Robotics
Exercise 9

Marc Toussaint
Lecturer: Duy Nguyen-Tuong
TAs: Philipp Kratzer, Janik Hager, Yoojin Oh
Machine Learning & Robotics lab, U Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

January 8, 2019

1 Particle Filtering the location of a car (6 points)

You are going to implement a particle filter. Access the code as usual:

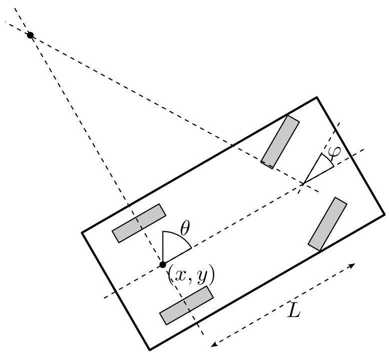
1. To make sure you have an updated version of the repository, run '*git pull*' and '*git submodule update*'
2. For python run: *'jupyter-notebook py/07-particle_filter/07-particle_filter.ipynb'*
3. For C++ run: *'cd cpp/07-particle_filter'*, *'make'*, *'./x.exe'*

The motion of the car is described by the following:

State $q = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$ **Controls** $u = \begin{pmatrix} v \\ \varphi \end{pmatrix}$

Sytem equation $\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v\ \cos\theta \\ v\ \sin\theta \\ (v/L)\ \tan\varphi \end{pmatrix}$

$|\varphi| < \Phi$



The `CarSimulator` simulates the described car (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 measures the relative positions of m landmark points (green cylinders), 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 above) to propagate each particle, and add some noise σ_{dynamics} to each particle (step 3 on slide 06:24). Draw the particles (their x, y component) into the display. Expected is that the particle cloud becomes larger and larger. (2 P)

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 position x_t^i. Since $\sum_i w_i = 1$, normalize the weights after this computation. (2 P)

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

2 Bayes Smoothing (6 points)

In the lecture we derived the Bayesian filter: given information on the past (observations $y_{0:t}$ and controls $u_{0:t\text{-}1}$) it estimates the current state x_t. However, we can use the available information on $y_{0:T}$ and $u_{0:T}$ also to get a Bayes-optimal estimate of a past state x_t at a previous time $t < T$. This estimate should be "better" than the forward filtered $P(x_t \,|\, y_{0:t}, u_{0:t\text{-}1})$ because it uses the additional information on $y_{t+1:T}$ and $u_{t:T}$. This is called Bayes smoothing (slides 06:32 – 06:33).

Derive the backward recursion $\beta_t(x_t) := P(y_{t+1:T}|x_t, u_{t:T})$ (the likelihood of all future observations given x_t and knowledge of all subsequent controls) of Bayes smoothing on slide 06:33. Explain in each step which rule/transformation you applied.