Robotics
Exercise 9

Marc Toussaint
Machine Learning & Robotics lab, U Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

January 6, 2015

1 Particle Filtering the location of a car

Start from the code in `RoboticsCourse/05-car`.

The `CarSimulator` simulates a car exactly as described on slide 04:48 (using Euler integration with step size 1sec). At each time step a control signal $u = (v, \phi)$ moves the car a bit and Gaussian noise with standard deviation $\sigma_{\text{dynamics}} = .03$ is added to x, y and θ. Then, in each step, the car meassures the relative positions of m landmark points, resulting in an observation $y_t \in \mathbb{R}^{m\times 2}$; these observations are Gaussian-noisy with standard deviation $\sigma_{\text{observation}} = .5$. In the current implementation the control signal $u_t = (.1, .2)$ is fixed (roughly driving circles).

a) Odometry (dead reckoning): First write a particle filter (with $N = 100$ particles) that ignores the observations. For this you need to use the cars system dynamics (described on 04:48) to propagate each particle, and add some noise σ_{dynamics} to each particle (step 3 on slide 07:22). Draw the particles (their x, y component) into the display. Expected is that the particle cloud becomes larger and larger.

b) Next implement the likelihood weights $w_i \propto P(y_t|x_t^i) = \mathcal{N}(y_t|y(x_t^i), \sigma) \propto e^{-\frac{1}{2}(y_t - y(x_t^i))^2/\sigma^2}$ where $y(x_t^i)$ is the (ideal) observation the car would have if it were in the particle possition x_t^i. Since $\sum_i w_i = 1$, normalize the weights after this computation.

c) Test the full particle filter including the likelihood weights (step 4) and resampling (step 2). Test using a larger ($10\sigma_{\text{observation}}$) and smaller ($\sigma_{\text{observation}}/10$) variance in the computation of the likelihood.

2 Kalman filter

We consider the same car example as above, but track the car using a Kalman filter.

a) To apply a Kalman filter (slide 07:27) we need Gaussian models for $P(x_t \,|\, x_{t\text{-}1}, u_{t\text{-}1})$ as well as $P(y_t|x_t)$. We assume that the dynamics model is given as a local Gaussian of the form

$$P(x_{t+1} \,|\, x_t, u_t) = \mathcal{N}(x_{t+1}|x_t + B(x_t)u_t, \sigma_{\text{dynamics}})$$

where the matrix $B(x_t) = \frac{\partial x_{t+1}(x_t,u_t)}{\partial u_t}$ gives the local linearization of the deterministic discrete time car dynamics

$$x_{t+1}(x, u) = x + \tau \begin{pmatrix} u_1 \cos x_3 \\ u_1 \sin x_3 \\ (u_1/L)\tan u_2 \end{pmatrix}$$

What is $B(x_t)$ for the car dynamics?

b) Concerning the observation likelihood $P(y_t|x_t)$ we assume

$$P(y_t|x_t, \theta_{1:N}) = \mathcal{N}(y_t \,|\, C(x_t)x_t + c(x_t), \sigma_{\text{observation}})$$

What is the matrix $C(x_t)$ (the Jacobian of the landmark positions w.r.t. the car state) in our example?

c) Start with the code in `RoboticsCourse/06-kalmanSLAM`.

Write a Kalman filter to track the car. You can use the routine `getObservationJacobianAtState` to access $C(x_t) = \frac{\partial y}{\partial x}$. Note that $c(x_t) = \widehat{y}_t - C(x_t)x_t$, where $\widehat{y}_t$ is the mean observation in state x_t (there is another routine for this).