is comprised of a ITG3200 MEMS 3-axis gyroscope and a ADXL345 3-axis linear accelerometer. The IMU is configured with the following properties:

1. 3.3 Volt input
2. I2C interface (this device serves as the *slave*)
3. $\pm 2000^{\circ}$ per second range for the gyroscope
4. $\pm 4g$ range and 10-bit resolution for the accelerometer
5. 100 Hz sample rate with 100 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 prevents Aliasing.

4.2.2 CMOS Machine Vision Camera

4.2.3 Hardware Integration

4.3 Software Configuration

4.4 Single Byte Write Cycle

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 Write Cycle

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

5 Analysis: Testing & Results

A Summary of Work done

B Achieved Exit Level Outcomes

C Theoretical Concepts

C.1 State Space Model

As previously discussed, the Extended Kalman FiIter requires a state transition (motion) 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 - from here on in to be referred to as the **camera** - is comprised of a monocular camera and an attached Inertial Measurement Unit (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 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

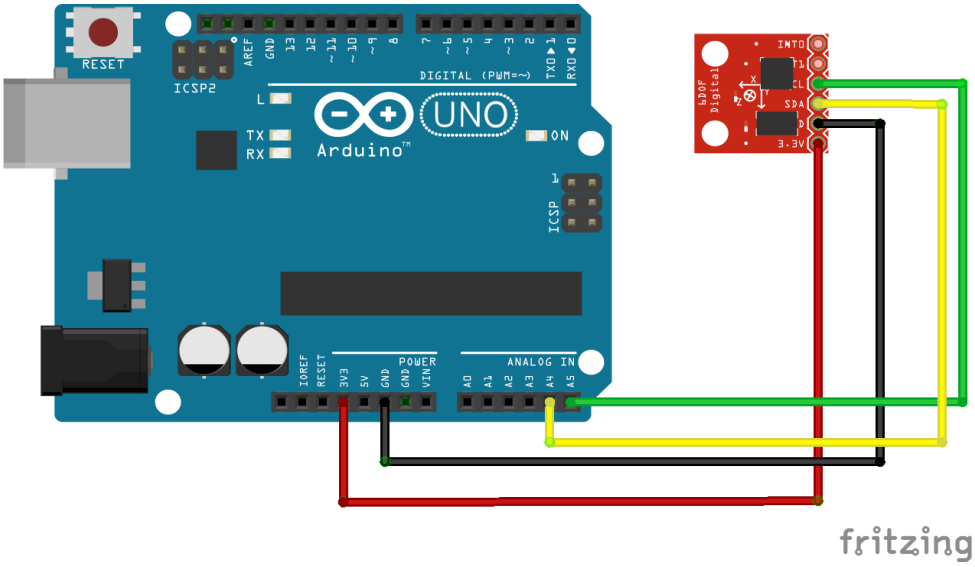


Figure 3: Circuit Diagram of the Inertial Measurement Unit (IMU).

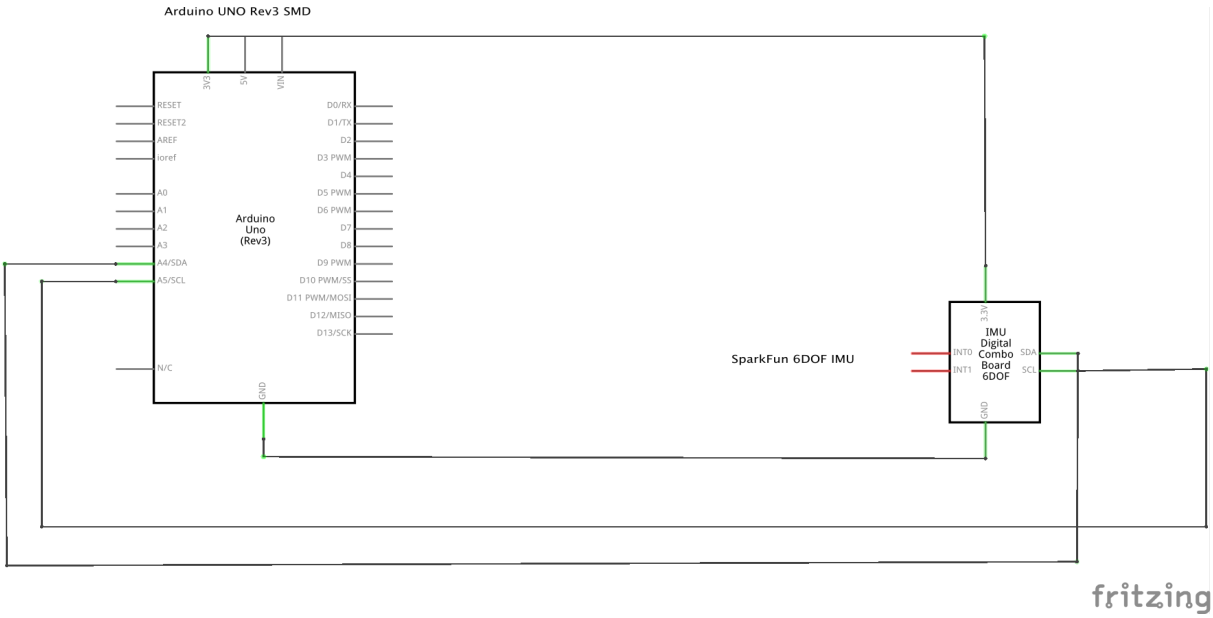


Figure 4: Schematic of the Inertial Measurement Unit (IMU).

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

References

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

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

[3] 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.

[4] 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.

[5] D. N. O. N. J. Bergen, "Visual odometry," in *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, vol. 1, pp. 652–659, 2004.

[6] T. Lemaire, C. Berger, I. kyun Jung, and S. Lacroix, "Vision-based slam: Stereo and monocular approaches," tech. rep., Int. J. Compt. Vision, 2006.

[7] 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.

[8] 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.

[9] S. Holmes, G. Klein, and D. W. Murray, "A square root unscented kalman filter for visual monoslam," in *in Proc. of the IEEE International Conference on Robotics and Automation*, pp. 3710–3716, 2008.

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

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

[12] D. Valiente, A. Gil, L. Fernández, and Reinoso, "A comparison of ekf and sgd applied to a view-based slam approach with omnidirectional images," *Robot. Auton. Syst.*, vol. 62, pp. 108–119, February 2014.

[13] P. Yang, "Efficient particle filter algorithm for ultrasonic sensor-based 2d range-only simultaneous localisation and mapping application," *Wireless Sensor Systems, IET*, vol. 2, no. 4, pp. 394–401, 2012.

[14] H. Alismail, L. D. Baker, and B. Browning, "Continuous trajectory estimation for 3d slam from actuated lidar," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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