1 Introduction

In the robot motion planning under uncertainty, we are seeking a policy that brings the robot from a start point A to a goal point B such that the probability of collision with obstacles is minimized. This problem is an example of the more general problem of stochastic optimal control problem.

In Figure (1) shown above, we have three different paths starting from point A and ending at point B. The blue, green, and brown ellipses along each path represent the evolution of uncertainty along that path. Landmarks are shown with the radio beacons. It is assumed that the measurement uncertainty is decreased as the robot gets closer to landmarks. That’s why the uncertainty along the brown path is smaller. Thus, in this case, the brown path is the best because there is more information, and thus the controller can keep the robot away from obstacles.
2 Formulating the stochastic optimal control problem

We formulate the stochastic control problem (problem of finding the control signals, actions, or commands) under the process and sensing uncertainties. Control under process and sensing uncertainty is based on four major pillars: system model, observational model, state estimator, and controller. Each one of them will be explained in detail in the sections to follow. (see Figure 2)

![Decision making (control) under uncertainty](image)

2.1 The System Model:

The system model describes the evolution of the system state. A system could be any structural body that takes inputs and evolves over time such as an aircraft, a robot, a car, or even a society. It is algebraically expressed by

$$x_{k+1} = f(x_k, u_k, w_k)$$

where $x_k$ represents the state at the $k^{th}$ time step, $u_k$ represents the action input, and $w_k$ represents the motion noise which is an external factor from the surrounding environment. The system then evolves recursively based on the system model to $x_{k+1}$. Note that $u_k$ is being fed to the system model through the controller. This will be explained in more details in the motion planner section.

**Example (1):** Let’s assume that we have a unicycle robot as a system model. (see figure 3) In this case $x_k = \begin{pmatrix} x_k & y_k & \theta_k \end{pmatrix}^T$ would represent the state of the robot at the $k^{th}$ time step where $(x_k, y_k)$ is the x-y coordinates, and $\theta_k$ is the heading angle. The action input would be $u_k = \begin{pmatrix} V_k & \omega_k \end{pmatrix}^T$ where $V_k$ is the linear velocity, and $\omega_k$ is the angular velocity of the robot. Finally, the motion noise input would be $w_k = \begin{pmatrix} \tilde{V}_k & \tilde{\omega}_k \end{pmatrix}^T$ where $\tilde{V}_k$ and $\tilde{\omega}_k$ represents the linear and the angular velocity noises respectively. So, the system model
Figure 3: A unicycle robot

would look as follows:

\[
x_{k+1} = \begin{pmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix} + \begin{pmatrix} (V_k + \tilde{V}_k) \cos \theta_k \\ (V_k + \tilde{V}_k) \sin \theta_k \\ \omega_k + \tilde{\omega}_k \end{pmatrix} \Delta t = f(x_k, u_k, w_k)
\]

Figure 4: Euler angles

**Example (2):** Another example of a system model is an aircraft. (see figure 4) The concept here is the same as in the unicycle robot except that the dimensions and the environment will be different. For instance, \( x_k \) would be

\[
x_k = \begin{pmatrix} x_k \\ y_k \\ z_k \\ \theta_k \\ \psi_k \\ \phi_k \end{pmatrix}^T
\]

where \((x, y, z)\) represents the \(x\)-, \(y\)-, and \(z\)-coordinates, and \((\theta, \psi, \phi)\) represents yaw, pitch, and roll angles respectively.
The control inputs for the aircraft are \( u_k = (V_k \omega_k^p \omega_k^\psi \omega_k^q)^T \) where \( V_k \) denotes the linear velocity produced by thrust in the heading direction, \( \omega_k^p \) denotes the yaw angular velocity caused by the rudder, \( \omega_k^\psi \) denotes the pitch angular velocity caused by the elevator, and \( \omega_k^q \) denotes the roll angular velocity caused by the ailerons. The disturbance \( w_k \) in this case could be the wind/drag, and/or gravity.

### 2.2 The Observation Model (Sensor model):

A sensor is a device that provides information about the system state and then sends the measurements to a controller where proper actions can be taken. Metaphorically speaking, a sensor is the eyes and the ears of a system. The sensor is expressed algebraically by

\[
z_k = h(x_k, v_k) \tag{2}
\]

The sensor measures the system state \( x_k \) and generates the measurement \( z_k \). However, this measurement is always noisy and here \( v_k \) denotes the measurement noise.

**Example (1):** There are so many different types of sensors. They differ according to the application they are used for. For example, a gyroscope is a sensing device used to measure the orientation of an aircraft. The gyroscope measures the provided orientation of the aircraft, meaning \( x_k \), and tries to maintain it at that level by detecting any change in orientation due to turning or climbing. \( v_k \) here denotes the error in gyroscope measurements. At every time step, gyroscope returns \( z_k \) based on the current state \( x_k \).

![Figure 5: A robot sensor measuring the distance from landmarks.](image)

**Example (2):** Another practical example is a mobile robot sensor that measures the distance of the robot from some landmarks. (see figure 5) Suppose we have \( n \) number of landmarks \( \{L_1, L_2, ..., L_n\} \), then measurements \( z \) can be modelled the distance between robot position \((x, y)\), and the landmark position \( L_i \) and adding it to the sensor noise \( v_i \) as follows:

\[
z = \begin{pmatrix} 
\zeta_1 \\
\zeta_2 \\
\vdots \\
\zeta_n 
\end{pmatrix} = \begin{pmatrix} 
\| (x, y)^T - L_1 \| + v_1 \\
\| (x, y)^T - L_2 \| + v_2 \\
\vdots \\
\| (x, y)^T - L_n \| + v_n 
\end{pmatrix} = h(x, v)
\]
where \( L_i = \begin{pmatrix} L_i^x & L_i^y \end{pmatrix}^T \) is the landmark, and the Euclidean distance is calculated as follows:
\[
\| (x, y)^T - L_i \| = \sqrt{(x - L_i^x)^2 + (y - L_i^y)^2}
\]

2.3 The State Estimator (filter):

**Partially observable system:** A system is said to be fully observable if \( z_k = x_k \), but in real-world applications, measurements are never perfect, i.e., noisy, which means that we do not have a perfect knowledge of the state. This is called partially observable system.

**Observation history and control history:** In partially-observable systems, we have to make decisions based on the available data, i.e., observation history \( z_0:k = \{z_1, z_2, ..., z_k\} \), and control history \( u_{0:k-1} = \{u_1, u_2, ..., u_{k-1}\} \).

**Belief:** However, observation history and control history are expanding with time. So, we compress these histories by computing the probability density function (pdf) of the system state conditioned on \( z_0:k, u_{0:k-1} \):
\[
b_k = p(x_k | z_{0:k}, u_{0:k-1})
\]  
Equation (3)

State estimator generates the ”belief” of the system which can be computed recursively by:
\[
b_k = \tau(b_{k-1}, u_{k-1}, z_k)
\]  
Equation (4)

\( b_k \) here is the belief state at time step \( k \), and ”belief” means the probability density function (pdf) over the state space.

**Gaussian Belief:** Gaussian belief could be expressed in the scalar version as follows:
\[
b(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(x-\mu)^2\sigma^{-2}(x-\mu)}
\]  
Equation (5)

or in the multi-dimension/vector version as follows:
\[
b(x) = \frac{1}{\sqrt{2\pi|P|}} e^{-\frac{1}{2}(x-\mu)^T P^{-1}(x-\mu)}
\]  
Equation (6)

where the Gaussian belief is parametrized by two variables \( \mu \), and \( P \), i.e., mean and covariance. Therefore, we can show it as: \( b \equiv \begin{pmatrix} \mu \\ P \end{pmatrix} \) (see the figure below).

**Kalman Filter:** Here we focus on the Kalman filtering as the state estimator. The intuition behind Kalman filter is that it performs the estimation in two steps: prediction and update.
Figure 6: Probability distribution of the next belief state $b(x)$

Suppose we have the following linear system:

$$x_{k+1} = Ax_k + Bu_k + Gw_k, w_k \sim N(0, Q) \text{ process noise}$$

$$z_k = Hx_k + Mv_k, v_k \sim N(0, R) \text{ observation/sensor noise}$$

During the prediction step, the mean and covariance of the previous step are calculated as follows:

$$b_k \equiv \begin{pmatrix} \hat{x}_k^+ \rightarrow \text{mean} \\ P_k^+ \rightarrow \text{covariance} \end{pmatrix}$$

$$\begin{pmatrix} \hat{x}_k^+ \\ P_k^+ \end{pmatrix} \xrightarrow{\text{prediction}} \begin{pmatrix} \hat{x}_{k+1}^- \\ P_{k+1}^- \end{pmatrix} = \begin{pmatrix} A\hat{x}_k^+ + Bu_k \\ AP_k^+ A^T + GQG^T \end{pmatrix}$$

In the update step, the next belief state is computed as follows:

$$\begin{pmatrix} \hat{x}_{k+1}^- \\ P_{k+1}^- \end{pmatrix} \xrightarrow{\text{update}} \begin{pmatrix} \hat{x}_{k+1}^+ \\ P_{k+1}^+ \end{pmatrix} = \begin{pmatrix} \hat{x}_{k+1}^- + K_{k+1}(z_{k+1} - H(\hat{x}_{k+1}^- + Bu_k)) \\ (I - K_{k+1}H)P_{k+1}^- \end{pmatrix}$$

Combining both steps results in:

$$b_{k+1} \equiv \begin{pmatrix} \hat{x}_{k+1}^+ \\ P_{k+1}^+ \end{pmatrix} = \begin{pmatrix} A\hat{x}_k^+ + Bu_k + K_{k+1}(z_{k+1} - H(\hat{x}_{k+1}^- + Bu_k)) \\ (I - K_{k+1}H)(AP_k^+ A^T + GQG^T) \end{pmatrix}$$

where $K_{k+1}$ is the Kalman gain at time step $k+1$, or:

$$K_{k+1} = (AP_k^+ A^T + GQG^T)H^T \times (H(AP_k^+ A^T + GQG^T)H^T + MRM^T)^{-1}$$

### 2.4 Motion Planner (controller):

The last major block in the stochastic control is where decisions are being made to generate control according to:

$$J^*(b_0) = c(b_0, u_0) + c(b_1, u_1) + \ldots = \sum_{k=0}^{\infty} c(b_k, u_k)$$
where \(c(b, u)\) is the cost of taking action \(u\) at belief \(b\), and \(J^\pi(b_0)\) is the cost-to-go under policy \(\pi\). So, the control is generated as:

\[
u_k = \pi(b_k) \tag{12}\]

**Cost-to-go:** The total cost of going from \(b_0\) to goal under the policy (controller) \(\pi\) Therefore, equation (11) would look like:

\[
J^\pi(b_0) = c(b_0, \pi(b_0)) + c(b_1, \pi(b_1)) + c(b_2, \pi(b_2)) + \ldots = \sum_{k=0}^{\infty} c(b_k, \pi(b_k)) \tag{13}\]

In order to minimize the cost to go, \(\pi^*\) should be calculated

\[
J^{\pi^*} = \min_{\pi} J^\pi(b_0) \tag{14}\]

**Example:** In this example we consider the quadratic cost:

\[
c(b_k, u_k) = (\hat{x}_k^+ - x_k^d)^T(\hat{x}_k^+ - x_k^d) + u_k^T u_k \tag{15}\]

where we are trying to minimize \((\hat{x}_k^+ - x_k^d)^T(\hat{x}_k^+ - x_k^d)\), i.e., to push the state \(x_k\) toward \(x_k^d\), and we are trying to minimize \(u_k^T u_k\), i.e., to have a reasonable control effort. Therefore, the cost-to-go will have the following form:

\[
\Rightarrow J^\pi = \sum_{k=0}^{\infty} (\hat{x}_k^+ - x_k^d)^T(\hat{x}_k^+ - x_k^d) + \pi(\hat{x}_k^+)^T \pi(\hat{x}_k^+) \tag{16}\]

2.5 Whole Loop:

After the control variable \(u_k\) is generated, it will be directly sent to the system model to generate the next state \(x_{k+1}\). Then again we measure \(x_{k+1}\) and we get the measurement \(z_{k+1}\), after that we feed \(z_{k+1}\) and \(u_k\) to the estimator to get the next belief state \(b_{k+1}\). Finally the controller generates \(u_{k+1}\) based on \(b_{k+1}\). This process occurs in a recursive manner until it reaches the goal state.

3 Summary:

We looked at the problem of motion planning under uncertainty as an example of the stochastic optimal control problem under the process and sensing uncertainties. The four major blocks involved in generating control in the stochastic optimal control process were discussed, explained and illustrated with some real-world examples. In future, we are planning to implement a basic stochastic control scheme, such as Linear Quadratic Gaussian (LQG) controller and test it on simple robot model motion planning.