Technische Universität Berlin | Faculty IV | Robotics and Biology Laboratory

Assignment 4

Group 2_Tue_A

Yufan Dong 462233
Erik Günther 462304
Jingsheng Lyu 398756

Berlin, February 6, 2021

Contents

Student	A	B1	B2	B3	C1	C2	C3	C4	C5	C6
Yufan Dong	x	x	x	x	x	x	x	x	x	x
Erik Günther	x	x	x	x	x	x	x	x	x	x
Jingsheng Lyu	x	x	x	x	x	x	x	x	x	x

A Bayes

Given probabilities:

- $P(z = 42 \mid \text{open}) = \frac{2}{3}$
- $P(z = 42 \mid \neg \text{ open}) = \frac{1}{3}$
- $P(\text{open}) = \frac{3}{5}$

Solution According to Bayes Law

$$P(\text{open} \mid z = 42) = \frac{P(z = 42 \mid \text{open}) \cdot P(\text{open})}{P(z = 42)}. \tag{1}$$

The equation above has two unknowns $P(\text{open} \mid z = 42)$ and $P(z = 42)$ so that it can not be solved immediately. But Bayes Law is also applicable for the event ¬open, leading to

$$P(\neg\text{open} \mid z = 42) = \frac{P(z = 42 \mid \neg\text{open}) \cdot P(\neg\text{open})}{P(z = 42)} \tag{2}$$

$$\Leftrightarrow P(z = 42) = \frac{P(z = 42 \mid \neg\text{open}) \cdot P(\neg\text{open})}{P(\neg\text{open} \mid z = 42)}. \tag{3}$$

The equation above is inserted in equation 1

$$P(\text{open} \mid z = 42) = \frac{P(z = 42 \mid \text{open}) \cdot P(\neg\text{open} \mid z = 42) \cdot P(\text{open})}{P(z = 42 \mid \neg\text{open}) \cdot P(\neg\text{open})} \tag{4}$$

and it is known that the door is either open or ¬open, so that

$$P(\text{open}) + P(\neg\text{open}) = 1 \tag{5}$$

$$\Leftrightarrow P(\neg\text{open}) = 1 - P(\text{open}) = \frac{2}{5}. \tag{6}$$

Also for the given event $z = 42$ the door could only either be open or ¬open, resulting in:

$$P(\text{open} \mid z = 42) + P(\neg\text{open} \mid z = 42) = 1 \tag{7}$$

$$\Leftrightarrow P(\neg\text{open} \mid z = 42) = 1 - P(\text{open} \mid z = 42) \tag{8}$$

Inserting this in equation 4

$$P(\text{open} \mid z = 42) = (1 - P(\text{open} \mid z = 42)) \left(\frac{P(z = 42 \mid \text{open}) \cdot P(\text{open})}{P(z = 42 \mid \neg\text{open}) \cdot P(\neg\text{open})} \right) \tag{9}$$

$$P(\text{open} \mid z = 42) = (1 - P(\text{open} \mid z = 42)) \left(\frac{\frac{2}{3} \cdot \frac{3}{5}}{\frac{1}{3} \cdot \frac{2}{5}} \right) \tag{10}$$

$$\Leftrightarrow \boxed{P(\text{open} \mid z = 42) = \frac{3}{4}} \tag{11}$$

The probability $P(\text{open} \mid z = 42)$, meaning the likelihood that the door is open by the obtained measurement of $z = 42$ is 0.75.

B Mapping with Raw Odometry

B.1 Simple Counting Method

This task is focused solely on mapping, meaning that the position of the robot ($x,\ y,\ \theta$) is given as an input. The *simple counting method* is used to compute an occupancy map. First a blank 2D map m is divided into a grid of cells and the algorithm computes for each cell $m[x, y]$ a believe

$$bel(m[x, y]) = \frac{hits(x, y)}{hits(x, y) + misses(x, y)} \tag{12}$$

that there is in obstacle *occupying* that cell. Based on laser scans the number of hits and misses are counted for each cell. If the laser beam ends at a certain cell, the number of *hits* is increased by 1 and if the laser passes trough a cell then the number of *misses* is increased by 1 for this cell. Hence, all cells linked to a laser (beam) measurement need to be identified. In the fowling the cell at the start and at the end of the laser beam is calculated. Then the method `MappingNode::bresenhamLine(...)` identifies all cells between the start and the end of a laser beam and the believe as shown above can be determined. The following information is given as an input:

- pose of robot $(x_r,\ y_r,\ \theta_r)_{map}$ in map coordinates
- length l_l of laser beam (s. code, `laserScan`)
- angle θ_l of the laser beam in robot frame of reference (s. code, `laserScan`)

In general x_{map} is in direction of map width and y_{map} is in direction of map height. The start of the laser beam is the position of the robot, already given in map coordinates:

$$\begin{bmatrix} x_{l,start} \\ y_{l,start} \end{bmatrix}_{map} = \begin{bmatrix} x_r \\ y_r \end{bmatrix}_{map} \tag{13}$$

The end point of the laser in robots frame of reference is calculated by

$$\begin{bmatrix} x_{l,end,r} \\ y_{l,end,r} \end{bmatrix}_r = \begin{bmatrix} cos(\theta_l) \\ sin(\theta_l) \end{bmatrix} \cdot l_{beam} \tag{14}$$

Now let's compute the end of the laser beam in map coordinates (x,y). Therefore

$$\vec{p}_{l,end,map} = R(\theta_r) \cdot \vec{p}_{l,end,r} + \vec{p}_r \tag{15}$$

$$\begin{bmatrix} x_{l,end} \\ y_{l,end} \end{bmatrix}_{map} = \begin{bmatrix} \cos\theta_r & -\sin\theta_r \\ \sin\theta_r & \cos\theta_r \end{bmatrix} \begin{bmatrix} x_{l,end} \\ y_{l,end} \end{bmatrix}_r + \begin{bmatrix} x_r \\ y_r \end{bmatrix}_{map} \tag{16}$$

Hence, inserting equation 14

$$\begin{bmatrix} x_{l,end} \\ y_{l,end} \end{bmatrix}_{map} = \begin{bmatrix} \cos\theta_r & -\sin\theta_r \\ \sin\theta_r & \cos\theta_r \end{bmatrix} \begin{bmatrix} \cos\theta_l \\ \sin\theta_l \end{bmatrix} \cdot l_{beam} + \begin{bmatrix} x_r \\ y_r \end{bmatrix}_{map} \tag{17}$$

is simplified to

$$\begin{bmatrix} x_{l,end} \\ y_{l,end} \end{bmatrix}_{map} = \begin{bmatrix} \cos(\theta_r + \theta_l) \\ \sin(\theta_r + \theta_l) \end{bmatrix} \cdot l_{beam} + \begin{bmatrix} x_r \\ y_r \end{bmatrix}_{map}. \tag{18}$$

B.2 Screenshots

At the beginning ($t \approx 2s$) the robot is not moving, resulting in a good mapping quality, see figure 2. In figure 3 it can be observed, that the robot moved from its initial position, so a larger area of the map is discovered. The mapping works also for this translational movement. But as shown in figure 4 a changed orientation leads to a large mapping error.

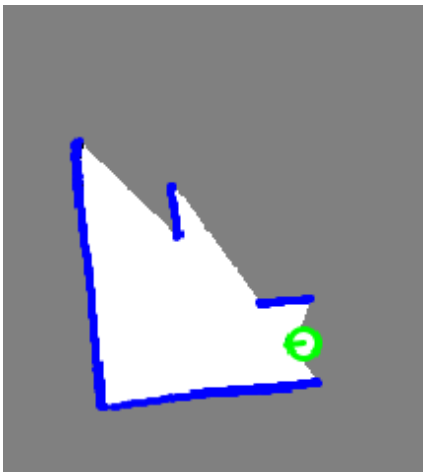


Figure 2: mapping at $t \approx 2s$

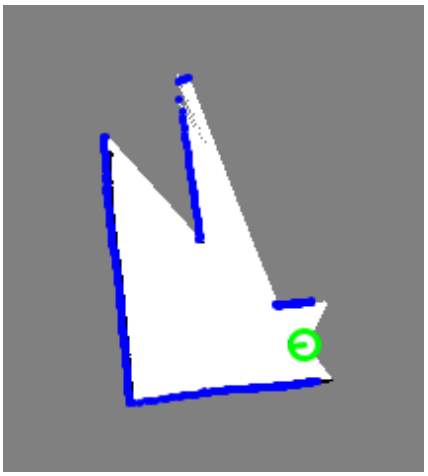


Figure 3: mapping at $t \approx 8s$

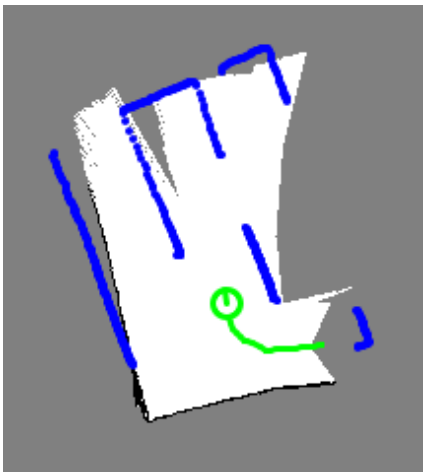


Figure 4: mapping at $t \approx 20s$

B.3

The mapping differs from reality, because we are focusing only on mapping in this task. The pose of the robot is given, but there is no localization.

C Monte Carlo Localization: Particle Filter

In this chapter of the report the particle filter that is implemented to solve the localization problem of this task is described. The particle filter is a non-parametric Bayes Filter. Each particle is a hypothesis of a state of the robot at a certain time. First a discrete set of particles (here 1000) is introduced on the map using the same weight/importance factor for each particle meaning initially it is equally likely that the robot is in that particular poses. section C.1 and C.2. Then based on motion section C.3 and sensor data section C.4 this importance factor is updated for each particle. The final step of each iteration is re-sampling as described in section C.5. This step replaces less likely particles by more likely particles.

C.1 Initial sample - uniform distribution

We will get the uniform distribution for the particle filter. Meaning that the robot could initially be equally probable at all locations on the map, there is no initial information/indication or *best hypothesis* to enclose the position/orientation of the robot. Therefore a large number of particles (1000) are distributed uniformly across the map, by utilizing the method `Util::uniformRandom`. This method generates a random number (uniformly distributed) for each

$$x \in [0,\ \text{map width}] \tag{19}$$
$$y \in [0,\ \text{map height}] \tag{20}$$
$$\theta \in [-\pi, +\pi]. \tag{21}$$

The importance factor/weight is initialized by

$$w_i = \frac{1}{n} \tag{22}$$

with i: index of particle
w_i: weight of every particle
n: total number of particles

C.2 Initial sample - Gaussian distribution

While in section C.1 there was no assumption or knowledge of the initial location of the robot, here the pose (x, y, θ) is given each by a mean and a standard deviation. Using this information the set of particles is generated by using the method `Util::gaussianRandom`. The importance factor/weight is again initialized by

$$w_i = \frac{1}{n} \tag{23}$$

with i: index of particle
w_i: weight of every particle
n: total number of particles

C.3 Odometry Motion Model

As measured by the robot's internal odometry, the odometry model uses the relative motion information. In the time interval $(t-1, t]$, the robot advances from a pose x_{t-1} to pose x_t.

The odometry reports back to us a related advance from $\bar{x}_{t-1} = (\bar{x}, \bar{y}, \bar{\theta})^T$ to $\bar{x}_t = \left(\bar{x}', \bar{y}', \bar{\theta}'\right)^T$.

Here the bar indicates that these are odometry measurements embedded in a robot-internal coordinate whose relation to the global world coordinates is un-known. The key insight for utilizing this information in state estimation is that the relative difference between $\bar{x}_{t-1}$ and $\bar{x}_t$, under an appropriate definition of the term "difference", is a good estimator for the difference of the true poses x_{t-1} and x_t.

It is transformed into a sequence of three steps (see below) to extract relative odometry u_t:

- motion (translation) δ_{trans}
- rotation δ_{rot1}
- rotation δ_{rot2}

The variables from α_1 to α_4 are robot-specific parameters that specify the noise in robot motion.
Step 1 - Based on the odometry information the *true* motion is

$$\delta_{trans} = \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2} \tag{24}$$

$$\delta_{rot1} = \operatorname{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta} \tag{25}$$

$$\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{\text{rot 1}} \tag{26}$$

so that the measured motion can be determined by corrupting the true values above with noise normal distribution - using method `Util::gaussianRandom` - (**Step 2**)

$$\hat{\delta}_{rot1} = \delta_{rot1} + \varepsilon_{\alpha_1|\delta_{rot1}|+\alpha_2|\delta_{trans}|} \tag{27}$$

$$\hat{\delta}_{trans} = \delta_{trans} + \varepsilon_{\alpha_3|\delta_{trans}|+\alpha_4|\delta_{rot1}+\delta_{rot2}|} \tag{28}$$

$$\hat{\delta}_{rot2} = \delta_{rot2} - \varepsilon_{\alpha_1|\delta_{rot2}|+\alpha_2|\delta_{trans}|} \tag{29}$$

Step 3 - Concluding, the new position of each particle can be computed

$$x' = x + \hat{\delta}_{trans} \cdot \cos(\theta + \hat{\delta}_{rot1}) \tag{30}$$

$$y' = y + \hat{\delta}_{trans} \cdot \sin(\theta + \hat{\delta}_{rot1}) \tag{31}$$

$$\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2} \tag{32}$$

C.4 Likelihood Field

In this section the model of *Likelihood Fields for Range Finders* are described. First in task C.4.1 the `likelihoodField` is prepared. Then in task C.4.2 the importance weight is updated and normalized.

C.4.1 Precompute the likelihood field for a given map in ParticleFilter::setMeasurementModelLikelihoodField

In this step, we should firstly create a likelihood field. To calculate p_{hit} (s. below) we first calculate the distance map, that stores the distance to the nearest occupied cell using the

method `ParticleFilter::calculateDistanceMap`. Further details of the implementation are shown in the method `ParticleFilter::setMeasurementModelLikelihoodField`. The main formula is

$$likelihood_i = log(p_{hit} \cdot z_{hit} + p_{random} \cdot z_{random}) \tag{33}$$

with

$$p_{hit}(z_t \mid x_t, m) = \varepsilon_{\sigma_{hit}}(dist) \tag{34}$$

described by a zero mean Gaussian that includes the sensor noise and is based on the distance map. Further the random distribution p_{rand} for random noise and the weights z_{hit} and z_{rand} are utilized.

C.4.2 The correction step of the particle filter in ParticleFilter::likelihoodFieldRangeFinderModel

Now the *likelihood field* that was calculated above is utilized to determine the importance weight. Assuming independence between the noise in different sensor beams the model multiplies the individual values of $p(z_t^k|x_t, m)$. The algorithm loops over every 5th laser beam (k) of the laser scan at a particular time t:

$$p(z_t \mid x_t, m) = \prod_{k=1}^{K} p(z_t^k \mid x_t, m) \tag{35}$$

In other words the likelihood of the measurement (laser scan) given the map and the position is computed. This is done for each particle.

The laser ends at cell $m(x, y)$ that is calculated by

$$x_{z_t^k} = x + z_t^k \cos(\theta + \theta_{k,\,\text{sens}}) \tag{36}$$

$$y_{z_t^k} = y + z_t^k \sin(\theta + \theta_{k,\,\text{sens}}) \tag{37}$$

where x and y are given by the position of the particle. At the end of the iteration, when all importance weights are calculated, they are normalized.

C.4.3 Screenshot of the likelihood field

Figure 5 shows the original map and the figure 6 illustrates the likelihood field map.

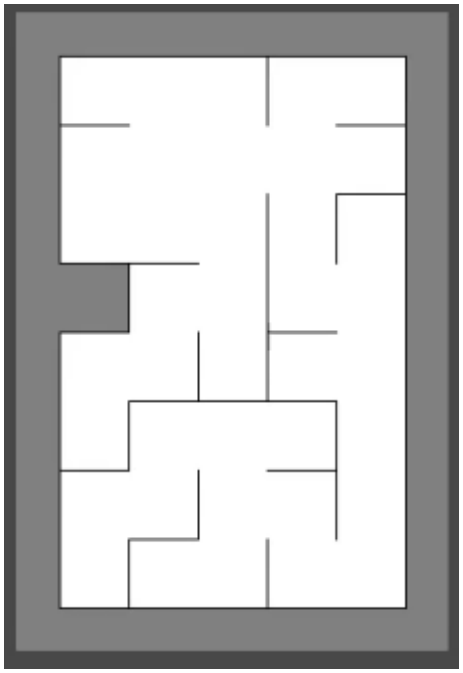


Figure 5: Screenshot for the original map

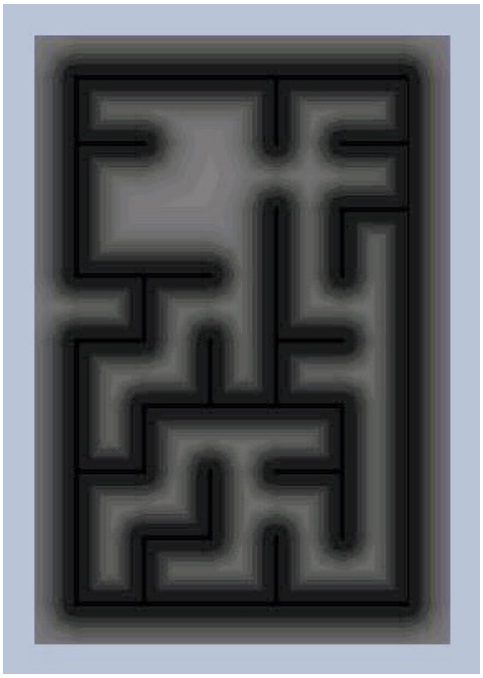


Figure 6: Screenshot for the likelihood field map

C.5 Resampling

In the previous section the *importance factor* $w_t^{[i]}$ of each particle i at the time step t was determined, which is mathematically

$$w_t^{[i]} = p(z_t \mid x_t^{[i]}) \tag{38}$$

or in other words, it is the probability of a measurement given the position/orientation $x_t^{[i]}$. The final step of this iteration of the particle filter is the re-sampling. As described in the training stochastic universal sampling is utilized. Details of this algorithm were described in the training session. The algorithm is implemented in `ParticleFilter::resample` based on the given pseudo-code.

C.5.1 Stochastic universal sampling

In the following only the general concept is described to give an broad overview. This re-sampling can been seen as similar to the survival of the fittest-concept. Initially each particle has the same weight, but as soon as there is additional information such as laser measurements, there will be particles that are more likely to describe the the real pose of the robot on the map and others that are less likely. Hence, the update of the *importance factor* of each particle. The goal to localize the robot on the map can be achieved by replacing less likely particles by more likely ones. Hence, the fittest particles are more likely the parents for the next generation or sample so that the location of the robot on the map is narrowed down sample by sample. The new generation is then weighted initially the same and the algorithm proceeds with the next time step.

C.5.2 Best Hypothesis

The best hypothesis of the robot's position is the particle with the highest weight. During our testing, this could lead to jumps in the calculated path, especially if there are several similar probable locations, let's call them particle swarms, and due to a new measurement a different location/area of the map is rated best hypothesis, so that the robot jumps to this new best location.

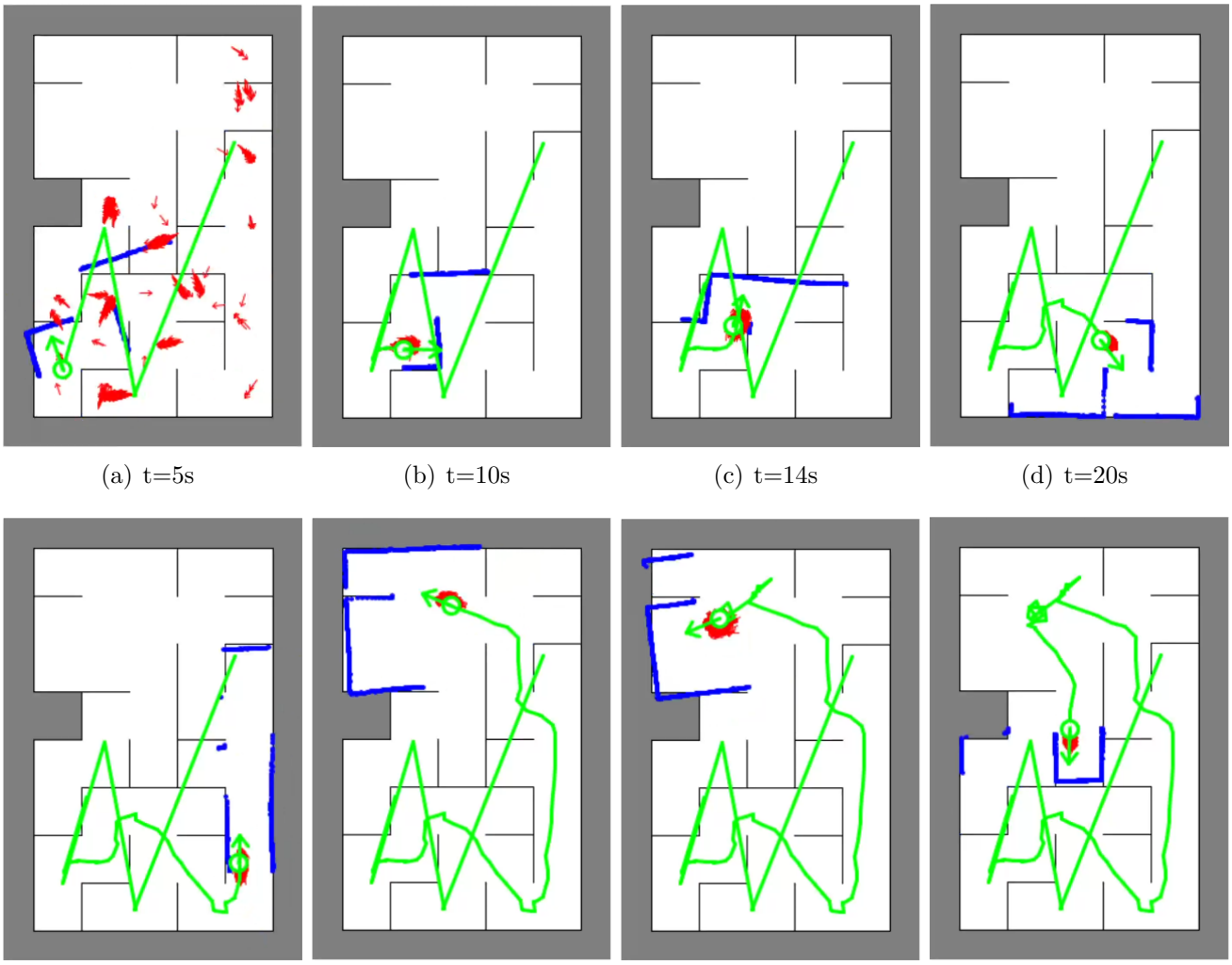
C.6 Result

C.6.1 Global Localization

For this task the initial sampling is a uniform distribution across the entire map, as implemented in task C.1. The initial position of the robot is of course unknown and there are several locations that look similar to the laser measurements. Meaning that there are several different particle swarms at the beginning. Then from time step to time step new information eliminated unlikely particle swarms and the real positions of the robot is obtained.

As shown in figure 6, at the beginning there were several particle swarms at some corners with similar structure. The robot could not tell which swarm should be the correct one. The "Z"-shaped line was not the real path of the robot, but the selection of the best hypothesis based on the current data.

After moving for 5s, the correct swarm was determined based on the new sensed data, which means the localization was successful.



(a) t=5s (b) t=10s (c) t=14s (d) t=20s

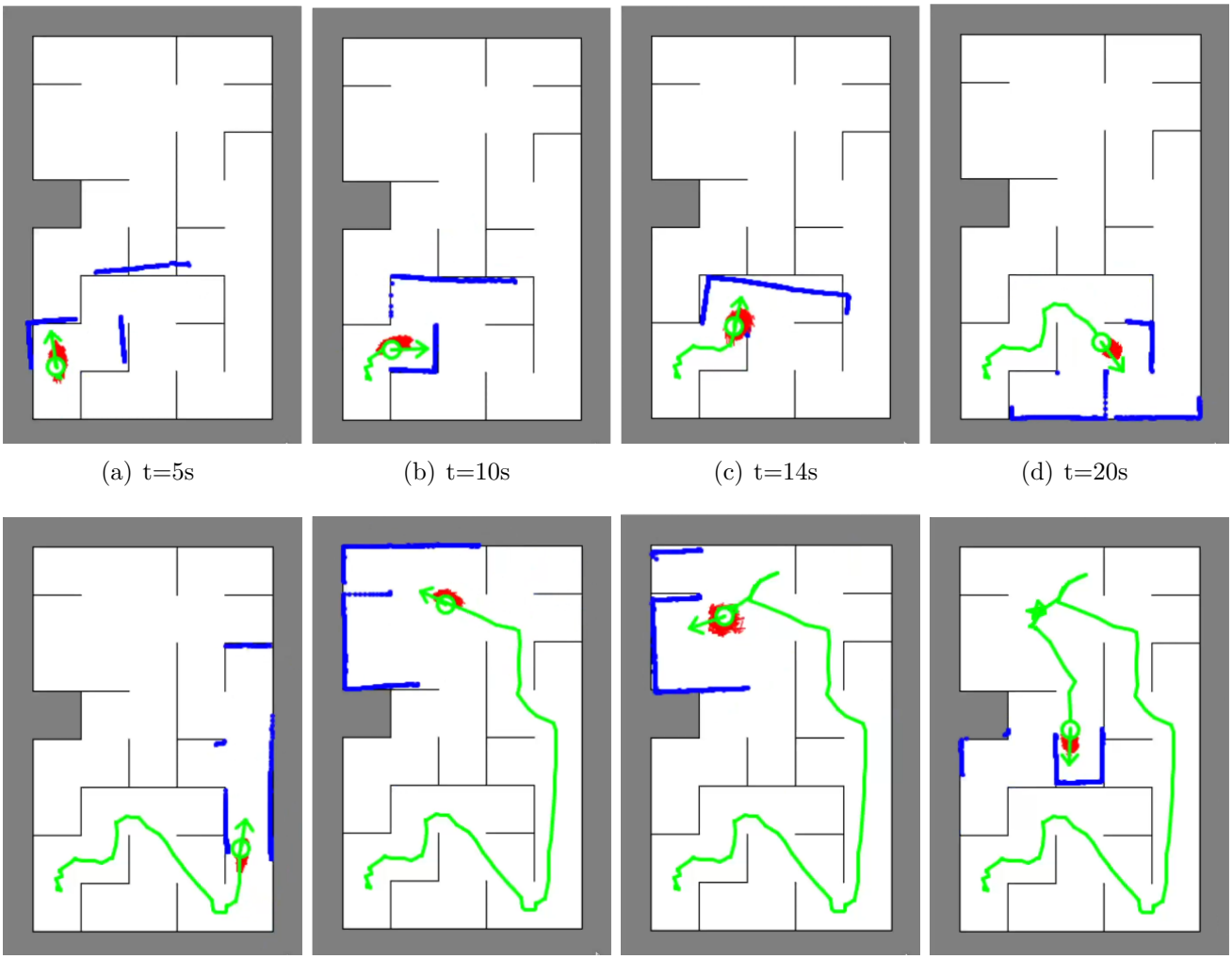
(e) t=30s (f) t=42s (g) t=60s (h) t=75s

Figure 6: Initial with Uniform Distribution

C.6.2 Tracking

While in task C.1 the initial position was completely unknown and thus uniformly distributed across the entire map, in this task the initial position is assumed and the robot can be track very well. The upside is, if the initial position is assumed wrong, all particles are around this position, so that there is no way to correct the location to another more distant point on the map.

As shown in figure 7, if the (probable) initial position of the robot was already known, the localization would be completed quickly.



(a) t=5s (b) t=10s (c) t=14s (d) t=20s

(e) t=30s (f) t=42s (g) t=60s (h) t=75s

Figure 7: Initial with Normal Distribution

References

1. THRUN, S. et al.: *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005. ISBN 0262201623 9780262201629.