Robotics Programming Laboratory

Bertrand Meyer
Jiwon Shin

Lecture 6: Localization

This lecture is based on “Probabilistic Robotics” by Thrun, Burgard, and Fox (2005).
Localization: process of locating an object in space

Types of localization

- Global localization: initial pose unknown
  - Markov localization
  - Particle filter localization
- Local localization: initial pose known
  - Kalman filter localization
Probabilistic robotics

Uncertainty!

- Environment, sensor, actuation, model, algorithm
- Represent uncertainty using the calculus of probability theory

Probability theory

- $X$: random variable
  - Can take on discrete or continuous values
- $P(X = x)$, $P(x)$: probability of the random variable $X$ taking on a value $x$
- Properties of $P(x)$
  - $P(X = x) \geq 0$
  - $\sum_X P(X = x) = 1$ or $\int_X p(X = x) = 1$
Probability

- $P(x,y)$: joint probability
  - $P(x,y) = P(x) P(y)$: $X$ and $Y$ are independent

- $P(x \mid y)$: conditional probability of $x$ given $y$
  - $P(x \mid y) = p(x)$: $X$ and $Y$ are independent
  - $P(x,y \mid z) = P(x \mid z) P(y \mid z)$: conditional independence
  - $P(x \mid y) = P(x,y) / P(y)$
  - $P(x,y) = P(x \mid y) P(y) = P(y \mid x) P(x)$

- $P(x \mid y) = \frac{P(y \mid x) P(x)}{P(y)} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$: Bayes' rule
  - $P(y) = \sum_x P(x,y) = \sum_x P(y \mid x) P(x)$: Law of total probability
Bayes’ rule

\[ P(\text{door}=\text{open} \mid \text{sensor}=\text{far}) = \frac{P(\text{far} \mid \text{open}) P(\text{open})}{P(\text{far})} \]

\[ = \frac{P(\text{far} \mid \text{open}) P(\text{open})}{P(\text{far} \mid \text{open}) P(\text{open}) + P(\text{far} \mid \text{closed}) P(\text{closed})} \]
Bayes’ filter

\[ \text{bel}(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) \] : belief on the robot’s state \( x_t \) at time \( t \)

Compute robot’s state: \( \text{bel}(x_t) \)

- Predict where the robot should be based on the control \( u_{1:t} \)
- Update the robot state using the measurement \( z_{1:t} \)
Markov localization

World

Measurement
Markov localization

Predict

Update

Belief
Markov localization

Markov_localize ( bel⁺₁: ARRAY[BELIEF_ROBOT_POSE];
    u⁺: ROBOT_CONTROL;
    z⁺: SENSOR_MEASUREMENT;
    m: MAP) : BELIEF_ROBOT_POSE

local

   bel⁺*: ARRAY[BELIEF_ROBOT_POSE_PARTICLE]
   bel⁺: ARRAY[BELIEF_ROBOT_POSE_PARTICLE]
   x⁺: ROBOT_POSE

   do

   create bel⁺*.make_from_array( bel⁺₁ )
   create bel⁺.make_from_array( bel⁺₁ )
   from i := bel⁺.lower until i > bel⁺.upper loop
       x⁺ := bel⁺[i].pose
       bel⁺*[i] := ∫p(x⁺ | u⁺, x⁺⁻¹, m) bel⁺⁻¹(x⁺⁻¹) dx⁺⁻¹

   Predict

   Update
       bel⁺[i] := η p(z⁺ | x⁺⁻¹, m) bel⁺*[i]
       i := i + 1

   end

Result := bel⁺

end
Representation of the robot states

\[ \text{bel}(x, y, \theta) \]
Markov localization

- Can be used for both local localization and global localization
  - If the initial pose \( (x^*_0) \) is known: point-mass distribution
    \[
    \text{bel}(x_0) = \begin{cases} 
    1 & \text{if } x_0 = x^*_0 \\
    0 & \text{otherwise}
    \end{cases}
    \]
  - If the initial pose \( (x^*_0) \) is known with uncertainty \( \Sigma \): Gaussian distribution with mean at \( x^*_0 \) and variance \( \Sigma \)
    \[
    \text{bel}(x_0) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (x_0 - x^*_0)^T \Sigma^{-1} (x_0 - x^*_0) \right\}
    \]
  - If the initial pose is unknown: uniform distribution
    \[
    \text{bel}(x_0) = \frac{1}{|X|}
    \]
- Computationally expensive
  - Higher accuracy requires higher grid resolution
What if we keep track of multiple robot pose?

Measurement
Particle filter

A sample-based Bayes filter

- Approximate the posterior $\text{bel}(x_t)$ by a finite number of particles
- Each particle represents the probability of a particular state vector given all previous measurements
- The distribution of state vectors within the particle is representative of the probability distribution function for the state vector given all prior measurements
Importance sampling

Generate samples from a distribution

\[
E_f[ I(x \in A) ] = \int f(x) I(x \in A) \, dx \\
= \int \frac{f(x)}{g(x)} g(x) I(x \in A) \, dx \\
= E_g[ \, w(x) I(x \in A) \, ]
\]

\[f(x): \text{target distribution}\]
\[g(x): \text{proposal distribution} \rightarrow f(x) > 0 \Rightarrow g(x) > 0\]
Particle filter localization

\[
\text{particle\_filter\_localize} \ (X_{t-1}: \text{ARRAY}[^{\text{BELIEF\_ROBOT\_POSE\_PARTICLE}}] ; u_t: \text{ROBOT\_CONTROL} ; z_t: \text{SENSOR\_MEASUREMENT} ; m: \text{MAP}) : \text{ARRAY}[^{\text{BELIEF\_ROBOT\_POSE\_PARTICLE}}]
\]

local
\[
X_t : \text{ARRAY}[^{\text{BELIEF\_ROBOT\_POSE\_PARTICLE}}]
\]
\[
x_t : \text{ROBOT\_POSE}
\]
do
\[
create \ X_t.make\_from\_array( X_{t-1} )
\]
from \ i := X_{t-1}.lower \ until \ i > X_{t-1}.upper \ loop
\[
x_{t-1} := X_{t-1}[i].\text{pose}
\]
\[\text{Predict} \quad X_t[i].\text{pose} := \text{sample\_motion\_model}( x_{t-1}, u_t, t_{\text{current}} - t_{\text{previous}} )\]
\[\text{Update} \quad X_t[i].\text{weight} := \text{compute\_sensor\_measurement\_prob}(z_t, m)\]
i := i + 1
end
\[\text{Result} := \text{resample}(X_t)\]
end
Sampling from motion model

\[ \text{sample\_motion\_mode}( \ x: \text{ROBOT\_POSE}; \ u: \text{ROBOT\_CONTROL} \ \Delta t: \text{REAL\_64} ): \text{ROBOT\_POSE} \]

\[
\text{local} \\
\quad x': \text{ROBOT\_POSE} \\
\quad u': \text{ROBOT\_CONTROL} \\
\text{do} \\
\quad u'.v := \text{Gaussian\_sample}( u.v, a_1 u.\sigma_v^2 + a_2 u.\sigma_\omega^2 ) \\
\quad u'.\omega := \text{Gaussian\_sample}( u.\omega, a_3 u.\sigma_v^2 + a_4 u.\sigma_\omega^2 ) \\
\quad x'.x := x.x - \frac{u'.v}{u'.\omega} \sin( x.\theta ) + \frac{u'.v}{u'.\omega} \sin( x.\theta + u'.\omega \Delta t ) \\
\quad x'.y := x.y + \frac{u'.v}{u'.\omega} \cos( x.\theta ) - \frac{u'.v}{u'.\omega} \cos( x.\theta + u'.\omega \Delta t ) \\
\quad x'.\theta := x.\theta + u'.\omega \Delta t + \text{Gaussian\_sample}( 0, a_5 u.\sigma_v^2 + a_6 u.\sigma_\omega^2 ) \Delta t \\
\text{Result} := x' \\
\text{end} \]
Resampling

Roulette wheel sampling

Stochastic universal sampling

distance between two samples = total weight / number of samples
starting sample: random number in [0, distance between samples]
Particle filter localization

- Global localization
  - Track the pose of a mobile robot without knowing the initial pose
- Can handle kidnapped robot problem with little modification
  - Insert some random samples at every iteration
  - Insert random samples proportional to the average likelihood of the particles
- Approximate
  - Accuracy depends on the number of samples
If we know the initial pose, can we do better?

Estimate the robot pose with a Gaussian distribution!

Measurement
Properties of Gaussian distribution

Univariate

\[ X \sim N (\mu, \sigma^2) \]
\[ Y = aX + b \]
\[ X_1 \sim N (\mu_1, \sigma_1^2) \]
\[ X_2 \sim N (\mu_2, \sigma_2^2) \]
\[ \Rightarrow p(X_1) \cdot p(X_2) \sim N \left( \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \mu_1 + \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \mu_2, \frac{1}{\sigma_1^{-2} + \sigma_2^{-2}} \right) \]

Multivariate

\[ X \sim N (\mu, \Sigma) \]
\[ Y = AX + B \]
\[ X_1 \sim N (\mu_1, \Sigma_1) \]
\[ X_2 \sim N (\mu_2, \Sigma_2) \]
\[ \Rightarrow p(X_1) \cdot p(X_2) \sim N \left( \frac{\Sigma_2}{\Sigma_1 + \Sigma_2} \mu_1 + \frac{\Sigma_1}{\Sigma_1 + \Sigma_2} \mu_2, \frac{1}{\Sigma_1^{-1} + \Sigma_2^{-1}} \right) \]
Kalman filter localization

A special case of Markov localization

Assumptions:

- The system is linear (describable as a system of linear equations)
- The noise in the system has a Gaussian distribution
- The error criteria is expressed as a quadratic equation (e.g. sum-squared error)
Kalman filter localization

Belief

Predict

Update
Kalman filter

Kalman_filter ( \( x_{t-1}: \text{ROBOT\_POSE}; \)
    \( u_t: \text{ROBOT\_CONTROL}; \)
    \( z_t: \text{SENSOR\_MEASUREMENT} \) : \text{ROBOT\_POSE}
)

\text{local}

\( \mu_{t-1}, \mu^*_t, \mu_t : \text{MEAN\_ROBOT\_POSE} \)
\( \Sigma_{t-1}, \Sigma^*_t, \Sigma_t : \text{ROBOT\_POSE\_COVARIANCE} \)
\( K_t : \text{KALMAN\_GAIN} \)

\text{do}

\( \mu_{t-1} := x_{t-1}.\text{mean} \)
\( \Sigma_{t-1} := x_{t-1}.\text{covariance} \)

\text{Predict}
\( \mu^*_t := A_t \mu_{t-1} + B_t u_t \)
\( \Sigma^*_t := A_t \Sigma_{t-1} A_t^T + R_t \)
\( K_t := \Sigma^*_t C_t^T (C_t \Sigma^*_t C_t^T + Q_t)^{-1} \)

\text{Update}
\( \mu_t := \mu^*_t + K_t (z_t - C_t \mu^*_t) \)
\( \Sigma_t := (I - K_t C_t) \Sigma^*_t \)

\text{Result} := \text{create} \{ \text{ROBOT\_POSE} \}.\text{make\_with\_variables}( \mu_t, \Sigma_t )

\text{end}
Kalman filter: prediction

\[ \mu^*_t = A_t \mu_{t-1} + B_t u_t \]

- system state estimation for time \( t \)

\[ \Sigma^*_t = A_t \Sigma_{t-1} A_t^\top + R_t \]

- estimation the system uncertainty

\( A_t \): process matrix that describes how the state evolves from \( t \) to \( t-1 \) without controls or noise

\( B_t \): matrix that describes how the control \( u_t \) changes the state from \( t \) to \( t-1 \)

\( R_t \): Process noise covariance
Kalman filter: update

\[ K_t = \Sigma^*_t C_t^T (C_t \Sigma^*_t C_t^T + Q_t)^{-1} \]

- Kalman gain: how much to trust the measurement
- The lower the measurement error relative to the process error, the higher the Kalman gain will be

\[ \mu_t = \mu^*_t + K_t (z_t - C_t \mu^*_t) \]

- update \( \mu_t \) with measurement

\[ \Sigma_t = (I - K_t C_t) \Sigma^*_t \]

- estimate uncertainty of \( \mu_t \)

\( C_t \): measurement matrix relating the state variable and measurement
\( Q_t \): measurement noise covariance
Extended Kalman filter

Extended_Kalman_filter ( x_{t-1}: \text{ROBOT_POSE}; 
    u_t: \text{ROBOT_CONTROL}; 
    z_t: \text{SENSOR_MEASUREMENT }): \text{ROBOT_POSE}

local
\begin{align*}
\mu_{t-1}, \mu^*_t, \mu_t & : \text{MEAN_ROBOT_POSE} \\
\Sigma_{t-1}, \Sigma^*_t, \Sigma_t & : \text{ROBOT_POSE_COVARIANCE} \\
K_t & : \text{KALMAN_GAIN}
\end{align*}
do
\begin{align*}
\mu_{t-1} & := x_{t-1}.\text{mean} \\
\Sigma_{t-1} & := x_{t-1}.\text{covariance}
\end{align*}

\textbf{Predict}
\begin{align*}
\mu^*_t & := g(u_t, \mu_{t-1}) \quad \text{-- linearized state transition: } g(u_t, x_{t-1}) = g(u_t, x_{t-1}) + G_t (x_{t-1} - u_{t-1}) \\
\Sigma^*_t & := G_t \Sigma_{t-1} G_t^T + R_t \\
K_t & := \Sigma^*_t H_t^T (H_t \Sigma^*_t H_t^T + Q_t)^{-1}
\end{align*}

\textbf{Update}
\begin{align*}
\mu_t & := \mu^*_t + K_t (z_t - h(\mu^*_t)) \quad \text{-- linearized measurement: } h(x_t) = h(u^*_t) + H_t (x_t - u^*_t) \\
\Sigma_t & := (I - K_t H_t) \Sigma^*_t
\end{align*}

\textbf{Result} := create \{ROBOT_POSE\}.make_with_variables( \mu_t, \Sigma_t )

den
Kalman filter localization

- Local localization
- Locally linearize update matrices for non-linear systems
- Unimodal model is not always realistic for many robot situations
- Matrix inversion is expensive
  - Limits the number of possible state values