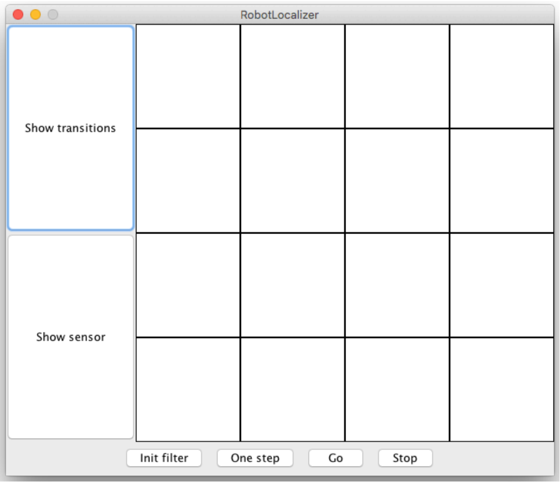
Using the RobotLocalisationViewer:

The RobotLocalisation Viewer assumes a state coding based on triplets (x,y,h), with x being the row, y the column (together the position) and h the heading of the robot in the grid world, with x=0 being the top row, y=0 the leftmost column and h running around the compass from 0 to 3 as NORTH-EAST-SOUTH-WEST. Whatever your state coding looks like internally, you will have to provide the visualisation tool with values according to this assumption.
For the sensor readings it assumes to receive n*m probabilities, one for each position (x, y), to have caused the reading r = (rX, rY) or r = (-1, -1), which means "nothing". See the comments in the EstimatorInterface for details!
Make sure that you provide the data from the observation and transition matrices that you actually **use in your implementation** in the respective methods.

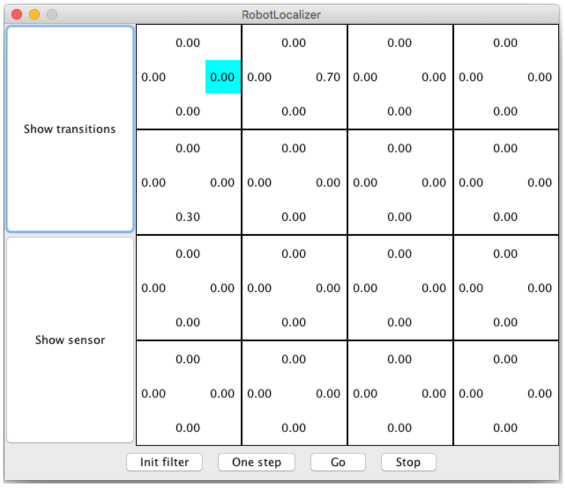
1 Starting:



The viewer starts up with an empty grid according to the dimension specifications retrievable from the EstimatorInterface methods. To get things going you can plug your own "localiser" (i.e., an instance of a class that implements EstimatorInterface) in the initialisation of the example main method (control.Main.java) - or you write your own main …

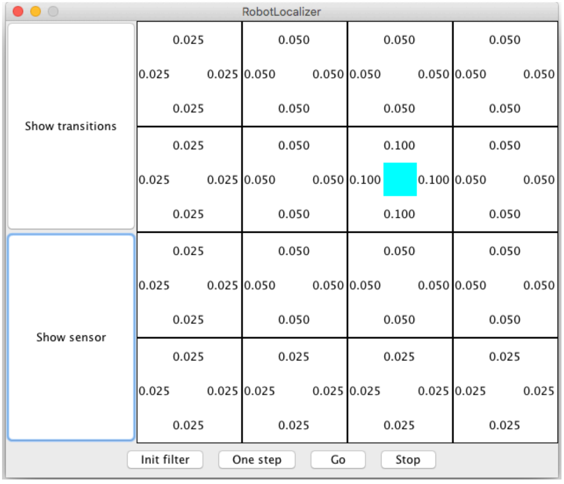
The figure shows the viewer for a 4x4-field grid.
Please observe: The dimensions shown here (4x4) are ONLY an example, easy to fit in the document. Your implementation should consider BIGGER layouts!

2 Checking the matrices



	0.00			0.00			0.00			0.00	
0.00		0.00	0.00		0.70	0.00		0.00	0.00		0.00
	0.00			0.00			0.00			0.00	
	0.00			0.00			0.00			0.00	
0.00		0.00	0.00		0.00	0.00		0.00	0.00		0.00
	0.30			0.00			0.00			0.00	
	0.00			0.00			0.00			0.00	
0.00		0.00	0.00		0.00	0.00		0.00	0.00		0.00
	0.00			0.00			0.00			0.00	
	0.00			0.00			0.00			0.00	
0.00		0.00	0.00		0.00	0.00		0.00	0.00		0.00
	0.00			0.00			0.00			0.00	

2.1 Checking the transition matrix
Clicking on the "Show transitions"-button shows the probabilities for the different poses (x, y, h) to be reached after having been in the given state (marked in cyan). Each further click steps through the states and wraps in the end. The figure shows the probabilities for state (0, 0, EAST) to end up in possible follow-up states. Only (0, 1, EAST) or (1, 0, SOUTH) are possible with p=0.7and p=0.3 respectively.



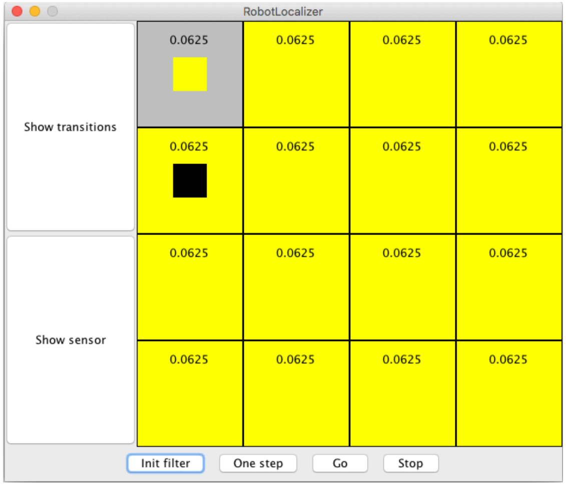
0.025 0.025 0.025 0.025	0.050 0.050 0.050 0.050	0.050 0.050 0.050 0.050	0.050 0.050 0.050 0.050
0.025 0.025 0.025 0.025	0.050 0.050 0.050 0.050	0.100 0.100 0.100 0.100	0.050 0.050 0.050 0.050
0.025 0.025 0.025 0.025	0.050 0.050 0.050 0.050	0.050 0.050 0.050 0.050	0.050 0.050 0.050 0.050
0.025 0.025 0.025 0.025	0.025 0.025 0.025 0.025	0.025 0.025 0.025 0.025	0.025 0.025 0.025 0.025

2.2 Checking the observation matrix

Clicking on the "Show sensor"-button shows the probabilities with which the sensor reports the given position (marked cyan) or "nothing" (all white), when the robot is actually in the other positions. Each further click steps through the sensor readings and wraps in the end. The figure shows the probabilities for sensor reading r = (1, 2) to be caused by the respective state represented as (x, y, h)-pose in the grid.

Rule of thumb for evaluation of your coding: If you stack all n*m+1 grid visualisations of the observation matrices on top of each other, the sum over the probabilities in each "pose-stack" should be exactly1.0 (give or take some rounding errors). The sum over all cells WITHIN one visualisation can in theory be anything between 0.0 and (n*m*4+1)*1.0.

3 Visualising the filtering steps and results

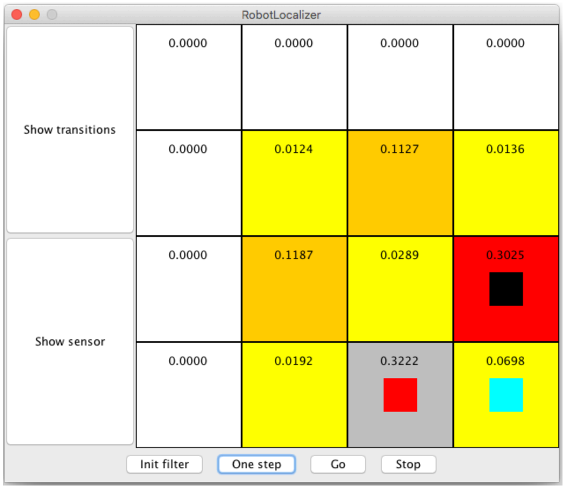


0.0625	0.0625	0.0625	0.0625
0.0625	0.0625	0.0625	0.0625
0.0625	0.0625	0.0625	0.0625
0.0625	0.0625	0.0625	0.0625

3.1 Initialising

Clicking on "Init filter" initialises the viewer / localiser. This step is necessary to get further steps running properly (and it shows you the initial state of the grid). The figure shows the starting position (black) at (1, 0) and the probability distribution (normally equally distributed). A light grey "ring" marker is on the field with highest probability (i.e., the left upper corner, as everything is the same).

Please note: clicking "Init filter" again does not do anything apart from skipping the sensor reading marker in one step - the viewer is currently meant to be started for a new run from scratch.



0.0000	0.0000	0.0000	0.0000
0.0000	0.0124	0.1127	0.0136
0.0000	0.1187	0.0289	0.3025
0.0000	0.0192	0.3222	0.0698

3.2 Stepping through

With "One step" you advance one step in the filtering process (hence the update() method for one step). Black: true position, cyan: sensor reading, light grey: highest probability (only the "first" one encountered is marked). White: assumed impossible (p = 0), yellow: low probability (0.0<p<=0.1), orange: higher probabilities (0.1< p <=0.3), red: "high" (p>0.3). In the upper part of each field the actual probability is shown (truncated to four decimals).

Please note: The numbers shown in the last figure are only **examples** for the visualisation - **do not assume them** as the final result for validation of your results!

3.3 Running continuously

With a click on "Go" you start the loop over the steps (delay according to time parameter for the driver thread). "Stop" interrupts the loop, it is possible to go stepwise again - or loop again.