

Humanoid Robots

Exercise Sheet 5 - ICP and Particle Filter

Exercise 9 (10 points)

Execute the ICP algorithm to align the set of 2D points P to those of set Q (as close as possible).

$$P = \left\{ \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 1.5 \\ 3 \end{pmatrix}, \begin{pmatrix} 2 \\ 3.8 \end{pmatrix}, \begin{pmatrix} 3 \\ 5 \end{pmatrix}, \begin{pmatrix} 2.7 \\ 6 \end{pmatrix}, \begin{pmatrix} 2.5 \\ 7 \end{pmatrix}, \begin{pmatrix} 3 \\ 8 \end{pmatrix}, \begin{pmatrix} 3.5 \\ 8.5 \end{pmatrix} \right\}$$
$$Q = \left\{ \begin{pmatrix} 2 \\ 3 \end{pmatrix}, \begin{pmatrix} 3 \\ 3 \end{pmatrix}, \begin{pmatrix} 4 \\ 3 \end{pmatrix}, \begin{pmatrix} 5 \\ 3 \end{pmatrix}, \begin{pmatrix} 5 \\ 4 \end{pmatrix}, \begin{pmatrix} 5 \\ 5 \end{pmatrix}, \begin{pmatrix} 5 \\ 6 \end{pmatrix}, \begin{pmatrix} 6 \\ 6 \end{pmatrix} \right\}$$

Also, compare between using the *closest point* and *point-to-line* matching methods to find the corresponding points of Q and P .

Note: The line is represented by the closest point and the closer of its two direct neighbours.

Exercise Steps:

Implement the missing parts in `'src/09_icp/src/ICP.cpp'` according to the following instructions:

- Implement the function `distance` that calculates the Euclidean distance between a pair of 2D points.
- Implement the function `closestPointOnLine` that computes the closest point that lies on a given line to a given 2D point.
- Implement the function `min` that gets the minimum value within a vector of values.
- Implement the function `euclideanCorrespondences` that computes the corresponding points in P to those points in Q , using the *closest point* matching method.
- Implement the function `closestPointToLineCorrespondences` that computes the corresponding points in P to those points in Q , using the *point-to-line* matching method.

- f) Implement the function `calculateAffineTransformation` that computes the affine transformation matrix needed to align the previously computed corresponding points to the points of Q .
- g) Implement the function `applyTransformation` that applies the affine transformation matrix on the points in P .
- h) Implement the function `computeError` that computes the error between the points in Q and the transformed corresponding points.
- i) Implement the function `iterateOnce` that performs one iteration of ICP and prints the error of that iteration.

The Gnuplot script `scripts/plot.gp` and the image on the Wiki show Q , P , and the aligned points P_1 according to the two correspondence strategies.

Exercise 10 (10 points)

A vacuum cleaner robot drives through a corridor with three ceiling lights. The robot does not know its starting position within the corridor, but it has a map of the corridor indicating the positions of the three light sources:

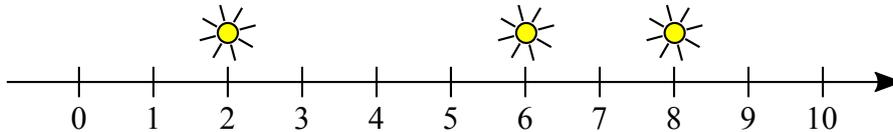


Figure 1: Corridor with three light sources

The robot is equipped with a light sensor and from the measured brightness it can calculate the distance to the nearest light source.

The robot provides a log file containing odometry measurements (i.e., the displacements of the robot) and the measured distance to the nearest light source for a number of time steps. Localize the robot on the map using a particle filter (Monte Carlo localization) according to the following steps:

- a) Implement the function `gaussianProbability` that calculates the probability of a measurement according to a Gaussian distribution. This function is called $\phi(d, \sigma)$ on slide 76. σ is the standard deviation of the distribution.
- b) Implement the function `sampleFromGaussian` that draws a sample from a Gaussian distribution with mean μ and standard deviation σ (see slide 64). For generating random numbers, you can use the standard C function `rand()` that returns a random integer between 0 and `RAND_MAX`.
- c) Implement the function `initParticles` that initializes the particles as follows:

- The position \mathbf{x} of the robot should be linearly distributed in the interval $[0, 10]$. You can either distribute the positions equidistantly or sample the positions from a linear distribution in the given interval.
 - All particles should have the same weight `weight` and the weights must sum up to 1.
- d) Implement the function `normalizeWeights` that normalizes the weights of the particles so that they sum up to 1.
- e) When the robot travels for the distance u_x , the particles have to be displaced accordingly. However, u_x is subject to measurement errors and noise. Hence, instead of shifting all particles by the same distance u_x , we sample the displacement from a Gaussian with mean u_x . Implement the function `integrateMotion` for sampling a new position for each particle according to this motion model.
- This step is called the *prediction step* of the Monte Carlo filter.
- f) Implement the method `getDistanceToNearestLight` that calculates the distance from the robot's position \mathbf{x} to the nearest light source according to map in Fig. 1.
- g) While traveling, the robot measures the distance to the nearest light source using its light sensor. Integrate the measurement in the method `integrateObservation` by changing the particles' weights according to the observation model

$$p(\tilde{z} | x) = \phi(z - \tilde{z}, \sigma) \quad (1)$$

where \tilde{z} is the measured distance (given as an argument to the function), z is the distance between the particle pose and the nearest light source (calculate this for each particle using `getDistanceToNearestLight`), and σ is the standard deviation (given as an argument).

This step is called the *correction step* of the Monte Carlo filter.

- h) Implement low variance resampling in the method `resample` according to slides 33–35.

Hint: When doing calculations, make sure to cast integer values to double first. The expression $1.5 + 3/2$ will evaluate to 2.5 because $3/2$ will be truncated to the integer 1, whereas $1.5 + 3.0/2.0$ will evaluate to 3.0 as expected.

If you have Gnuplot installed on your computer, then you can run the script `scripts/plot.gp` to generate 20 plots in the folder `data` that show how your particle set evolves over time.

Deadline: 16 June 2017, 11:59 am