IIIT Hyderabad **Group Number: 8**
- **Shaantanu Kulkarni (2019102031)**
- **Shubransh Singhvi(2019102025)**
- **Charchit Gupta (2019102034)**
- **Aditya Sinha (2019102003)**
- **Ashirwad Sinha (2019102035)**

Probability And Random Processes (MA6.102) **Assignment-1**
Instructor: Dr. Lalitha Vadlamani

Probability In Robotics

1 Introduction

At the core of probabilistic robotics is the idea of estimating state from sensor data. State estimation addresses the problem of estimating quantities from sensor data that are not directly observable, but that can be inferred. Probabilistic state estimation algorithms compute belief distributions over possible world states.

1.1 Background Knowledge

In probabilistic robotics, quantities such as sensor measurements, controls, and the states a robot and its environment might assume are all modeled as random variables. **Gaussian Function :**

$$p(x) = 2\pi\sigma^2{}^{\frac{-1}{2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{1}$$

This can be abbreviated as $\mathcal{N}(x; \mu, \sigma^2)$, which specifies the random variable, its mean, and its variance. The Normal distribution assumes that x is a scalar value. Often, x will be a multi-dimensional vector. Normal distributions over vectors are called multivariate. Multivariate normal distributions are characterized by density functions of the following form.

$$p(x) = det(2\pi \sum)^{\frac{-1}{2}} e^{\frac{-1}{2}(x-\mu)^T \sum^{-1}(x-\mu)} \tag{2}$$

Here μ is the mean vector and $\sum$ a positive semidefinite symmetric matrix called covariance matrix.

Theorem of total probability

$$p(x) = \sum_y p(x|y)p(y) \qquad (discrete\ \ case) \tag{3}$$

$$p(x) = \int p(x|y)p(y)dy \qquad (continuous\ \ case) \tag{4}$$

Bayes Rule :

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\sum_{x'} p(y|x')p(x')} \qquad (discrete\ \ case) \tag{5}$$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\int p(y|x')p(x')dx'} \qquad (discrete\ \ case) \tag{6}$$

Bayes rule plays a predominant role in probabilistic robotics. If x is a quantity that we would like to infer from y, the probability p(x) will be referred to as prior probability distribution, and y is called the data (e.g., a sensor measurement). The distribution p(x) summarizes the knowledge we have regarding X prior to incorporating the data y. The probability p(x | y) is called the posterior probability distribution over X. Bayes rule provides a convenient way to compute a posterior p(x | y) using the "inverse" conditional probability p(y | x) along with the prior probability p(x).

In other words, if we are interested in inferring a quantity x from sensor data y, Bayes rule allows us to do so through the inverse probability, which specifies the probability of data y assuming that x was the case. In robotics, this inverse probability is often coined "generative model," since it describes, at some level of abstraction, how state variables X cause sensor measurements Y .

An important observation is that the denominator of Bayes rule, p(y), does not depend on x. Thus, the factor $p(y)^{-1}$ For this reason, $p(y)^{-1}$ is often written as a normalizer variable, and generically denoted η:

$$p(x|y) = \eta\, p(y|x)p(x) \tag{7}$$

Instead of explicitly providing the exact formula for a normalization constant—which can grow large very quickly in some of the mathematical derivations to follow. We have used the normalizer η to indicate that the final result has to be normalized to 1.
It is perfectly fine to condition any of the rules discussed so far on arbitrary other random variables, such as the variable Z. For example, conditioning Bayes rule on Z = z gives us:

$$p(x|y,z) = \frac{p(y|x,z)p(x|z)}{p(y|z)} \tag{8}$$

Conditional indpendence :

$$p(x,y|z) = p(x|z)p(y|z) \tag{9}$$

Conditional independence plays an important role in probabilistic robotics. It applies whenever a variable y carries no information about a variable x if another variable's value z is known.

2 ROBOT ENVIRONMENT INTERACTION

The environment, or world, of a robot is a dynamical system that possesses internal state. The robot can acquire information about its environment using its sensors. However, sensors are noisy, and there are usually many things that cannot be sensed directly. As a consequence, the robot maintains an internal belief with regards to the state of its environment, depicted on the left in the figure below.

2.1 State

Environments are characterized by state.It will be convenient to think of state as the collection of all aspects of the robot and its environment that can impact the future.A state may be denoted by x , the state at time t will be denoted by x_t.Some typical state variables are:

- The robot pose, which comprises its location and orientation relative to a global coordinate frame.It is often referred to as *kinematicstate*.

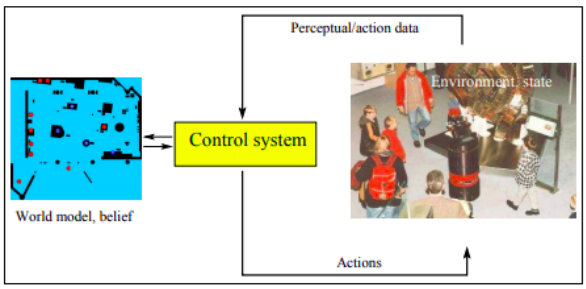

Figure 1: Robot Environment Interaction

- The configuration of the robot's actuators, such as the joints of robotic manipulators.
- The robot velocity and the velocities of its joints.Velocities are commonly referred to as *dynamicstate*.
- The location and features of surrounding objects in the environment. Depending on the granularity of the state that is being modeled, robot environments possess between a few dozen and up to hundreds of billions of state variables.
- The location and velocities of moving objects and people
- There can be a huge number of other state variables. For example, whether or not a sensor is broken is a state variable, as is the level of battery charge for a battery-powered robot.

A state x_t will be called complete if it is the best predictor of the future. Put differently, completeness entails that knowledge of past states, measurements, or controls carry no additional information that would help us to predict the future more accurately.

2.2 Environment interaction

There are two fundamental types of interactions between a robot and its environment: The robot can influence the state of its environment through its actuators. And it can gather information about the state through its sensors.

- **Sensor measurements** Perception is the process by which the robot uses its sensors to obtain information about the state of its environment.
- **Control actions** change the state of the world. They do so by actively asserting forces on the robot's environment.

Hypothetically, a robot may keep a record of all past sensor measurements and control actions. We will refer to such a the collection as the data . In accordance with the two types of environment interactions, the robot has access to two different data streams.

- Measurement data provides information about a momentary state of the environment. Examples of measurement data include camera images, range scans, and so on
  The measurement data at time t will be denoted z_t The notation

$$z_{t1:t2} = z_{t1} \ , \ z_{t1+1} \ , \ z_{t1+2} \cdot \cdot \cdot \cdot \ , \ z_{t2}$$

  denotes the set of all measurements acquired from time t_1 to time t_2, for $t_1 \leq t_2$.

- Control data carry information about the change of state in the environment. In mobile robotics, a typical example of control data is the velocity of a robot.
  Control data will be denoted u_t. The variable u_t will always correspond to the change of state in the time interval $(t-1;t]$. As before, we will denote sequences of control data by $z_{t1:t2}$, for $t_1 \leq t_2$

$$u_{t1:t2} = u_{t1} \ , \ u_{t1+1} \ , \ u_{t1+2}. \ldots , \ u_{t2}$$

2.3 Probabilistic Generative Laws

The evolution of state and measurements is governed by probabilistic laws. In general, the state at time x_t is generated stochastically. Thus, it makes sense to specify the probability distribution from which x_t is generated. At first glance, the emergence of state x_t might be conditioned on all past states, measurements, and controls. Hence, the probabilistic law characterizing the evolution of state might be given by a probability distribution of the following form:

$$p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t}) \tag{10}$$

(Notice that through no particular motivation we assume here that the robot executes a control action u_1 first, and then takes a measurement z_1.) However, if the state x is complete then it is a sufficient summary of all that happened in previous time steps. From all the variables in the expression above, only the control u_t matters if we know the state x_{t-1} . In probabilistic terms,this insight is expressed by the following equality:

$$p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t) \tag{11}$$

The property expressed by this equality is an example of *conditionalindependence*
Similarly, one might want to model the process by which measurements are being generated. Again, if x t is complete, we have an important conditional independence:

$$p(z_t|x_{0:t}, z_{1:(t-1),u_{1:t}}) = p(z_t|x_t) \tag{12}$$

The probability $p(x_t|x_{t-1}, u_t)$ is the state transition probability. It specifies how environmental state evolves over time as a function of robot controls u_t. Robot environments are stochastic, which is reflected by the fact that $p(x_t|x_{t-1}, u_t)$ is a probability distribution, not a deterministic function.

2.4 Belief Distributions

A belief reflects the robot's internal knowledge about the state of the environment. For example, a robot's pose might be x = (14.12, 12.7, 0.755) in some global coordinate system, but it usually cannot know its pose, since poses are not measurable directly . Instead, the robot must infer its pose from data. We therefore distinguish the true state from its internal belief, or state of knowledge with regards to that state.

Probabilistic robotics represents beliefs through conditional probability distributions. A belief distribution assigns a probability to each possible hypothesis with regards to the true state. Belief distributions are posterior probabilities over state variables conditioned on the available data. We will denote belief over a state variable x_t by bel(x_t).

$$bel(x_t) = p(x_t|z_{1:t}, u_{1:t}) \tag{13}$$

We assume that the belief is taken after incorporating the measurement z_t . a posterior before incorporating z_t, just after executing the control u_t.

$$\overline{bel}(x_t) = p(x_t|z_{1:t-1}, u_{1:t}) \tag{14}$$

This probability distribution is often referred to as prediction in the context of probabilistic filtering. This terminology reflects the fact that bel(x_t) predicts the state at time t based on the previous state posterior, before incorporating the measurement at time t. Calculating bel(x_t) from $\overline{bel}(x_t)$ is called correction or the measurement update.

3 BAYES FILTERS

3.1 The Bayes Filter Algorithm

The most general algorithm for calculating beliefs is given by the Bayes filter algorithm. This algorithm calculates the belief distribution bel from measurement andS control data.

1: **Algorithm Bayes_filter**($bel(x_{t-1}), u_t, z_t$):
2: for all x_t do
3: $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\ dx$
4: $bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t)$
5: endfor
6: return $bel(x_t)$

Figure 2: The general algorithm for Bayes filtering

The Bayes filter is recursive, that is, the belief $bel(x_t)$ at time t is calculated from the belief $bel(x_{t-1})$ at time $t-1$. Its input is the belief bel at time $t-1$, along with the most recent control u_t and the most recent measurement z_t. Its output is the belief bel(x_t) at time t.

The Bayes filter algorithm possesses two essential steps. In Line 3, it processes the control u_t. It does so by calculating a belief over the state xt based on the prior belief over state x_{t-1} and the control ut.

The second step of the Bayes filter is called the measurement update. In Line 4, the Bayes filter algorithm multiplies the belief $\overline{bel}(x_t)$ by the probability that the measurement z_t may have been observed
To compute the posterior belief recursively, the algorithm requires an initial belief $bel(x_0)$ at time $t = 0$ as boundary condition.

Example 3.1. *Consider a robot estimating the state of a door ass shown in figure . For simplicity, we assume that the door can only be in two possible states open or closed, and that only robot can change the state of the door. Assuming that the robot doesn't know the state of the door initially :*

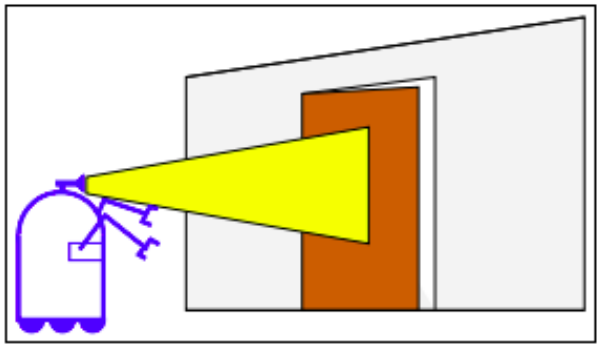

Figure 3: A mobile robot estimating the state of a door

$$bel(X_0 = \textbf{\textit{open}}) = 0.5$$
$$bel(X_0 = \textbf{\textit{closed}}) = 0.5$$

Consider that the robot's sensors are noisy , and that the noise can be modeled as follows:

$$p(Z_t = \textbf{sense\_ open}|X_t = \textbf{is\_ open}) = 0.6$$

$$p(Z_t = \textbf{sense\_ closed}|X_t = \textbf{is\_ open}) = 0.4$$

$$p(Z_t = \textbf{sense\_ open}|X_t = \textbf{is\_ closed}) = 0.2$$

$$p(Z_t = \textbf{sense\_ closed}|X_t = \textbf{is\_ closed}) = 0.8$$

if the robot can choose to use its manipulator to open the door:

$$p(X_t = \textbf{is\_ open}|U_t = \textbf{push}, X_{t-1} = \textbf{is\_ open}) = 1$$

$$p(X_t = \textbf{is\_ closed}|U_t = \textbf{push}, X_{t-1} = \textbf{is\_ open}) = 0$$

$$p(X_t = \textbf{is\_ open}|U_t = \textbf{push}, X_{t-1} = \textbf{is\_ closed}) = 0.8$$

$$p(X_t = \textbf{is\_ closed}|U_t = \textbf{push}, X_{t-1} = \textbf{is\_ closed}) = 0.2$$

otherwise,i.e., if the robot chooses to do nothing then:

$$p(X_t = \textbf{is\_ open}|U_t = \textbf{do\_ nothing}, X_{t-1} = \textbf{is\_ open}) = 1$$

$$p(X_t = \textbf{is\_ closed}|U_t = \textbf{do\_ nothing}, X_{t-1} = \textbf{is\_ open}) = 0$$

$$p(X_t = \textbf{is\_ open}|U_t = \textbf{do\_ nothing}, X_{t-1} = \textbf{is\_ closed}) = 0$$

$$p(X_t = \textbf{is\_ closed}|U_t = \textbf{do\_ nothing}, X_{t-1} = \textbf{is\_ closed}) = 1$$

Suppose at time t, the robot takes no control action but it senses an open door , the resulting posterior belief is calculated by the Bayes filter using the prior belief bel(X_0), the control u_1 = do nothing,

$$\overline{bel}(x_1) = \int p(x_1|u_1, x_0)bel(x_0)dx_0$$

$$= \sum_{x_0} p(x_1|u_1, x_0)bel(x_0)$$

$$= p(x_1|U_1 = do\ nothing\ ,\ X_0 = is\ \ open)\ \ bel(X_0 = is\ \ open)$$

For the hypothesis X_1 = is open

$$\overline{bel}(X_1 = is_open) = p(X_1|U_1 = do - nothing\ ,\ X_0 = is - open)bel(X_0 = is - open)$$

$$+p(X_1|U_1 = do - nothing\ ,\ X_0 = is - closed)bel(X_0 = is - closed)$$

$$= 0.(0.5) + 1.(0.5) = 0.5$$

For the hypothesis X_1 = is closed

$$\overline{bel}(X_1 = is_closed) = p(X_1|U_1 = do - nothing\ ,\ X_0 = is - closed)bel(X_0 = is - open)$$

$$+p(X_1|U_1 = do - nothing\ ,\ X_0 = is - closed)bel(X_0 = is - closed)$$

$$= 0.(0.5) + 1.(0.5) = 0.5$$

The fact that the belief bel(x_1) equals our prior belief bel(x_0) should not surprise, as the action do nothing does not affect the state of the world.

$$bel(x_1) = \eta p(Z_1 = sense - open|x_1)\overline{bel}(x_1) \tag{15}$$

$$bel(X_1 = \textbf{is\_open}) = \eta p(Z_1 = \textbf{sense\_open}|X_1 = is\_open)\overline{bel}(X_1 = \textbf{is\_open})$$

$$= \eta 0.60.5 = 0.3\eta$$

$$bel(X_1 = \textbf{is\_closed}) = \eta p(Z_1 = \textbf{sense\_open}|X_1 = is\_closed)\overline{bel}(X_1 = \textbf{is\_closed})$$

$$= \eta 0.20.5 = 0.1\eta$$

Now η can be calculated as follows:

$$0.3\eta + 0.1\eta = 1$$

$$\eta = 2.5$$

$$bel(X_1 = is - open) = 0.75$$

$$bel(X_1 = is - closed) = 0.25$$

This calculation is now easily iterated for the next time step.

$$\overline{bel}(X_2 = \textbf{is\_open}) = 1.(0.75) + 0.8.(0.25) = 0.95$$

$$\overline{bel}(X_2 = \textbf{is\_closed}) = 1.(0.75) + 0.2.(0.25) = 0.05$$

$$bel(X_2 = \textbf{is\_open}) = \eta(0.6)(0.95) \approx 0.983$$

$$bel(X_2 = \textbf{is\_closed}) = \eta(0.2)(0.05) \approx 0.017$$

4 Gaussaian Filter

Historically, Gaussian filters constitute the earliest tractable implementations of the Bayes filter for continuous spaces. They are also by far the most popular family of techniques to date—despite a number of shortcomings. Gaussian techniques all share the basic idea that beliefs are represented by multivariate normal distributions. The definition of the multivariate normal distribution can be stated as following:

$$p(x) = \frac{1}{\sqrt{|2\pi\Sigma|}} e^{-1/2(x-\mu)^T \Sigma^{-1}(x-\mu)} \tag{16}$$

This density over the variable x is characterized by two sets of parameters: The mean μ and the covariance Σ. The mean μ is a vector that possesses the same dimensionality as the state x. The covariance is a quadratic matrix that is symmetric and positivesemidefinite. Its dimension is the dimensionality of the state x squared. Thus, the number of elements in the covariance matrix depends quadratically on the number of elements in the state vector.

The commitment to represent the posterior by a Gaussian has important ramifications. Most importantly, Gaussians are unimodal, that is, they posses a single maximum. Such a posterior is characteristic of many tracking problems in robotics, in which the posterior is focused around the true state with a small margin of uncertainty. Gaussian posteriors are a poor match for many global estimation problems in which many distinct hypotheses exist, each of which forming its own mode in the posterior. The representation of a Gaussian by its mean and covariance is called the moments representation. This is because the mean and covariance are the first and second moments of a probability distribution; all other moments are zero for normal distributions.

5 THE KALMAN FILTER

5.0.1 Linear Gaussian Systems

The Kalman filter represents beliefs by the moments representation: At time t, the belief is represented by the the mean µt and the covariance t. Posteriors are Gaussian if the following three properties hold, in addition to the Markov assumptions of the Bayes filter.

1. The next state probability $p(x_t|u_t, x_{t-1})$ must be a linear function in its arguments with added Gaussian noise. This is expressed by the following equation:

$$x_t = A_t x_t + B_t \mu_t + \varepsilon_t \tag{17}$$

Here x_t and x_{t-1} are state vectors, and u_t is the control vector at time t. In our notation, both of these vectors are vertical vectors, that is, they are of the form

$$x_t = A_t x_t + B_t \mu_t + \varepsilon_t \tag{18}$$

Here x_t and $xt-1$ are state vectors, and u_t is the control vector at time t. In our notation, both of these vectors are vertical vectors, that is, they are of the form

$$x_t = \begin{pmatrix} x_{1,t} \\ x_{2,t} \\ . \\ . \\ . \\ x_{n,t} \end{pmatrix} \quad and \quad u_t = \begin{pmatrix} u_{1,t} \\ u_{2,t} \\ . \\ . \\ . \\ u_{n,t} \end{pmatrix} \tag{19}$$

A_t and B_t are matrices. A_t is a square matrix of size n × n, where n is the dimension of the state vector x_t. B_t is of size n × m, with m being the dimension of the control vector u_t. By multiplying the state and control vector with the matrices A_t and B_t, respectively, the state transition function becomes linear in its arguments. Thus, Kalman filters assume linear system dynamics.

The random variable ε_t is a Gaussian random vector that models the randomness in the state transition. It is of the same dimension as the state vector. Its mean is zero and its covariance will be denoted R_t. A state transition probability is called a linear Gaussian, to reflect the fact that it is linear in its arguments with additive Gaussian noise.

From the previous equation we know that our the state transition probability is $p(x_t|u_t, x_{t-1})$. This probability is obtained by plugging Equation (3.2) into the definition of the multivariate normal distribution (3.1). The mean of the posterior state is given by $A_t x_{t-1}$ + $B_t u_t$ and the covariance by R_t:

$$p(x_t|u_t, x_{t-1}) = \frac{1}{\sqrt{|2\pi R_t|}} e^{-1/2(x_t - A_t x_{t-1} - B_t \mu_t)^T R_t^{-1} (x_t - A_t x_{t-1} - B_t \mu_t)} \tag{20}$$

2. The measurement probability $p(z_t|x_t)$ must also be linear in its arguments, with added Gaussian noise:

$$z_t = C_t x_t + \delta_t \tag{21}$$

Here C_t is a matrix of size k × n, where k is the dimension of the measurement vector z_t. The vector δ_t describes the measurement noise. The distribution of δ_t is a multivariate Gaussian with zero mean

and covariance Q_t. The measurement probability is thus given by the following multivariate normal distribution:

$$p(z_t|x_t) = \frac{1}{\sqrt{|2\pi Q_t|}} e^{-1/2(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t)} \tag{22}$$

1: **Algorithm Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2: $\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$
3: $\bar{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t$
4: $K_t = \bar{\Sigma}_t\ C_t^T (C_t\ \bar{\Sigma}_t\ C_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\ \bar{\mu}_t)$
6: $\Sigma_t = (I - K_t\ C_t)\ \bar{\Sigma}_t$
7: return μ_t, Σ_t

Figure 4: The Kalman filter (KF) algorithm.

3. Finally, the initial belief bel(x_0) must be normal distributed. We will denote the mean of this belief by μ_0 and the covariance by Σ_0:

$$bel(x_0) = p(x_0) = \frac{1}{\sqrt{|2\pi\Sigma_0|}} e^{-1/2(x_0 - \mu_0)^T \Sigma_0^{-1}(x_0 - \mu_0)} \tag{23}$$

These three assumptions are sufficient to ensure that the posterior bel($_t$) is always a Gaussian, for any point in time t. The proof of this non-trivial result is also included in this report as the mathematical derivation of the Kalman filter.

5.0.2 The Kalman Filter Algorithm

The Kalman filter algorithm is mentioned above. Kalman filters represent the belief bel(x_t) at time t by the mean μ_t and the covariance Σ_t. The input of the Kalman filter is the belief at time t 1, represented by μ_{t-1} and Σ_{t1}. To update these parameters, Kalman filters require the control μ_t and the measurement z_t. The output is the belief at time t, represented by μ_t and Σ_t.

In Lines 2 and 3, the predicted belief $\overline{\mu}$ and $\overline{\Sigma}$ is calculated representing the belief $\overline{bel(x_t)}$ one time step later, but before incorporating the measurement z_t. This belief is obtained by incorporating the control μ_t. The mean is updated using the deterministic version of the state transition function , with the mean μ_{t1} substituted for the state x_{t1}. The update of the covariance considers the fact that states depend on previous states through the linear matrix A_t. This matrix is multiplied twice into the covariance, since the covariance is a quadratic matrix.

The belief bel(x_t) is subsequently transformed into the desired belief bel(x_t) in Lines 4 through 6, by incorporating the measurement zt. The variable K_t, computed in Line 4 is called Kalman gain. It specifies the degree to which the measurement is incorporated into the new state estimate. Line 5 manipulates the mean, by adjusting it in proportion to the Kalman gain K_t and the deviation of the actual measurement, z_t, and the measurement predicted according to the measurement probability. Finally, the new covariance of the posterior belief is calculated in Line 6, adjusting for the information gain resulting from the measurement.

The Kalman filter is computationally quite efficient. For today's best algorithms, the complexity of matrix inversion is approximately O($d^{2.8}$) for a matrix of size d × d. Each iteration of the Kalman filter algorithm, as stated here, is lower bounded by (approximately) O($k^{2.8}$), where k is the dimension of the

measurement vector z_t. This (approximate) cubic complexity stems from the matrix inversion in Line 4. It is also at least in $O(n^2)$, where n is the dimension of the state space, due to the multiplication in Line 6 (the matrix $K_t C_t$ may be sparse). In many applications—such as the robot mapping applications discussed in later chapters—-the measurement space is much lower dimensional than the state space, and the update is dominated by the $O(n^2)$ operations.

6 THE EXTENDED KALMAN FILTER

The assumptions of linear state transitions and linear measurements with added Gaussian noise are rarely fulfilled in practice. For example, a robot that moves with constant translational and rotational velocity typically moves on a circular trajectory, which cannot be described by linear next state transitions. This observation, along with the assumption of unimodal beliefs, renders plain Kalman filters, as discussed so far, inapplicable to all but the most trivial robotics problems.

The extended Kalman filter (EKF) overcomes one of these assumptions: the linearity assumption. Here the assumption is that the next state probability and the measurement probabilities are governed by nonlinear functions g and h, respectively:

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t \tag{24}$$

$$z_t = h(x_t) + \delta_t \tag{25}$$

This model strictly generalizes the linear Gaussian model underlying Kalman filters. The function g replaces the matrices A_t and B_t, and h replaces the matrix C_t. Unfortunately, with arbitrary functions g and h, the belief is no longer a Gaussian. In fact, performing the belief update exactly is usually impossible for nonlinear functions g and h, in the sense that the Bayes filter does not possess a closed-form solution.

The extended Kalman filter (EKF) calculates an approximation to the true belief. It represents this approximation by a Gaussian. In particular, the belief $\text{bel}(x_t)$ at time t is represented by a mean μ_t and a covariance Σ_t. Thus, the EKF inherits from the Kalman filter the basic belief representation, but it differs in that this belief is only approximate, not exact as was the case in Kalman filters.

6.1 The EKF Algorithm

The table given under, states the EKF algorithm. In many ways, this algorithm is similar to the Kalman filter algorithm stated previously. The most important differences are summarized by the following table:

1: **Algorithm Extended_Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
3: $\bar{\Sigma}_t = G_t \, \Sigma_{t-1} \, G_t^T + R_t$
4: $K_t = \bar{\Sigma}_t \, H_t^T (H_t \, \bar{\Sigma}_t \, H_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$
6: $\Sigma_t = (I - K_t \, H_t) \, \bar{\Sigma}_t$
7: return μ_t, Σ_t

Figure 5: The extended Kalman filter (EKF) algorithm.

	Kalman filter	EKF
state prediction (Line 2)	$A_t \, \mu_{t-1} + B_t \, u_t$	$g(u_t, \mu_{t-1})$
measurement prediction (Line 5)	$C_t \, \bar{\mu}_t$	$h(\bar{\mu}_t)$

That is, the linear predictions in Kalman filters are replaced by their nonlinear generalizations in EKFs. Moreover, EKFs use Jacobians G_t and H_t instead of the corresponding linear system matrices A_t, B_t, and C_t in Kalman filters. The Jacobian G_t corresponds to the matrices A_t and B_t, and the Jacobian H_t corresponds to C_t.

6.2 Linearization Via Taylor Expansion

The key idea behind EKF is linearization. Suppose we are given a nonlinear next state function g. Due to non-linearity, it will destroy the gaussian nature of belief. Linearization approximates g by a linear function that is tangent to g at the mean of the Gaussian.

There exist many techniques for linearizing nonlinear functions. EKFs utilize a method called (first order) Taylor expansion. Taylor expansion construct a linear approximation to a function g from g's value and slope. The slope is given by the partial derivative

$$g^{'}(u_t, x_{t-1}) := \frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}} \tag{26}$$

We can see that both the value of g and its slope depend on the arguement of g. A logical choice for selecting the argument is to chose the state deemed most likely at the time of linearization. For Gaussians, the most likely state is the mean of the posterior $_{t-1}$.

$$\begin{aligned} g(u_t, x_{t-1}) &\approx g(u_t, \mu_{t-1}) + g^{'}(u_t, \mu_{t-1})(x_{t-1} - \mu_{t-1}) \\ &= g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}) \end{aligned} \tag{27}$$

where $g^{'}(u_t, \mu_{t-1}) := G_t$

Written as Gaussian, the next state probability is approximated as follows:

$$\begin{gathered} p(x_t|u_t, x_{t-1}) \\ \approx det(2\pi R_t)^{\frac{-1}{2}} e^{\frac{-1}{2}[x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})]^T} \\ R_t^{-1}[x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})] \end{gathered}$$

G_t is a matrix of size n $\times$ n, with n denoting the dimension of the state . This matrix is often called the Jacobian. The value of the Jacobian depends on u_t and μ_{t-1}, hence it differs for different points in time.

EKFs implement the exact same linearization for the measurement function h. Here the Taylor expansion is developed around μ¯t, the state deemed most likely by the robot at the time it linearizes h:

$$\begin{aligned} h(x_t) &\approx h(\overline{\mu_t})h'(\overline{\mu_t})(x_t - \overline{\mu_t}) \\ &= h(\overline{\mu_t}) + H_t(x_t - \overline{\mu_t}) \end{aligned}$$

with $h'(x_t) = \frac{\partial h(x_t)}{\partial x_t}$ Written as a Gaussian , we have

$$p(z_t|x_t) = det(2\pi Q_t)^{-\frac{1}{2}} exp\{-\frac{1}{2}[z_t - h(\overline{\mu_t}) - H_t(x_t - \overline{\mu_t})]^T Q_t^{-1}[z_t - h(\overline{\mu_t}) - H_t(x_t - \overline{\mu_t})]\}$$

6.3 Contributions

We all watched a lot of YouTube videos for state estimation, Bayes filters, Kalman filter, and Extended Kalman filter. We also went through some good books specially known for this topic. Aditya and ashirwad's primary responsibility was the EKF implementation code. We all contributed to solving bugs. Charchit and Shantanu were responsible for preparing the report. Shubhransh's main responsibility was to do the mathematical derivations of the filters.