Scripsie: State Estimation & Observation

Aidan Landsberg

April 2, 2015

Contents

List of Figures

List of Tables

1 Single Camera Simultaneous Localisation and Mapping

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 positions of every feature observed. These 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 process regarding the construction of this map of features 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 predictions and measurements 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 a sparse set of higher-quality landmarks rather than a dense set of ordinary landmarks within the environment.
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 frame 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 dot 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 scalars. An example can be shown regarding the variable x: $\mathbf{x}$ denotes a vector while its scalar counterpart would be represented as x.

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.

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. The table below presents a pseudo-algorithmic interpretation of the Bayes Filter algorithm and is based on the definition as in Probabilistic Robotics...:

The recursive nature of the algorithm can thus be seen from table 2.1 above; 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

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.

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 through these sources as well as the appendix (Probabilistic Robotics, Cyril Stachnis etc.).
This summary is taken from Probabilistic Robotics... Remember to reference.

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:

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

where the density across the state variable x_t is characterised through two fundamental parameters: The mean μ and the covariance Σ. 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:

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

2. The observation model probability - with function $h(\bar{\mu}_t)$ - **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 **optimally** estimates the state of the robots pose as well as 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 (μ_{t-1}, Σ_{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 $g(\mu_t, \mu_{t-1})$. Once an estimate is obtained for the transitioned state estimate ($\bar{\mu}_t$, $\bar{\Sigma}_t$), 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 $h(\bar{\mu}_t)$. Thereafter, an actual measurement, $\mathbf{z}_t$ is then obtained through the system sensors in order to determine the actual state of the system (μ_t, Σ_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 an 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. Table 2.2 also incorporates a systematical and mathematical representation for the following system definition of the state transition and the observation models respectively:

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

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.	$\bar{\mu}_t = g(\mathbf{u}_t, \mu_{t-1}) = \mathbf{A}_t \mu_{t-1} + \mathbf{B}_t \mu_t + \mathbf{w}_t$
2.	$\bar{\Sigma}_t = \mathbf{A}_t \Sigma_{t-1} \mathbf{A}_t^T + R_{w,t}$
	Correction step
3.	$K_t = \bar{\Sigma}_t \mathbf{C}_t^T (\mathbf{C}_t \bar{\Sigma}_t \mathbf{C}_t^T + Q_{v,t})^{-1}$
4.	$\mu_t = \bar{\mu}_t + K_t[\mathbf{z}_t - \mathbf{C}_t \bar{\mu}_t]$
5.	$\Sigma_t = (I - K_t \mathbf{C}_t)\bar{\Sigma}_t$

Insert the 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 to many 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:

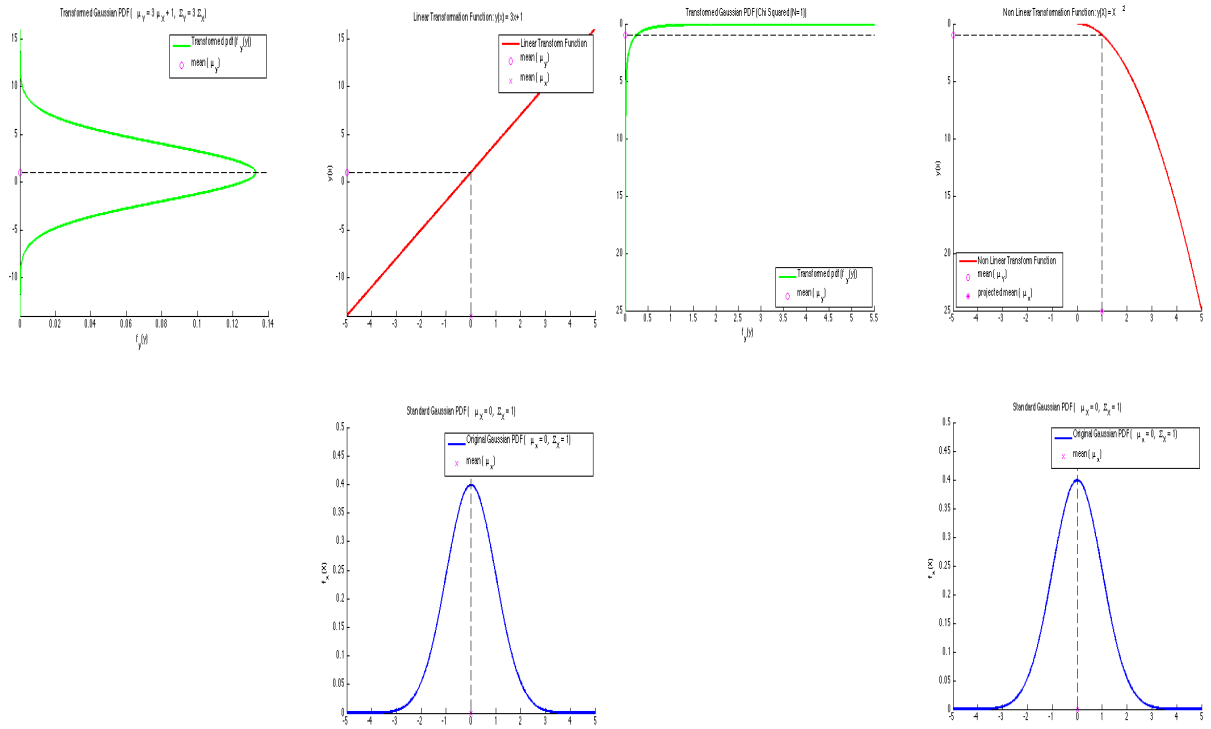

Figure 1: Graphical representation depicting both a linear (left) and a non linear (right) transformation of a Gaussian random variable.

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 $g(\mu_t, \mu_{t-1})$, 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. The linearisation process of EKF aims to linearise these functions so that the basic fundamental operations of the Kalman Filter algorithm can be validated.

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. If the Gaussian is then projected through this new linear approximation, the resultant transformation would yield a random variable that is Gaussian in nature. 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}:

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

where $\mathbf{F}_t = f'(\mathbf{u}_t, \mu_{t-1})$ 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. Other variants of the 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.

Table 2.3: The Extended 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.	$\bar{\mu}_t = g(\mathbf{u}_t, \mu_{t-1})$
2.	$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_{w,t}$
	Correction step
3.	$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_{v,t})^{-1}$
4.	$\mu_t = \bar{\mu}_t + K_t[z_t - h(\bar{\mu}_t)]$
5.	$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$

3 EKF MonoSLAM: EKF Monocular 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 positions of every feature observed. This scenario is typically defined as the SLAM problem. Such a problem, is typically solved through the utilisation of the EKF. The goal of a particular SLAM algorithm termed MonoSLAM (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 2003]. Various **successful** SLAM algorithms exist that utilise sensors other than cameras (laser range finders, ultrasonic sensors etc.). Cameras though, prove a better economical alternative to these sensors. Another successful and popular implementation is stereo vision (two calibrated cameras), yet the obvious disadvantage regarding such an approach is that double the cost is required as opposed to a single camera approach.
It can be argued though, that the general approach presented by [Davison 2003] as well as variants thereof [Other sources] can be improved through the utilisation of additional information regarding the motion of the robot. 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. Inertial measurement unit's have been implemented in SLAM based systems before, as implemented in [Other sources]. 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 *observed* accordingly. 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.
Furthermore, the remaining components of the proposed approach presented in this paper are identical to that of the general MonoSLAM algorithm of Davison: that is, using the EKF to successfully obtain the desired probabilistic map.

This section will initially denote the adequate state representation of the system at hand, 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 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.

We seek to explain the main idea behind EKF

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 $\mathbf{y}$ 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:

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

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 cameras position and orientation in a 3D space. According to most implementations of robot localisation, there exists no concern to 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. 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* - to be mathematically defined and described in the appendix - 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.
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 *quarternion*, $\mathbf{q}^{WC}$. The linear velocity vector, $\mathbf{V}^W$, forms the additional information required for system modelling. We want to say here that the control inputs are of such a nature that intermediary states (such as linear velocities) are required to describe the

control inputs effect on the 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 $\mathbf{y}$ - comprising of N landmarks - is mathematically described through three individual cartesian coordinates - x, y and z respectively:

$$\mathbf{y}_n = (x_n \;\; y_n \;\; z_n)^T, \tag{3.3}$$

where n corresponds to a specific, single landmark.

3.1.3 Control Inputs

In most instances of robotics, it is essential to describe the dynamics involving a robot's movement. In the context of the specific implementation discussed in this paper, the 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, 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), the model contains very little, if any information on the movement of the camera. The approach proposed in this paper, aims to utilise inertial sensors in order to obtain the necessary information regarding the camera's movement.
The inertial sensors, in the form of an inertial measurement unit (IMU), should ideally be mounted onto the camera. This will allow 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 ω_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:

$$\mathbf{f}_t = \mathbf{R}(\mathbf{a}_t - \mathbf{g}_t) \tag{3.4}$$

where $\mathbf{R}$ is the rotation matrix that transforms the body coordinate frame data into the inertial reference frame, $\mathbf{a}_t$ is the linear acceleration vector and $\mathbf{g}_t$ is the gravity vector.
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:

$$\mathbf{u}_t = \begin{pmatrix} \mathbf{a}_t \\ \omega_t \end{pmatrix} = \begin{bmatrix} \ddot{x}_t & \ddot{y}_t & \ddot{z}_t & \dot{q}_{0,t} & \dot{q}_{1,t} & \dot{q}_{2,t} & \dot{q}_{3,t} \end{bmatrix}^T \tag{3.5}$$

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:

$$\mathbf{w}_t = \begin{pmatrix} \mathbf{n}_{\mathbf{a},t} \\ \mathbf{n}_{\omega,t} \end{pmatrix} = \begin{bmatrix} n_{\ddot{x}_k} & n_{\ddot{y}_k} & n_{\ddot{z}_k} & n_{\dot{q}_{0,k}} & n_{\dot{q}_{1,k}} & n_{\dot{q}_{2,k}} & n_{\dot{q}_{3,k}} \end{bmatrix}^T \tag{3.6}$$

where the aforementioned noise model yielding each of the above elements a Gaussian random variable.
Furthermore, the resultant IMU data to be used are to contain the measurements of the linear accelerations and angular rotations as well as the appropriate additive noise.

Reconsider how to incorporate this section elsewhere or define it properly.

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:

$$\begin{aligned} bel(\mathbf{x}_t) &= p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) \\ &= \frac{1}{\sqrt{|2\pi\mathbf{R}_w|}} \exp\left\{ \frac{1}{2} \big[\mathbf{x}_t - g(\mu_t,\ \mu_{t-1}) - G_t(\mathbf{x}_{t-1} - \mu_{t-1})\big]^T \right. \\ &\left. \mathbf{R}_w^{-1} \big[\mathbf{x}_t - g(\mu_t,\ \mu_{t-1}) - G_t(\mathbf{x}_{t-1} - \mu_{t-1})\big] \right\}, \end{aligned} \tag{3.7}$$

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 State Transition 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 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 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, non linear discrete-time model is adopted to approximate the model. We begin by describing all relevant states and control inputs:

$$\begin{aligned}
\mathbf{x}_t &= \begin{bmatrix} \mathbf{r}_t^W & \mathbf{q}_t^{WC} & \mathbf{v}_t^W \end{bmatrix}^T \\
&= \begin{bmatrix} x_t & y_t & z_t & \dot{x}_t & \dot{y}_t & \dot{z}_t & q_{0,t} & q_{1,t} & q_{2,t} & q_{3,t} \end{bmatrix}^T \\
\mathbf{u}_t &= \begin{bmatrix} \mathbf{a}_t^C & \dot{\omega}_t^C \end{bmatrix}^T \\
&= \begin{bmatrix} \ddot{x}_t & \ddot{y}_t & \ddot{z}_t & \dot{q}_{0,t} & \dot{q}_{1,t} & \dot{q}_{2,t} & \dot{q}_{3,t} \end{bmatrix}^T
\end{aligned} \tag{3.8}$$

The state transition function $g_v(\mathbf{u}_t, \mu_{t-1})$ is defined as follows:

$$\begin{aligned}
g_v(\mathbf{u}_t, \mu_{t-1}) = \begin{pmatrix} \mathbf{r}_t^W \\ \mathbf{q}_t^{WC} \\ \mathbf{v}_t^W \end{pmatrix} &= \begin{pmatrix} \mathbf{r}_{t-1}^W + \mathbf{v}_t^W \Delta T \\ \mathbf{q}_{t-1}^{WC} \otimes \mathrm{quat}\big(\dot{\omega}_t^C \Delta T\big) \\ \mathbf{v}_t^W + \mathbf{a}_t^C \Delta T \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{r}_{t-1}^W + \dot{\mathbf{r}}^W \Delta T \\ \mathbf{q}_{t-1}^{WC} \otimes \mathrm{quat}\big(\dot{\mathbf{q}}_t^C \Delta T\big) \\ \dot{\mathbf{r}}_t^W + \mathbf{R}_t^{CW}\big(\ddot{\mathbf{r}}_t^C \Delta T\big) \end{pmatrix}
\end{aligned} \tag{3.9}$$

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 probability distribution. The observation model however, models the uncertainty regarding a measurement taken at an instance $\mathbf{z}_t$ given that the locations of both the robot as well as the landmarks are known. This uncertainty can be described in the following form:

$$P(\mathbf{z}_t \mid \mathbf{x}_t) = \frac{1}{\sqrt{|2\pi\mathbf{Q}_t|}} \exp\left\{\frac{1}{2}\big[\mathbf{z}_t - h(\bar{\mu}) - H_t(\mathbf{x}_t - \bar{\mu}_t)\big]^T \mathbf{Q}_t^{-1}\big[\mathbf{z}_t - h(\bar{\mu}) - H_t(\mathbf{x}_t - \bar{\mu}_t)\big]\right\}. \tag{3.10}$$

where 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 $\hat{\mathbf{z}}_k$ 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 $\mathbf{h}_i$.

3.3.1 Measurement Function

The correction step of the Extended Kalman filter aims to ultimately correct the previously estimated robot pose and landmark position through exterior sensor measurements. The measurement process generally involves a measurement estimate that incorporates an uncertainty. With regard to the implantation proposed in this paper, landmarks are required to be observed and measured through the use of a camera. To mathematically describe this process, the previously mentioned measurement function is used to effectively model the measurement estimation. It is essential that the measurement function, like the motion model, be **linear** in nature and additionally, the measurement function must describe the position of a **point** feature with regard to the previously estimated states - namely the robot pose and the landmark positions.

Considering that the camera observations are obtained with regard to its own reference frame C, the definition of the measurement function is adapted in order to be described with regard to the inertial reference frame. The measurement function $\mathbf{h}_i^W$ that describes a directional vector in relation to the cameras body frame is thus mathematically defined as follows:

$$\mathbf{h}_i^W = \mathbf{R}^{CW}\big(\mathbf{y}_i^W - \mathbf{r}^W\big) = \left(\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} - \mathbf{r}^W\right) \tag{3.11}$$

where the subscript i corresponds a directional vector $\mathbf{h}^C$ to its cartesian point $\mathbf{y}^W$, $\mathbf{r}^W$ describes the cartesian position of the camera, $\mathbf{y}^W$ describes the cartesian position of a given landmark and $\mathbf{R}^{CW}$ represents the rotational matrix that is required to transform the aforementioned positional vectors from the inertial reference frame into the cameras body frame coordinate system.

With reference to the section regarding perspective cameras, it can be recalled that a given features position is described by a 2-dimensional position of the image frame of the camera. Recalling, the standard pinhole camera model defines this position mathematically as follows:

$$\mathbf{h}_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} u_0 - fk_u \frac{h^R_{i,x}}{h^R_{i,z}} \\ v_0 - fk_v \frac{h^R_{i,y}}{h^R_{i,z}} \end{pmatrix} \tag{3.12}$$

where fk_u, fk_v, u_0 and v_0 are the previously described camera calibration parameters.

3.3.2 Feature Tracking

3.3.3 System Update

A Linear State Space Model

A.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:

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

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:

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

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):

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

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:

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

A.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:

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

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

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

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

it can be observed from the model above that the motion model adheres to the forward method of discretisation derived in (2.8). 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.