REX 2015 opgave 5

Self Localization

Kim S. Pedersen, Søren Olsen

In this exercise you will program the robot to estimate its own position from visual observations. More precisely, the robot knows the location of two unique landmarks (two boxes covered with checkers with unique colors (red and green)). When the robot sees one of the landmarks it can improve its estimate of its own position.

You should solve this problem by implementing a particle filter for estimation of the robots pose (its position and orientation). We provide code for recognizing the two different landmarks and for measuring distances to them as well as a template for the particle filter algorithm (you need to fill in the blanks). You find this code in Absalon as part of this exercise (read the README.txt file).

The robot needs to know the position of the two landmarks (it is fixed), so it has to be represented some how in your program. In the code we provide, one landmark is located in (0,0) and the other in (300,0), which means that the landmarks should be physically positioned with 300 cm distance in between.

As indicated above, the robot has to be able to distinguish between the two landmarks, and we do this by the colors of the checkers (alternatively the orientation of the checkers pattern may be used - possibilities are horizontal or vertical). In order for the robot camera to see the landmarks it can be necessary to raise them a bit from the ground.

Once you have an estimate of the robot pose (position and orientation) you should drive the robot into the centre between the two landmarks. Notice that you probably need to have a driving strategy to make sure you have seen both landmarks before you attempt to find the centre between the landmarks.

Hint 1: The distance between the two landmarks does not have to be precisly 300 cm in order for the program to work. Just make sure that the distance corresponds roughly (an error of around 10 cm should not make a big difference).

Hint 2: You have to estimate the robots position and orientation, which means three parameters (x, y, θ). However to begin with, we suggest that you focus on estimating the position (x, y). When this work, you can try to estimate the orientation θ as well.

Hint 3: For the deterministic part of the dynamical model you may use the motor control commands that you issue to the robot or you can try to use odometry estimates, if your robot provides this.

Hint 4: The distance measurement code uses information about the focal length of the camera. The provided code provides a roughly right estimate that work for the different robot cameras. If you like you can try to improve this focal length estimate by using the `camera_calibrator` program that you find in the source code. Talk to Kim about instructions on how to use this code.

Hint 5: We suggest that you assume that your dynamical model is only dependent on the current velocity of the robot and that you update each particle based on its current orientation and moving the particle position along this direction by the amount given by the current velocity. The stochastic part of the model is then given by simply adding some Gaussian noise to the position and some noise to the orientation. We provide a simple function for adding noise called `add_uncertainty` which adds Gaussian noise to position and von Mises distributed noise to the orientation.

Hint 6: As observation model you can use a Gaussian distribution on distances. It should have mean equal to the distance from the particle state to the identified landmark and have some variance which reflects the measurement error. For orientation you can also use a Gaussian distribution with mean equal to the direction to the identified landmark and a variance that reflects the measurement error. You should split the two aspects of the measurement (distance and orientation) into two Gaussian models and simply multiply them to get the weight update (hence assume independence between measured distance d_M and orientation angle θ_M):

$$p(d_M, \theta_m | x, y, \theta) = p(d_M | x, y) p(\theta_M | x, y, \theta) \tag{1}$$

For particle (i), we have for the distance distribution:

$$p(d_M | x^{(i)}, y^{(i)}) = \mathcal{N}(d_M | d^{(i)}, \sigma_d^2) = \frac{1}{\sqrt{2\pi\sigma_d^2}} \exp\left(-\frac{(d_M - d^{(i)})^2}{2\sigma_d^2} \right) \ . \tag{2}$$

Here $d^{(i)}$ is the distance from the particle state position to the landmark we are observing.
Similarly for orientation of the (i)'th particle, we will assume

$$p(\theta_M | x^{(i)}, y^{(i)}, \theta^{(i)}) = \mathcal{N}(\theta_M | \delta\theta^{(i)}, \sigma_\theta^2) = \frac{1}{\sqrt{2\pi\sigma_\theta^2}} \exp\left(-\frac{(\theta_M - \delta\theta^{(i)})^2}{2\sigma_\theta^2} \right) \ , \tag{3}$$

where $\delta\theta^{(i)} = \theta^{(i)} - \phi^{(i)}$ and if $\delta\theta^{(i)} \geq \pi$ then $\delta\theta^{(i)} = \delta\theta^{(i)} - 2\pi$. Here $\phi^{(i)}$ denotes the angle to the landmark given the particle state position and $\theta^{(i)}$ denotes the particles state orientation. We need the last part to handle periodicity of the orientation angle.

NOTICE: You can come up with much more complicated measurement models. Alternatively, you can use the von Mises distribution instead of a Gaussian for the angular distribution. Von Mises is a Gaussian-like distribution on periodic variables (e.g. orientation angles). Code for this distribution is available.