Particle Filters

Alfredo Weitzenfeld

Particle Filters

- The goal of localization is to estimate the robot state *s*.
- A Particle Filter or Monte Carlo Localization filter is a Bayesian based filter that samples the complete workspace by a weight function derived from the belief distribution of the previous stage
- Each particle represents a belief or estimate of the robot state using random samples.

Particle Filters

- Upon robot movement, the particle state *s* is updated based on the motion model estimation.
- Upon robot sensing, the particle state *s* is updated based on the measurement model estimation.
- The particle state distribution is then resampled according to their likelihood.

Particle Filters

- S_t are the samples of a posterior particle distribution
$$S_t = s_t^{[1]}, s_t^{[2]}, \dots s_t^{[m]}, \dots. s_t^{[M]}$$
- Each particle $s_t^{[m]}$ (with $1 \leq m \leq M$) is an instantiation of each particle state at time t.
- M is the total number of particles, e.g. M = 1,000.
- Each particle $s_t^{[m]}$ is a hypothesis of what the true robot state may be at time t.
- The goal is to approximate the posterior belief $bel(s_t)$ by the set of particles S_t.

Particle Filters

- The algorithm computes a state $s_t^{[m]}$ for time each particle at time t based on particle $s_{t-1}^{[m]}$ and control u_t:

$$s_t^{[m]} \sim p(\, s_t^{[m]} \,|\, s_{t-1}^{[m]}, u_t \,)$$

- The resulting sample is indexed by m, indicating that it is generated from the m-th particle in S_{t-1}.
- This step involves sampling from the state transition distribution $p(s_t^{[m]} \,|\, s_{t-1}^{[m]}, u_t)$
- The set of particles obtained after M iterations is the particle filter's representation of $\overline{bel}(s_t^{[m]})$.
- Particle predictions are compared with actual measurements.

Particle Filters

- The algorithm calculates for each particle $s_t^{[m]}$ an *importance factor*, denoted by $w_t^{[m]}$
- Importance factors are used to incorporate measurement z_t into the particle set.
- The importance factor is the probability of observing measurement z_t for particle $s_t^{[m]}$

$$w_t^{[m]} = p(z_t \mid s_t^{[m]})$$

- $w_t^{[m]}$ is interpreted as the weight of a particle.
- The set of weighted particles represents the Bayes filter posterior $\mathrm{bel}(s_t^{[m]})$.

Particle Filters

- Resampling transforms a set of M particles into another particle set of the same size.
- Resampling is based on weight in order to create a new distribution.
- Before resampling, particles are distributed according to $\overline{\mathrm{bel}}(s_t^{[m]})$.
- After resampling, particles are distributed according to the posterior

$$\mathrm{bel}(s_t^{[m]}) = \eta\, p(z_t \mid s_t^{[m]})\, \overline{\mathrm{bel}}(s_t^{[m]})$$

- Resampling refocuses the particle set to regions in state space with highest posterior probability.

Particle Filter

- **Prediction**: A prediction for each particle $\overline{\text{bel}}(s_t^{[m]})$ provides the estimated probability of the particle being in state $s_t^{[m]}$ after control u_t. The prediction is denoted by:

$$\overline{\text{bel}}(s_t^{[m]}) = \text{p}(s_t^{[m]} | u_t, s_{t-1}^{[m]})$$

- **Measure (Importance Factor)**: A measurement z_t probability is computed for each particle in state $s_t^{[m]}$ to calculate its importance factor denoted by:

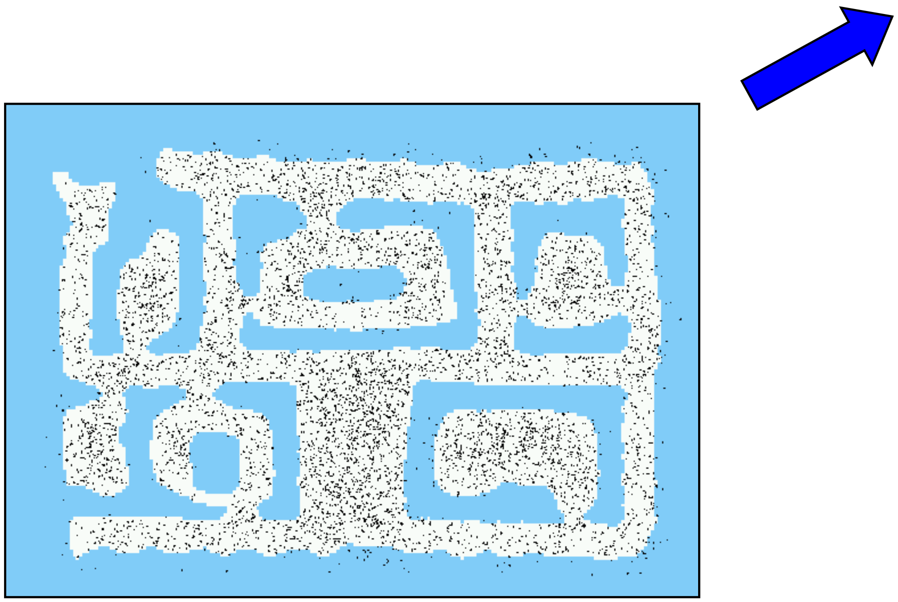
$$w_t^{[m]} = \text{p}(z_t | s_t^{[m]})$$

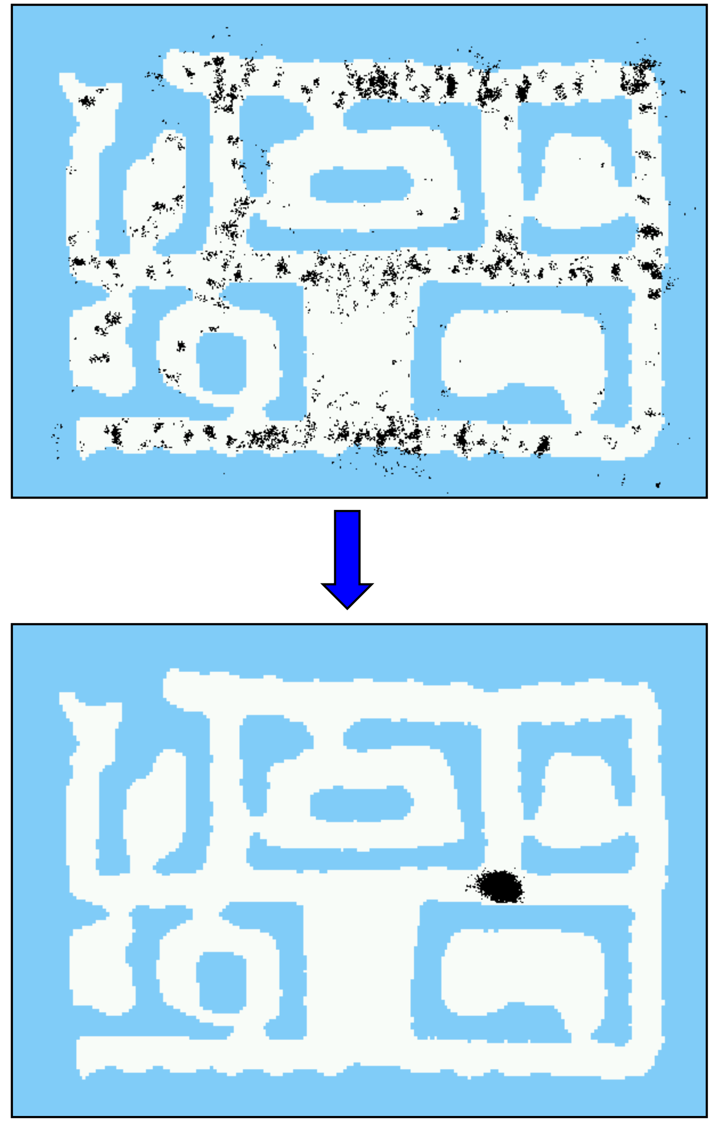
- **Update (Resampling)**: Particle resampling is performed to update the belief of the state based on the measurement importance factor and motion prediction:

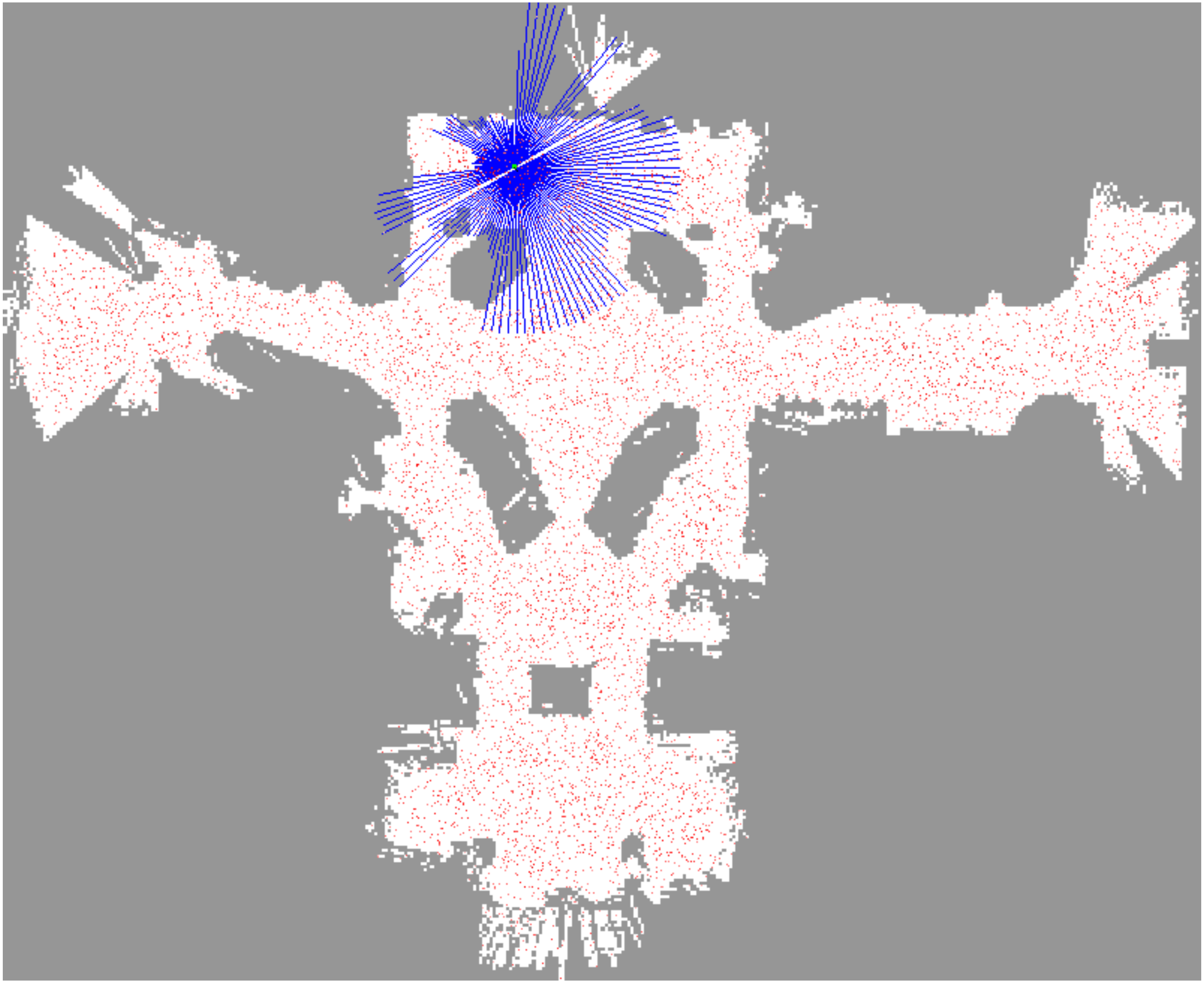
$$\text{bel}(s_t^{[m]}) = \text{p}(s_t^{[m]} | z_t) = w_t^{[m]}\, \overline{\text{bel}}(s_t^{[m]})$$

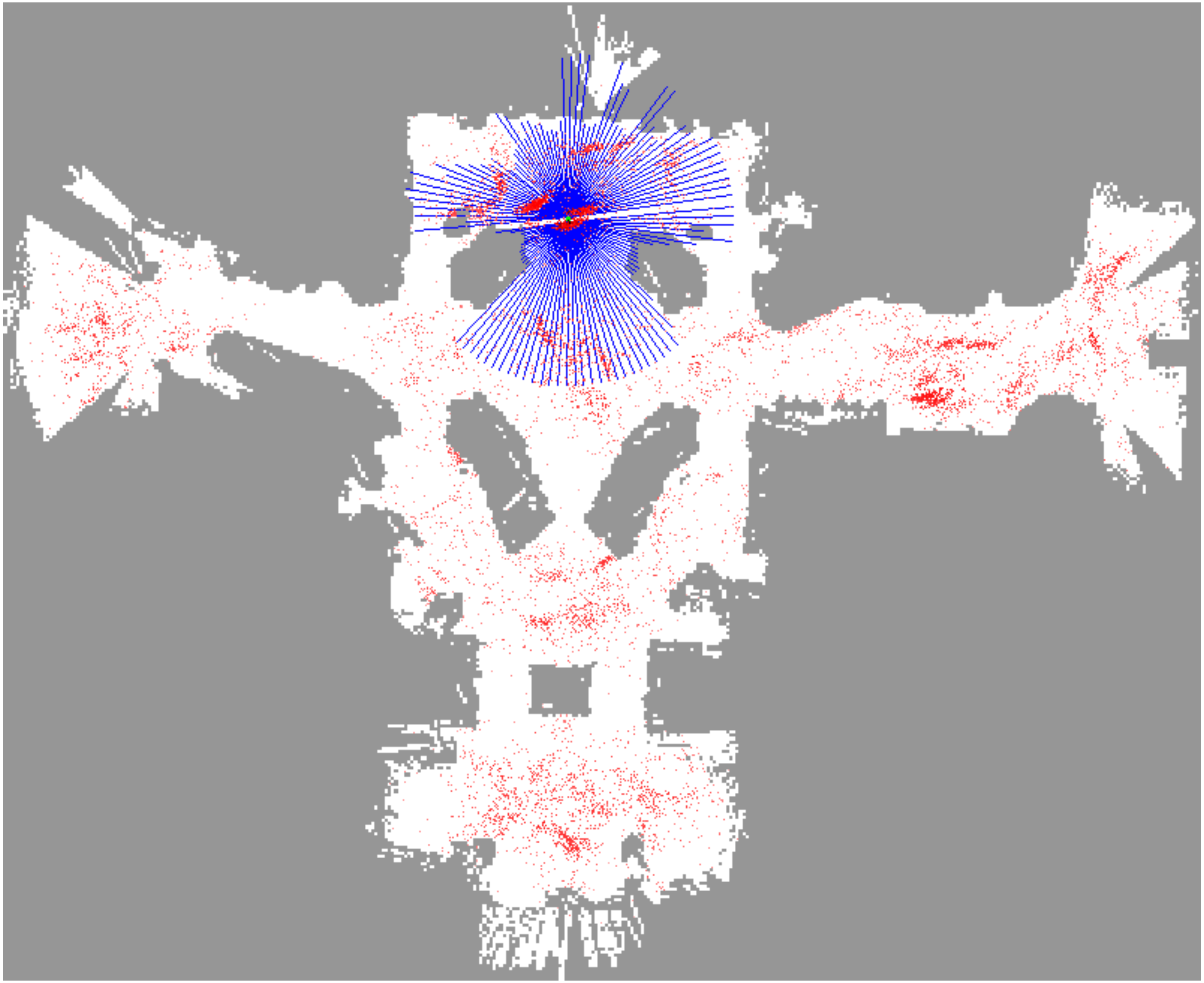
Particle filter

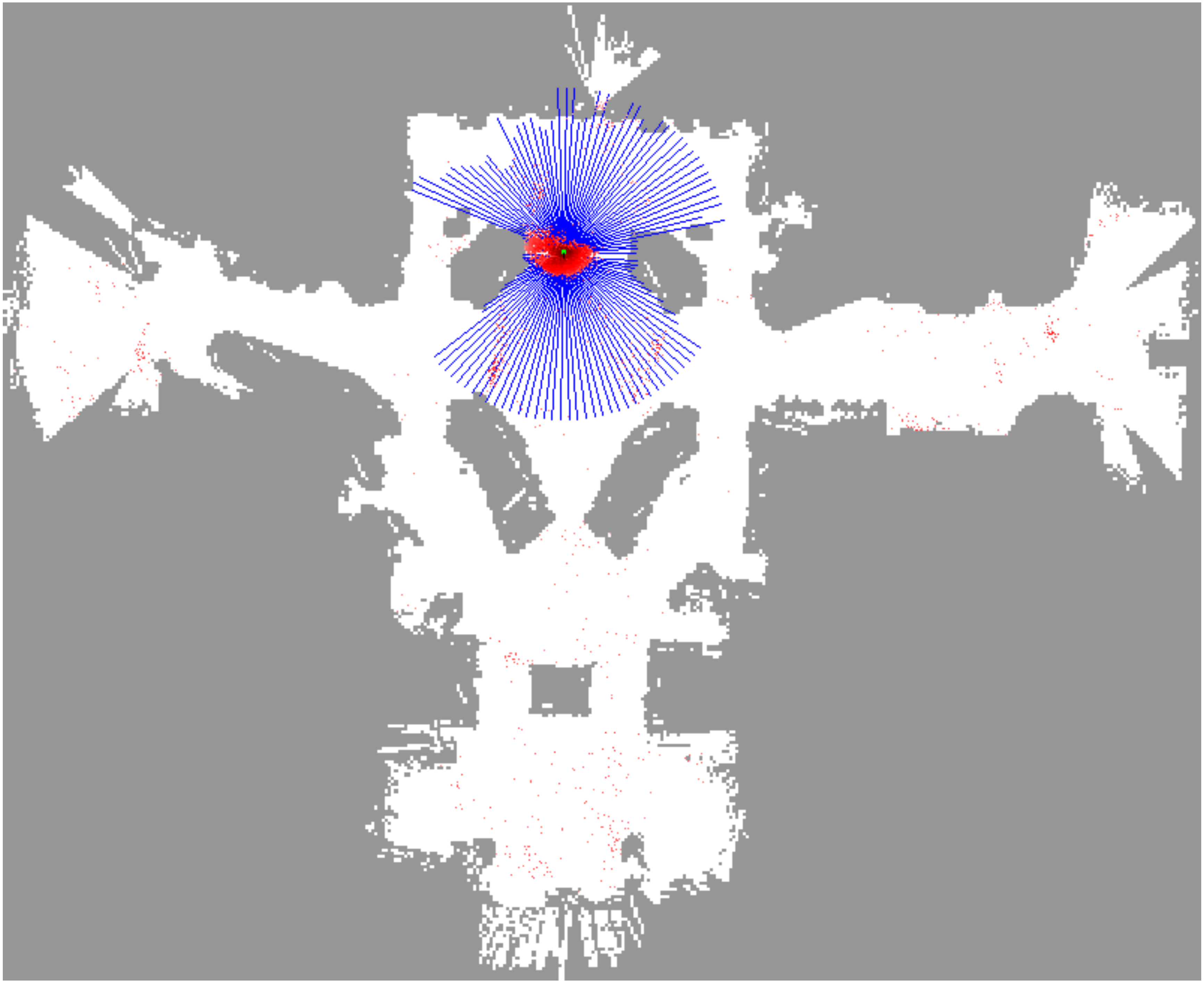
Particle representation of the belief:

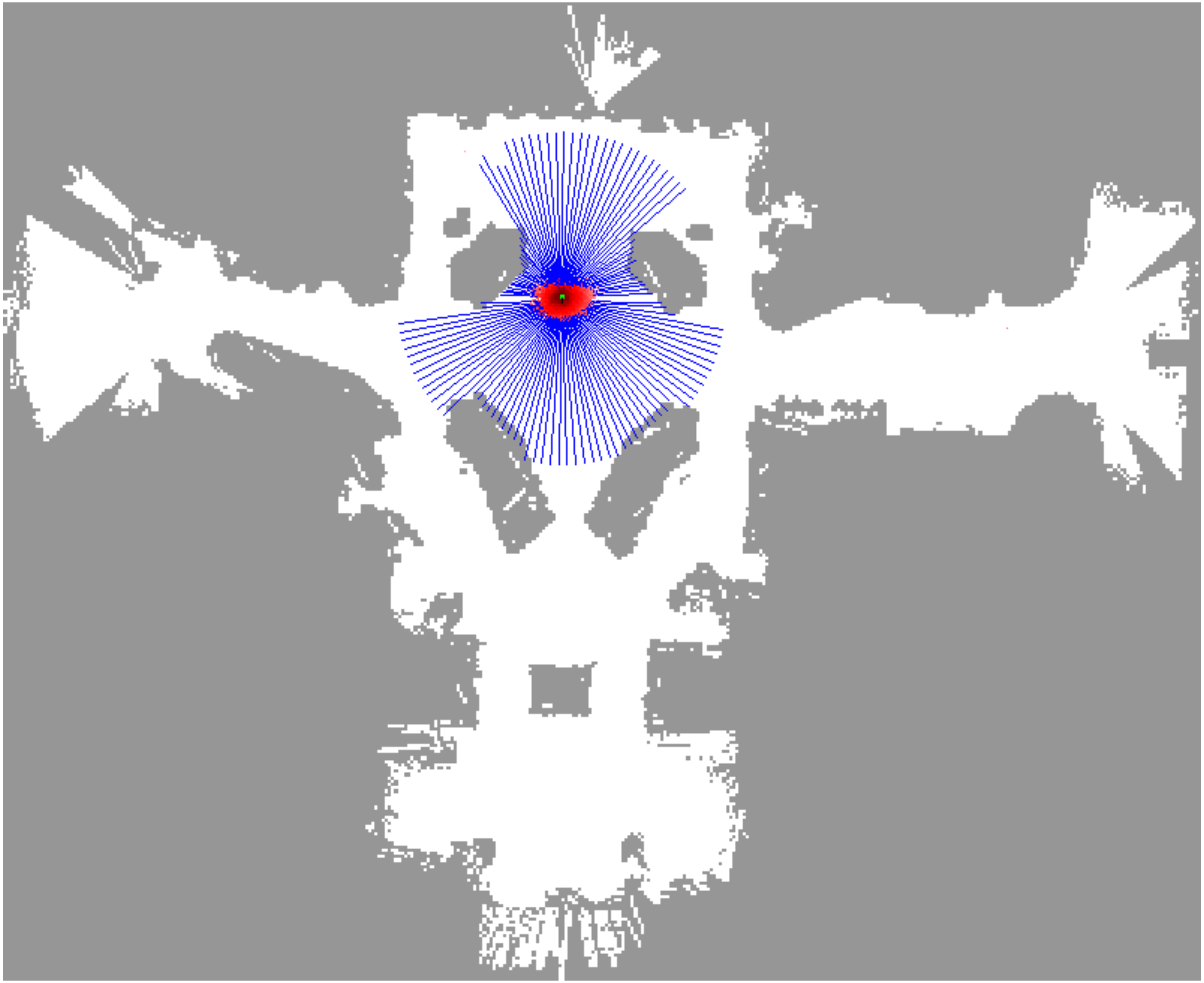


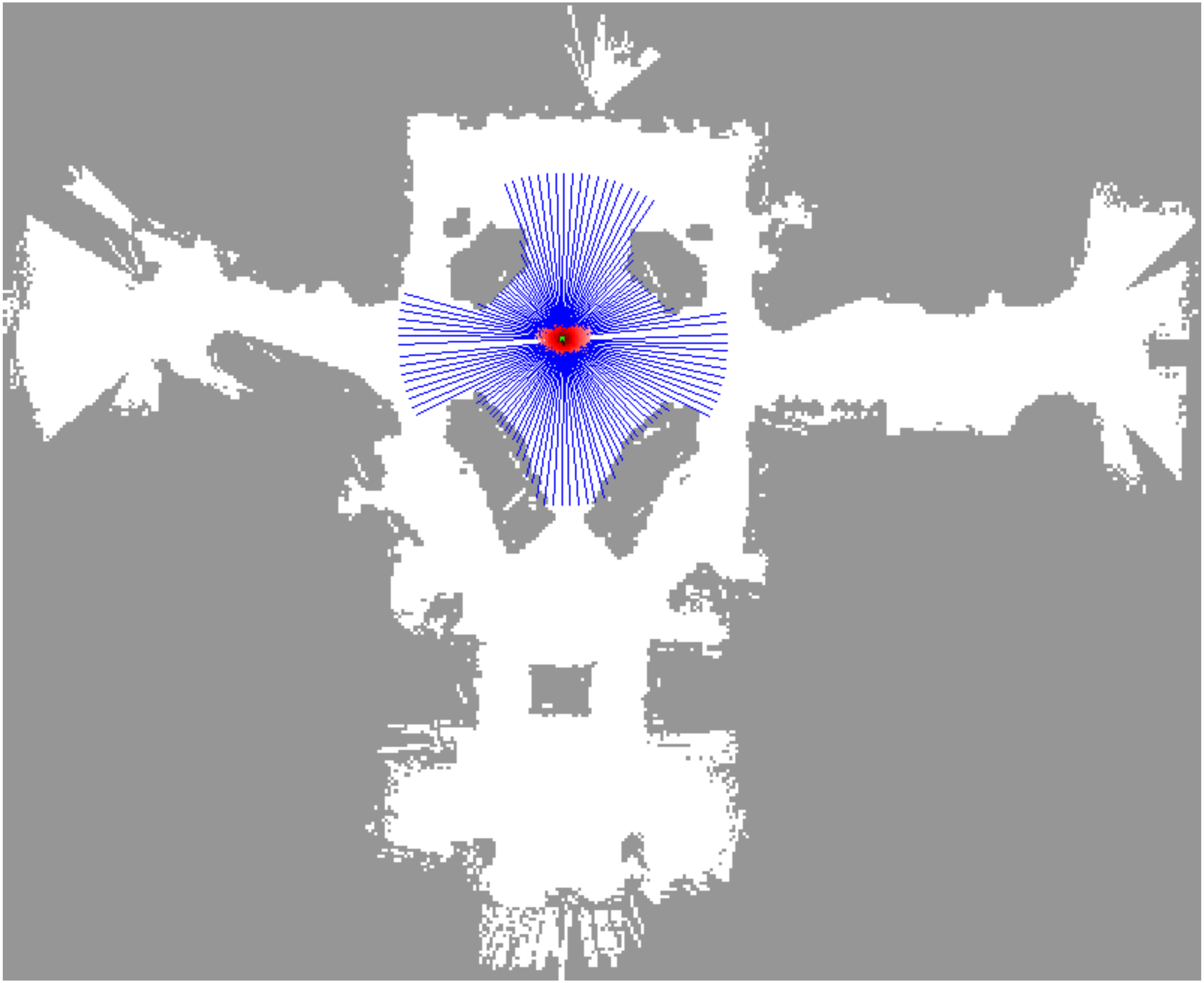


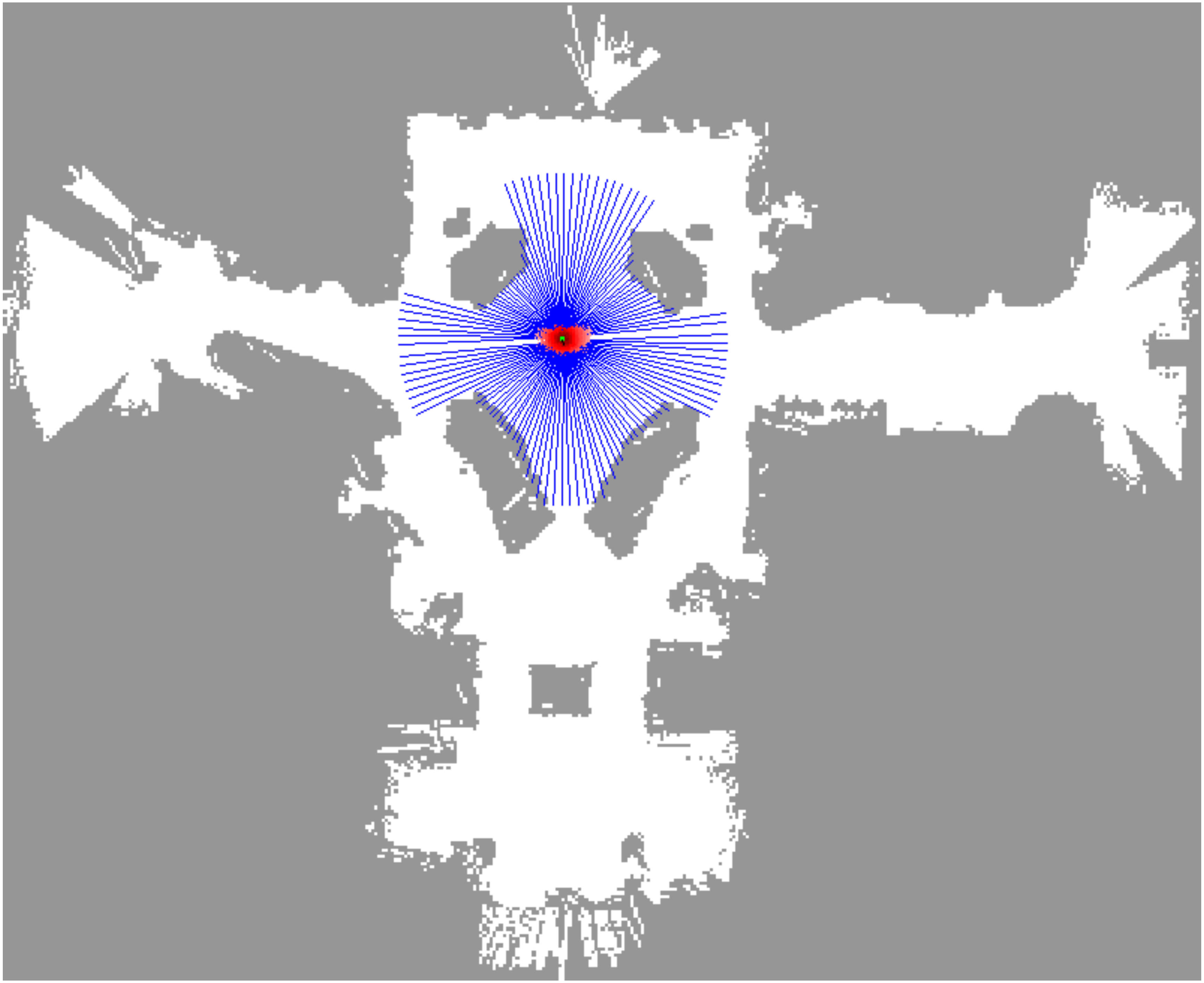




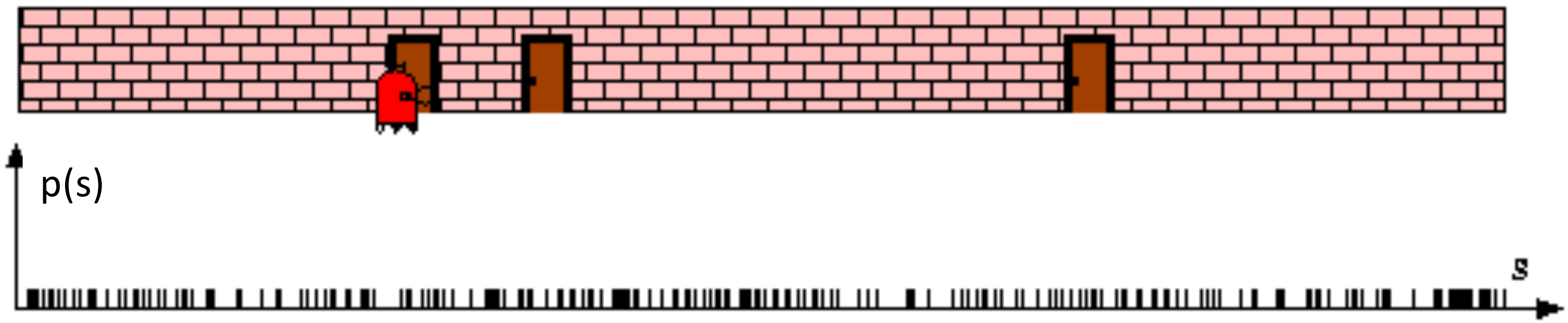






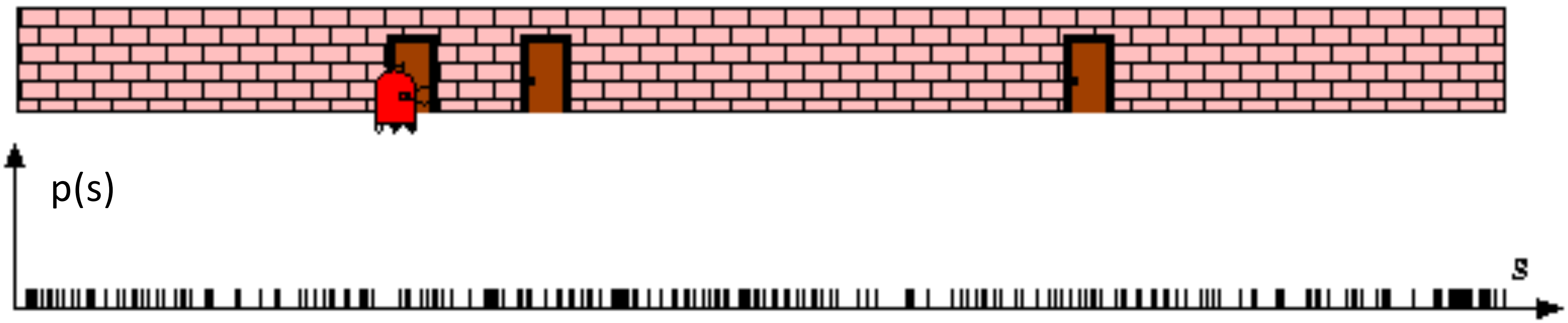


Initialization



Initial state with particles drawn at random and uniformly over the entire pose space

Measurements & Importance Factor

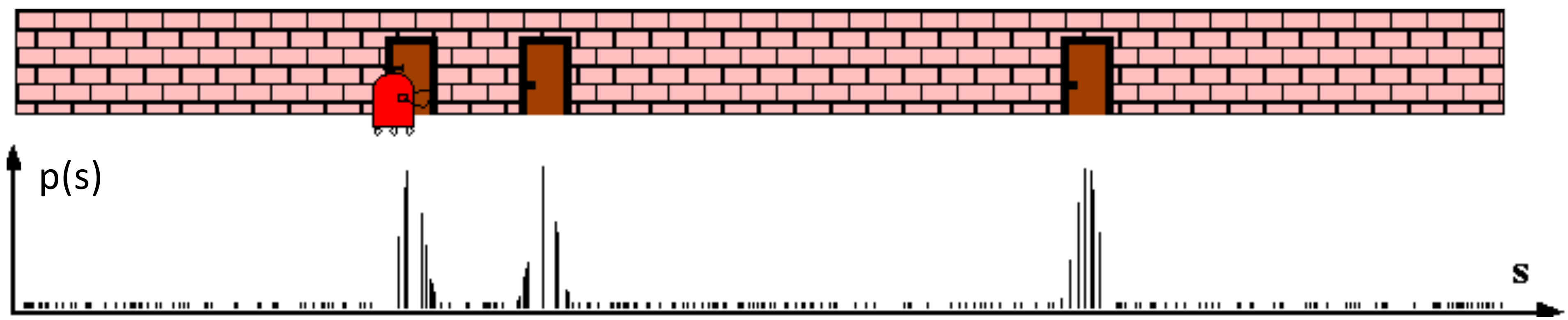


Particles drawn at random and uniformly over the entire state space

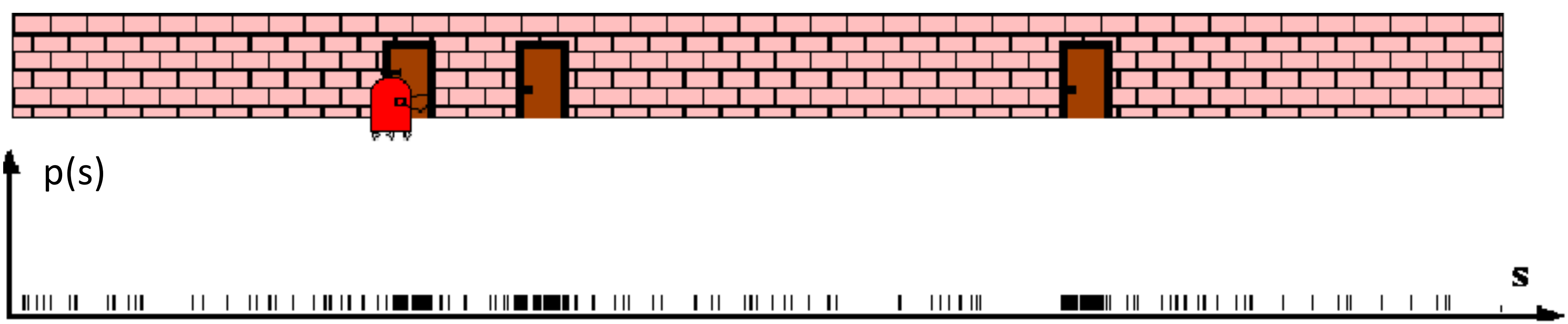
After robot senses the door, MCL assigns importance factors to each particle

Particles near the three likely places have an increased weight or importance factor

Resampling

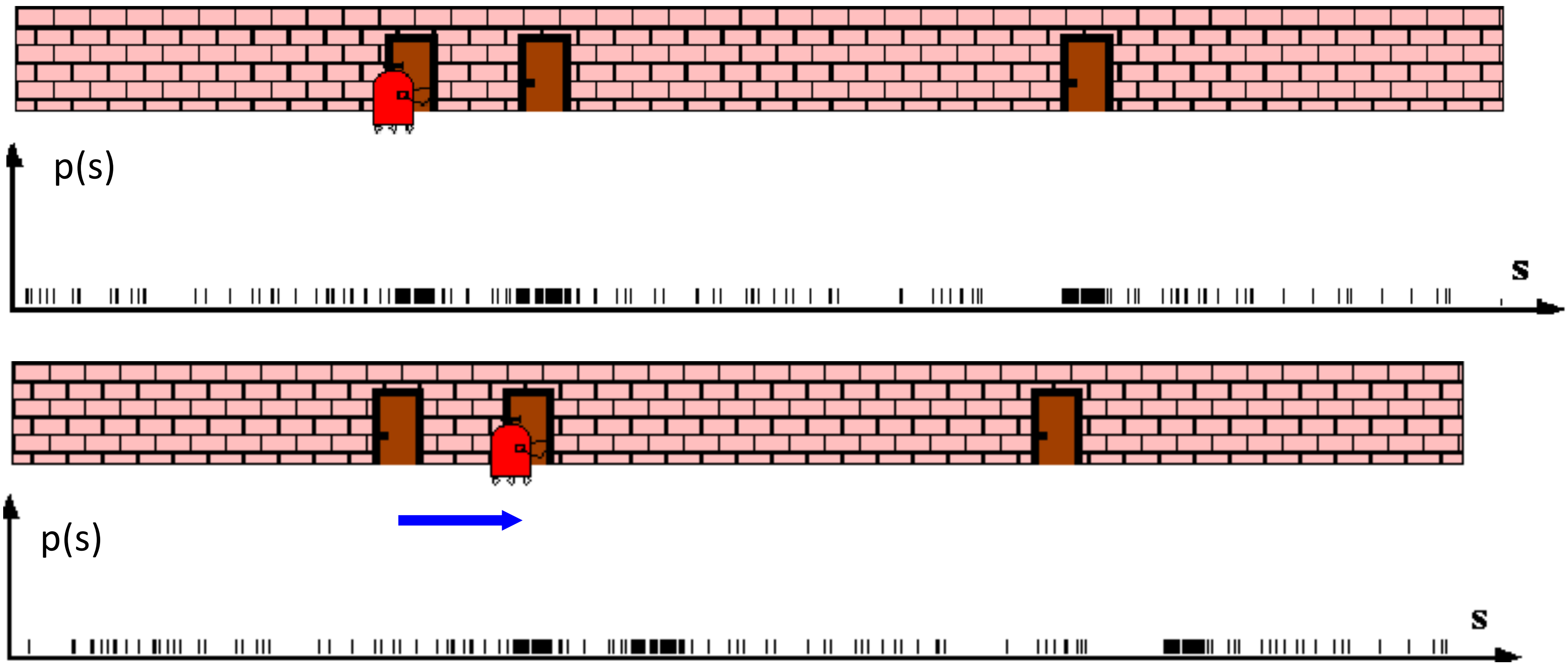


Particles near the three likely places have an increased weight or importance factor



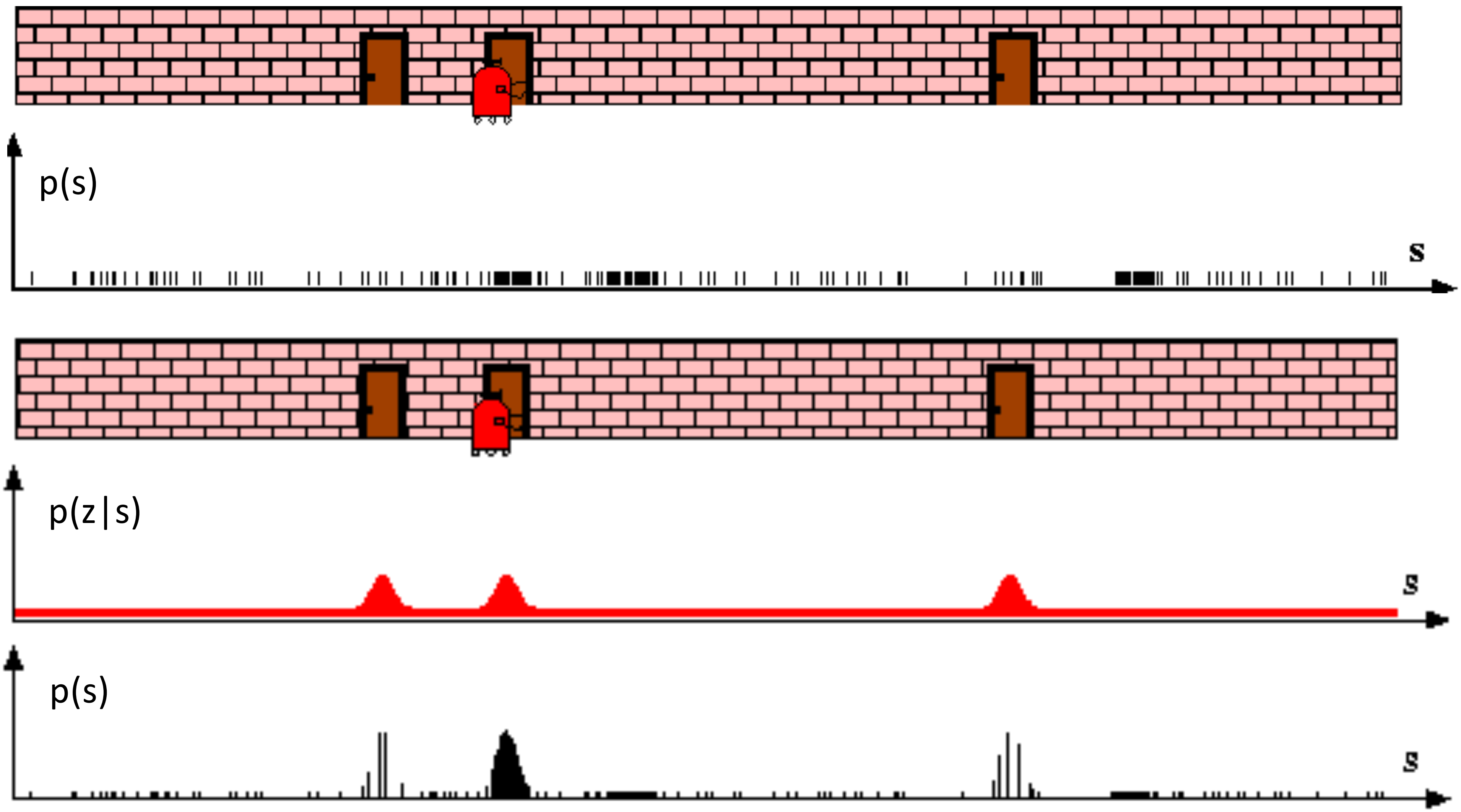
Resampling or importance sampling leads to new particle set with uniform importance weights, but with an increased number of particles near the three likely places

Motion Prediction



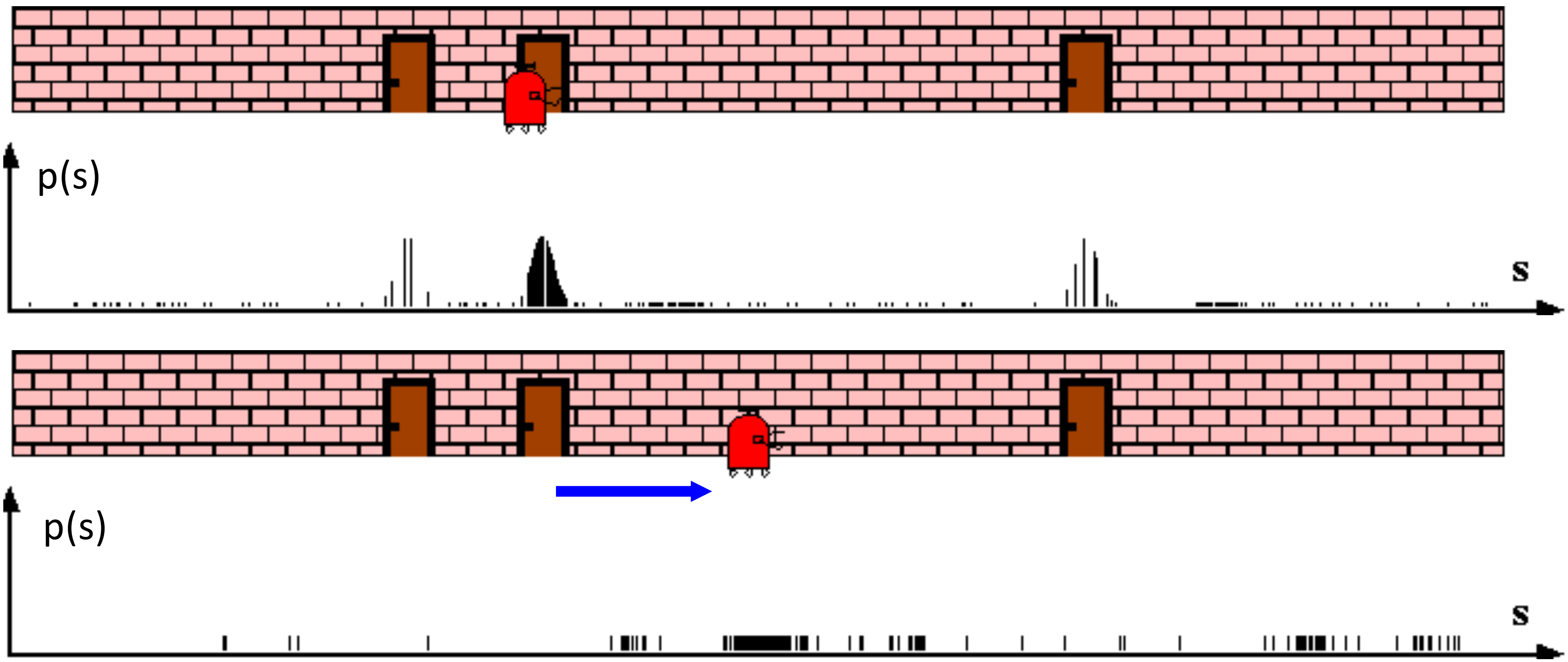
As robot moves the motion probability distribution is applied to all particles

Measurements & Importance Factor



New measurements assign non-uniform importance weights to the particle sets, most of the cumulative probability mass is centered on the second door

Resampling & Motion Prediction



Further motion leads to another re-sampling step, and a step in which a new particle set is generated according to the motion model

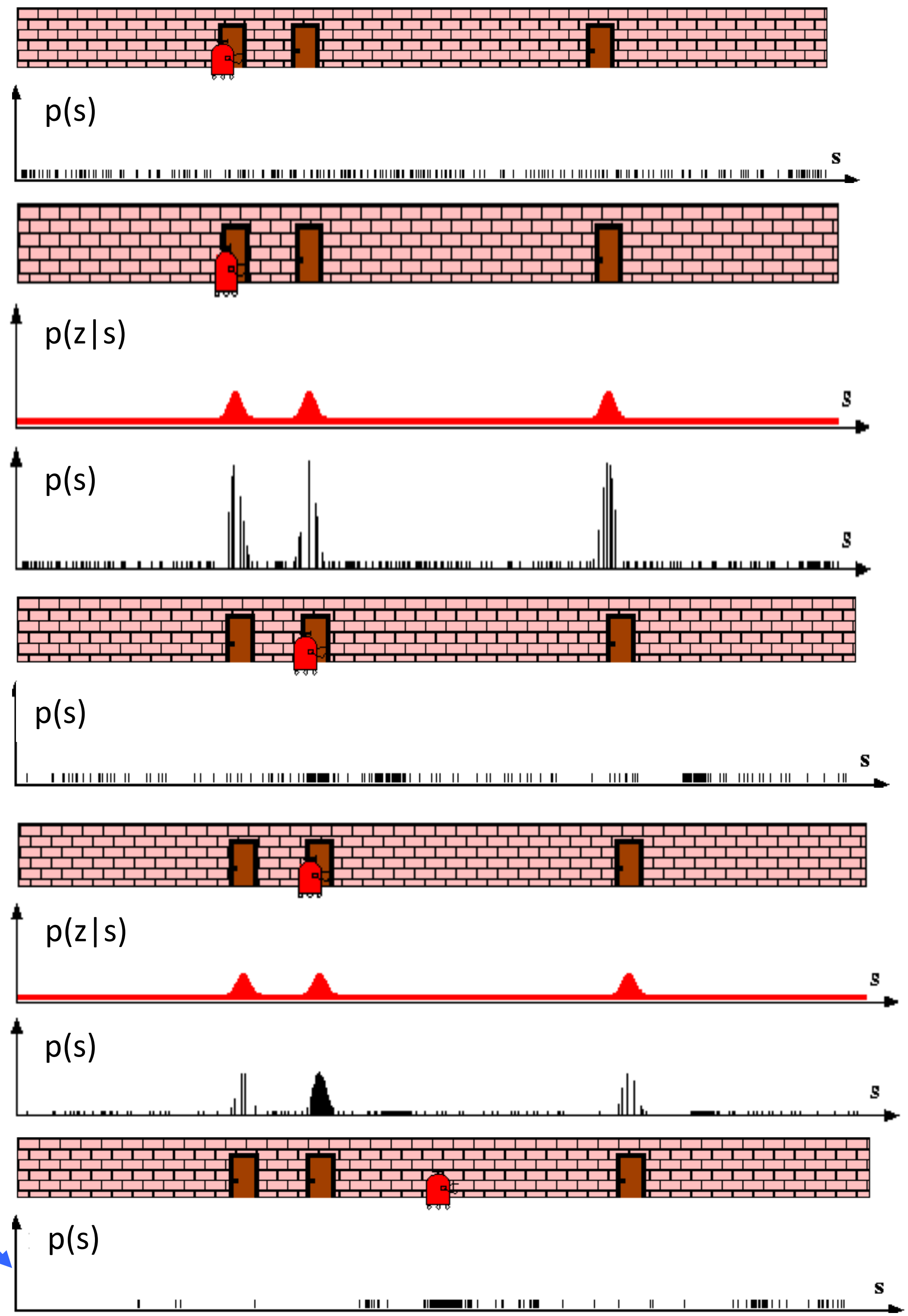
Particle Filter

Initialization

Measurements

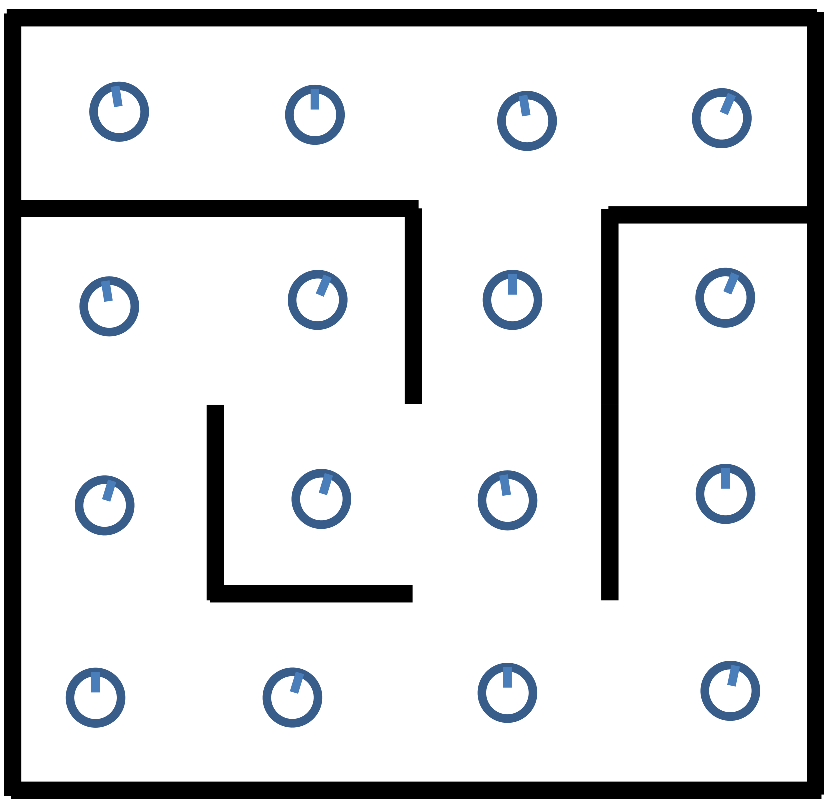
Importance Factor

Resampling & Motion Update

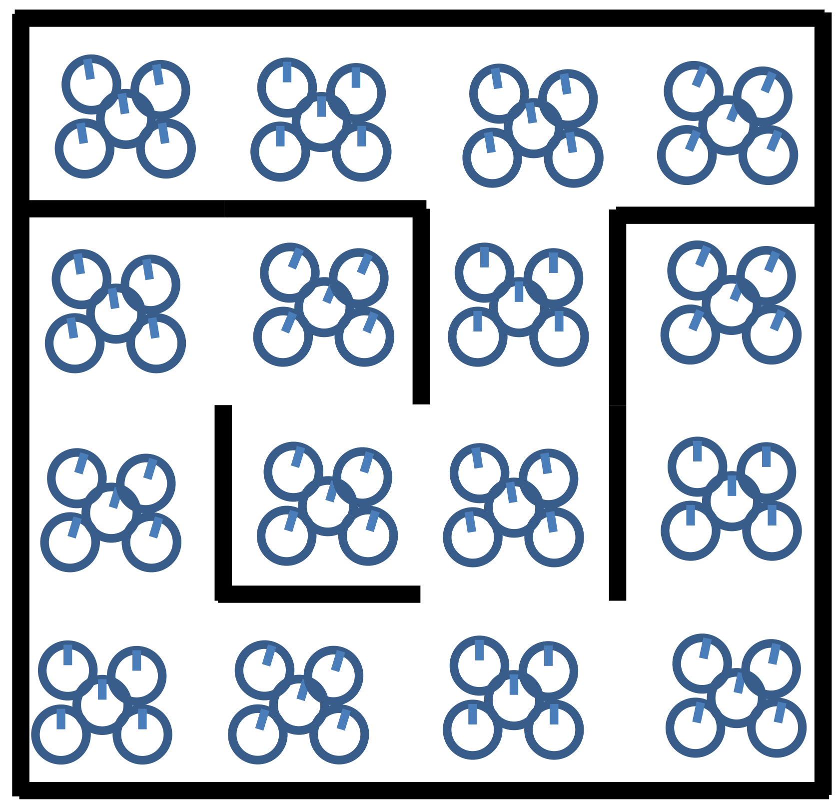


Initialization

- Initial particle distribution $s_0^{[m]}$ $(1 \leq m \leq M)$.
- Start by assuming $p(s_0)$ is the uniform distribution.
- Take M samples of s_0 and weight each with an importance factor $w_t^{[m]} = 1/M$



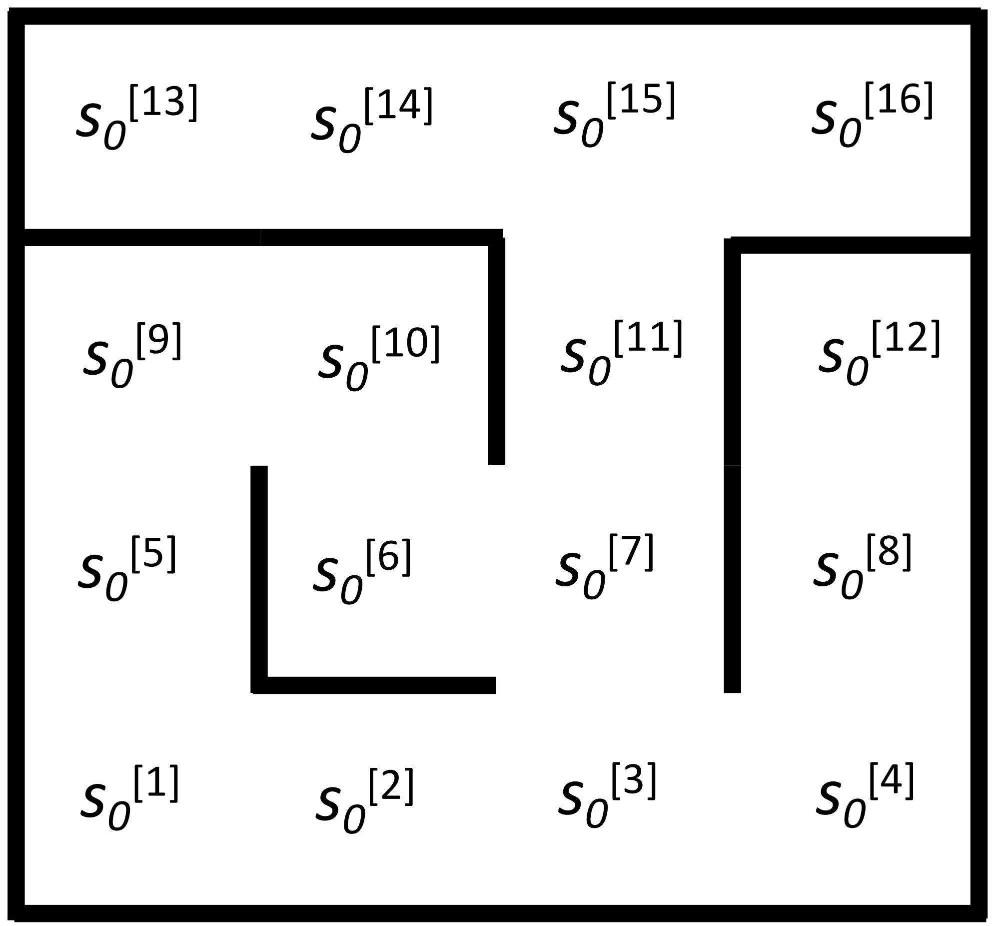
M=16



M=80

Initialization

- Initial particle distribution $s_0^{[m]}$ ($1 \le m \le M=16$)

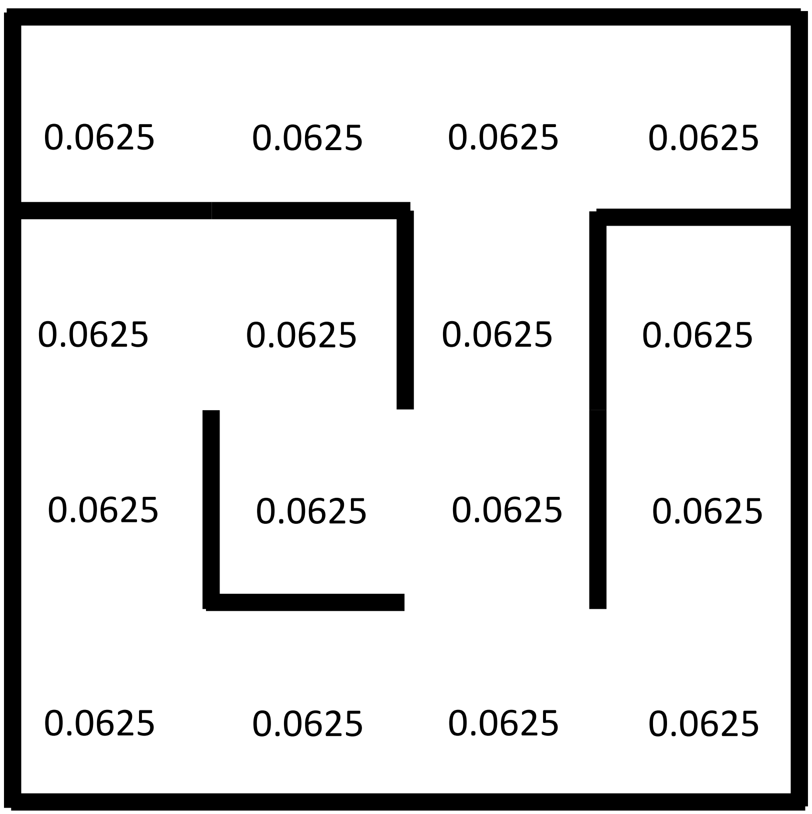


Initialization (16 particles)

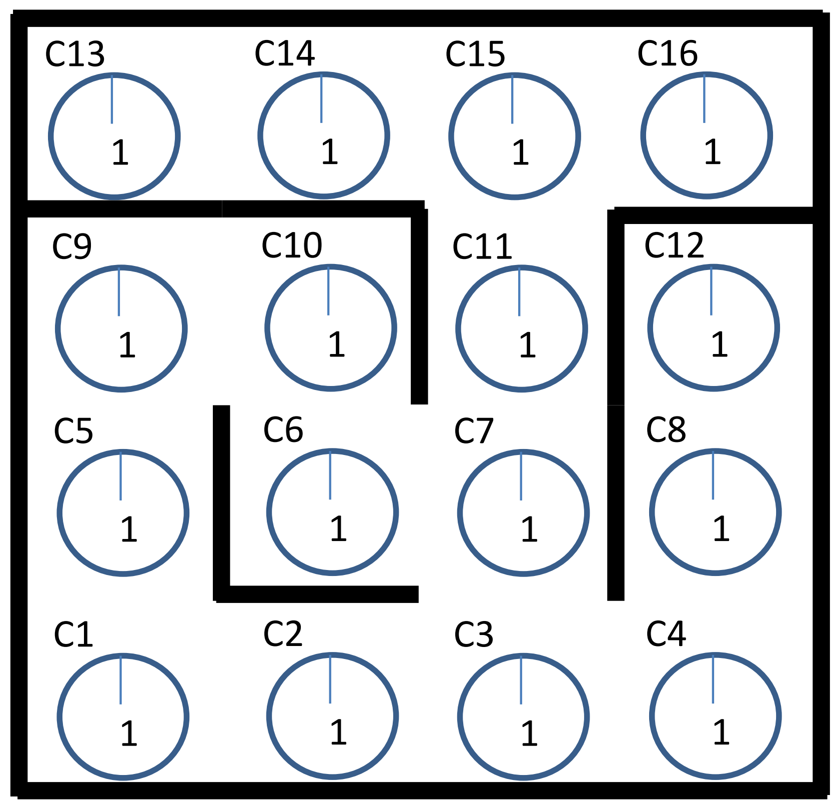
- Initial particle distribution $s_0^{[m]}$ $(1 \le m \le M)$
- $p(s_0^{[m]})$, $w_t^{[m]} = 1/M$ (importance factor)

Grid Cell Labels ("CN"), N=1,…M

Number of Particles inside Circle



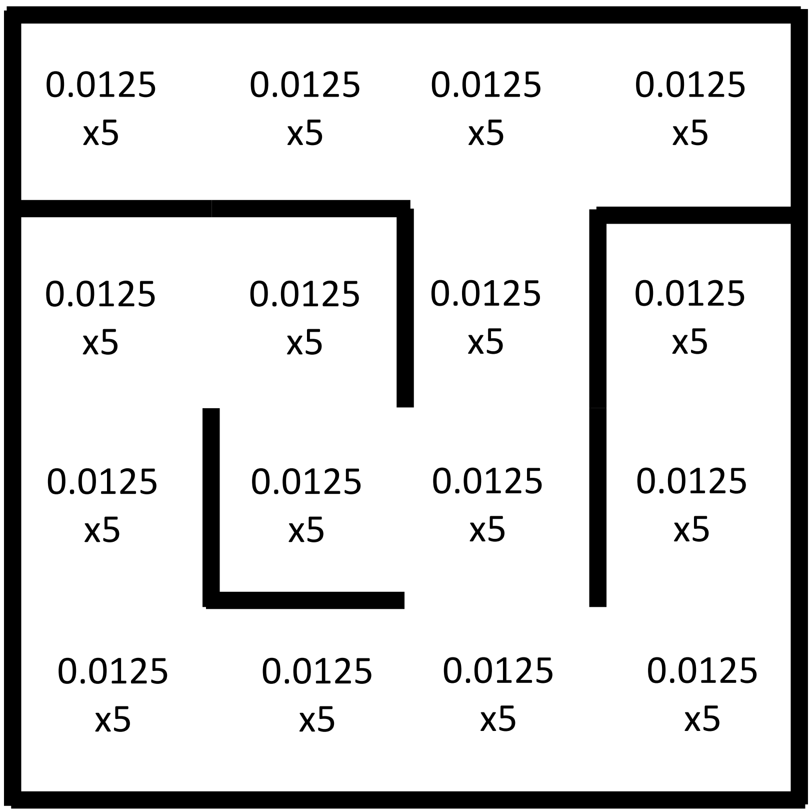
M=16



M=16

Initialization (80 particles)

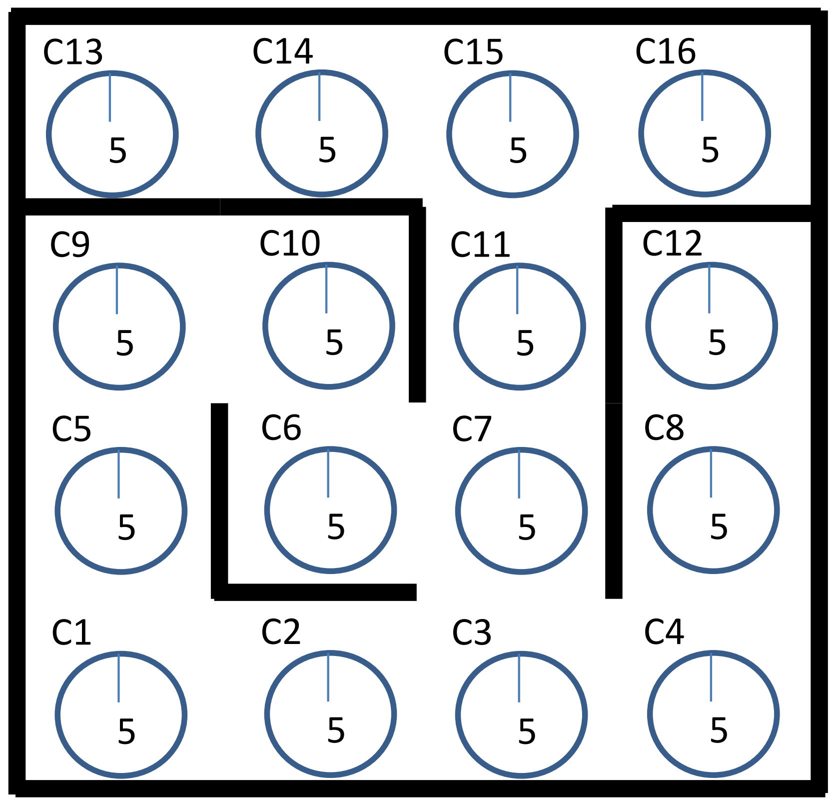
- Initial particle distribution $s_0^{[m]}$ $(1 \le m \le M)$
- $p(s_0^{[m]})$, $w_t^{[m]} = 1/M$ (importance factor)



M=80

Grid Cell Labels ("CN"), N=1,…M

Number of Particles inside Circle



M=80

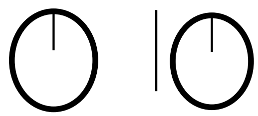
Measurement Model

- Assume z="wall" (z=1) corresponds to the sensor reading "wall".
- Assume z="no wall" (z=0) corresponds to the sensor reading "no wall".
- Assume p(z="wall"|s) (or p(z=1) or p(z=1|s)) corresponds to the probability of the sensor reading "wall" given the current state s.
- Assume p(z="no wall"|s) (or p(z=0) or p(z=0|s)) corresponds to the probability of the sensor reading "no wall" given the current state s.
- Assume p(z="no wall"|s="not in front of a wall") = 0.7 is the probability of the sensor reading "no wall" given that the robot is "not in front of a wall".
- Assume p(z="wall"|s="not in front of a wall") = 0.3 is the probability of the sensor reading "wall" given that the robot is "not in front of a wall".
- Assume p(z="no wall"|s="in front of a wall") = 0.1 is the probability of the sensor reading "no wall" given that the robot is "in front of a wall".
- Assume p(z="wall"|s="in front of a wall") = 0.9 is the probability of the sensor reading "wall" given that the robot is "in front of a wall".

Measurement Distribution Model

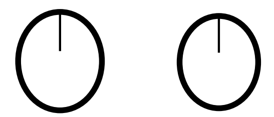
Robot Left Sensor

	"no wall"	"wall"	
	s = 0	s = 1	
$p(z = 0 \mid s = 0)$	.6	.2	$p(z = 0 \mid s = 1)$
$p(z = 1 \mid s = 0)$	.4	.8	$p(z = 1 \mid s = 1)$



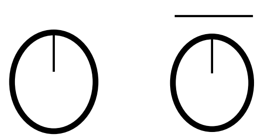
Robot Right Sensor

	"no wall"	"wall"	
	s = 0	s = 1	
$p(z = 0 \mid s = 0)$	.6	.2	$p(z = 0 \mid s = 1)$
$p(z = 1 \mid s = 0)$	.4	.8	$p(z = 1 \mid s = 1)$



Robot Front Sensor

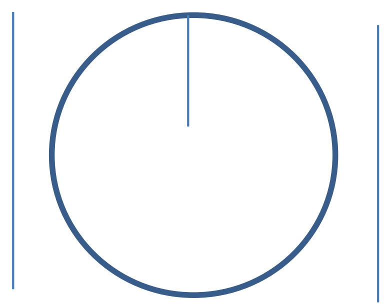
	"no wall"	"wall"	
	s = 0	s = 1	
$p(z = 0 \mid s = 0)$	.7	.1	$p(z = 0 \mid s = 1)$
$p(z = 1 \mid s = 0)$	.3	.9	$p(z = 1 \mid s = 1)$



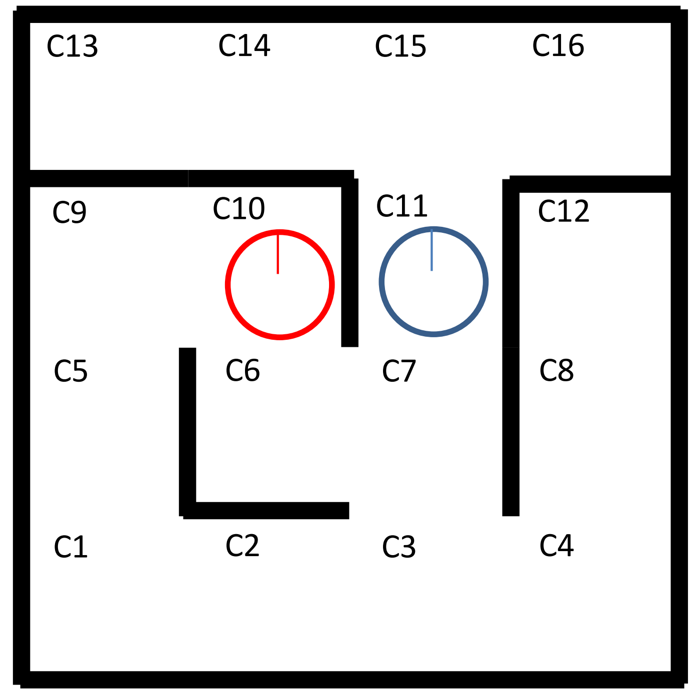
Measurement Estimation

- Assume current readings are “no front wall” and “side walls”.

Robot Sensor Readings



L(1)F(0)R(1)

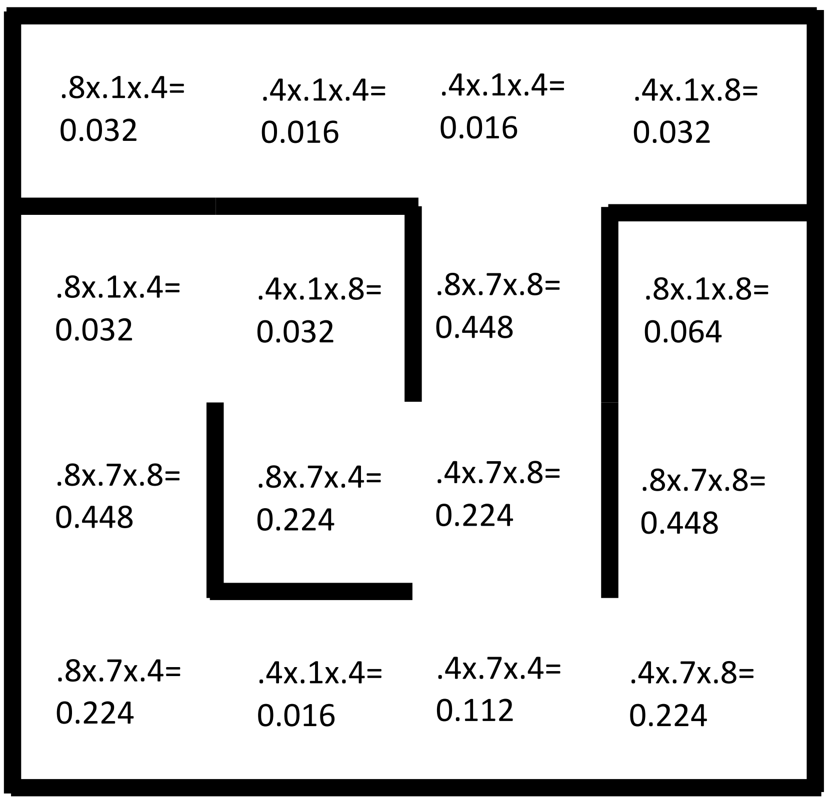


- Assume L(1)F(0)R(1) is current reading “left wall”, “no front wall”, “right wall”.
- At C11 (blue circle), the measurement probabilities $p(z|s^{[11]})$ are as follows:
  $p(L(z=1)|s^{[11]})=0.8$, $p(F(z=0)|s^{[11]})=0.7$, $p(R(z=1)|s^{[11]})=0.8$
- At C10 (red circle), the measurement probabilities $p(z|s)$ are as follows:
  $p(L(z=1)|s^{[10]})=0.4$, $p(F(z=0)|s^{[10]})=0.1$, $p(R(z=1)|s^{[10]})=0.8$

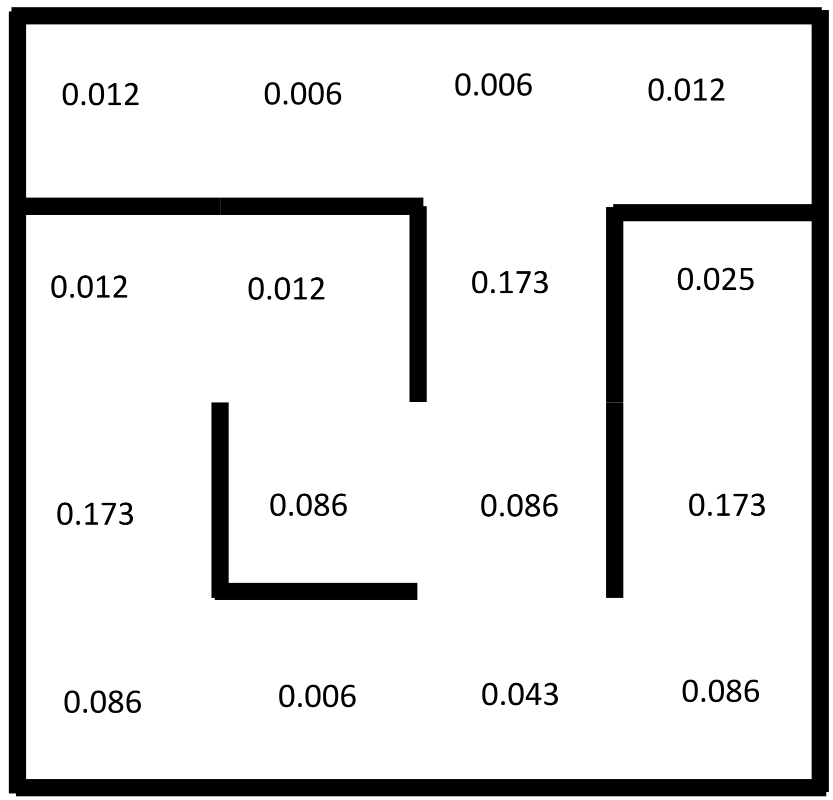
Measurement Estimation

- Estimate probability of measurement z_t given particle state $s_t^{[m]}$

  $p(z_t \mid s_t^{[m]})$
- Normalize probabilities (take probabilities from the left diagram and divide by total to obtain probabilities on the right, e.g. at C13: 0.032/2.592=0.012)



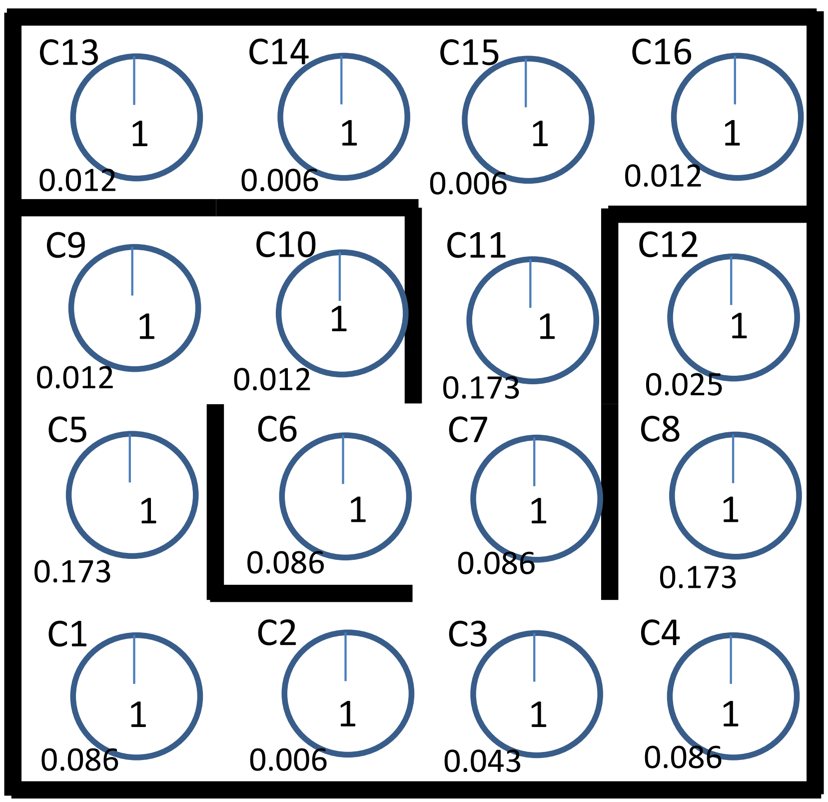
Before normalizing $s_1^{[m]}$
(Total: 2.592)



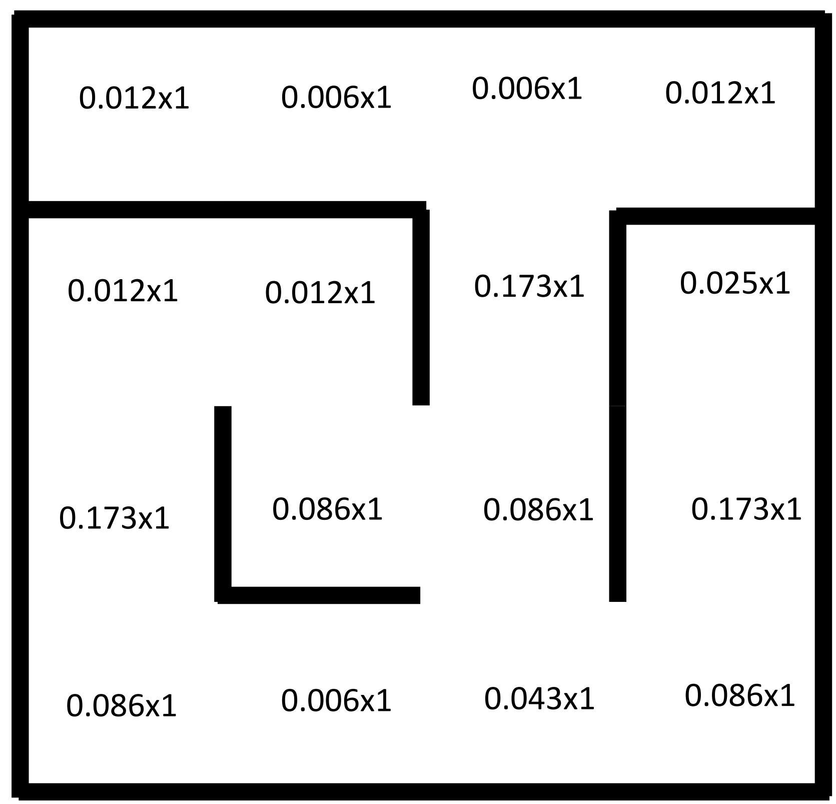
After normalizing $s_1^{[m]}$

Importance Factor (16 particles)

- The importance factor $w_t^{[m]}$ is computed from probability measurement z_t given particle state $s_t^{[m]}$ current distribution, i.e. corresponding to number of particles in each cell, $w_t^{[m]} = p(z_t \mid s_t^{[m]})$
- Multiply each cell (state) probabilities by number of particles on that cell (left diagram) and then normalize, at C13: 0.012 x 1 = 0.012



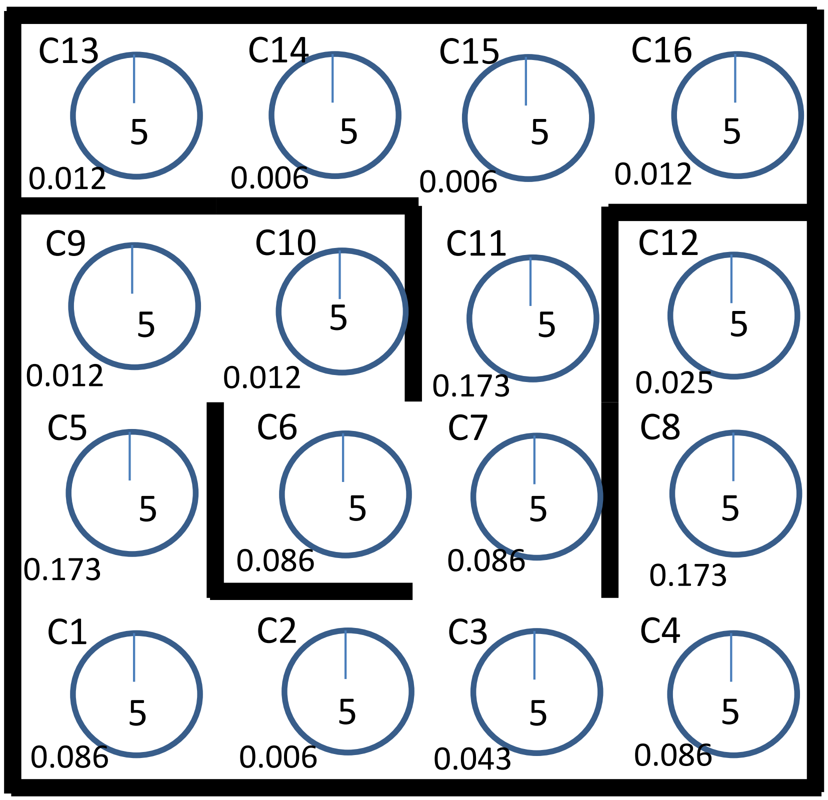
Current Distribution



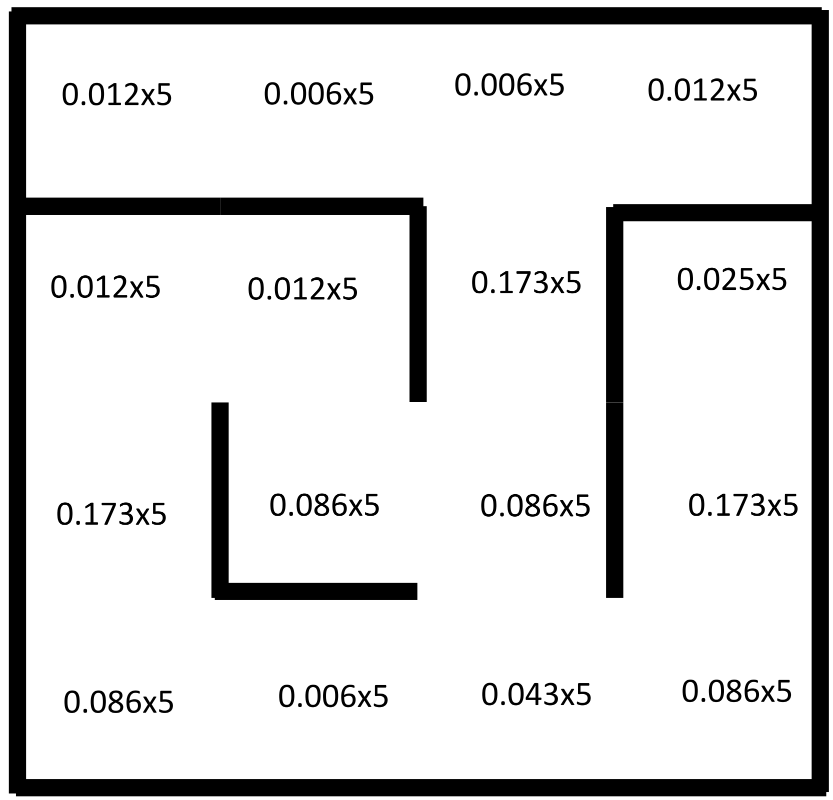
Importance Factor (#particles * z_t)

Importance Factor (80 particles)

- The importance factor $w_t^{[m]}$ is computed from measurement estimation z_t given particle state $s_t^{[m]}$ current distribution, i.e. corresponding to number of particles in each cell, $w_t^{[m]} = p(z_t \mid s_t^{[m]})$
- Multiply each cell (state) probabilities by number of particles on that cell (left diagram) and then normalize, at C13: 0.012 x 5 = 0.060



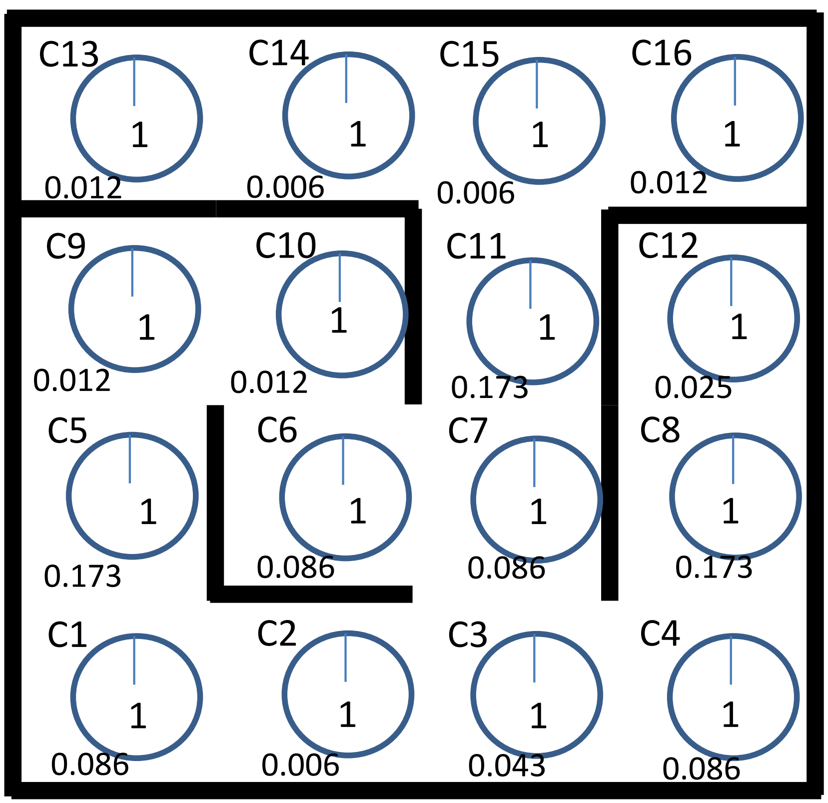
Current Distribution



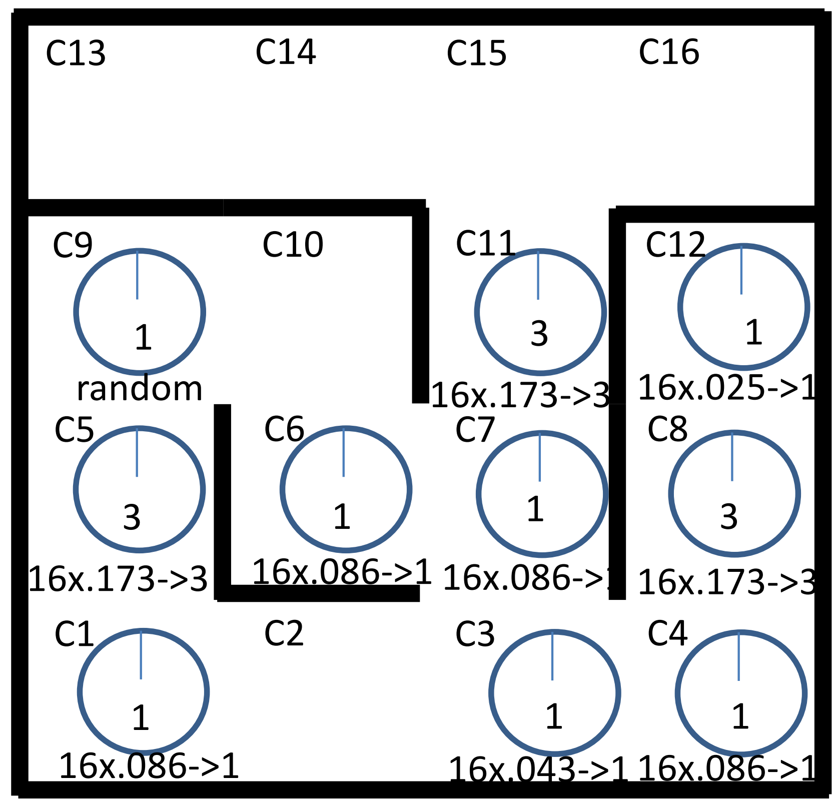
Importance Factor (#particles * z_t)

Resampling (16 particles)

- Resampling or importance sampling replaces particles proportional to their measured likelihood, e.g. draw new samples at cells with highest probabilities.
- Same total number of particles. Normalize particle distribution $s_t^{[m]}$ ($1 \le m \le$ M).
- Multiply total number of particles by state probability distribution, at C11: 16x0.173=2.768 ~ 3.



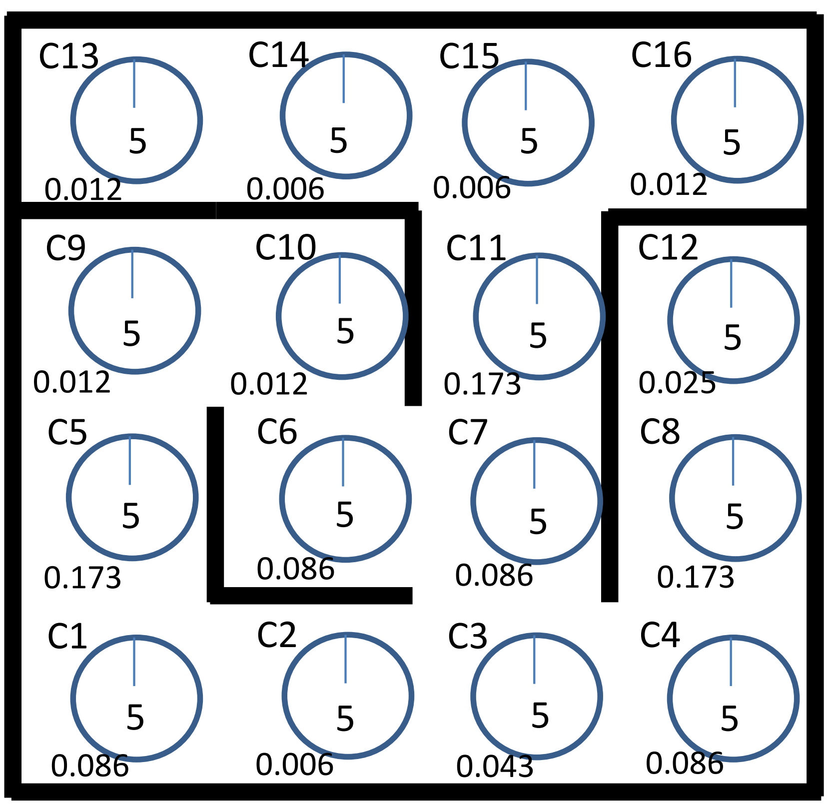
Before resampling $s_1^{[m]}$



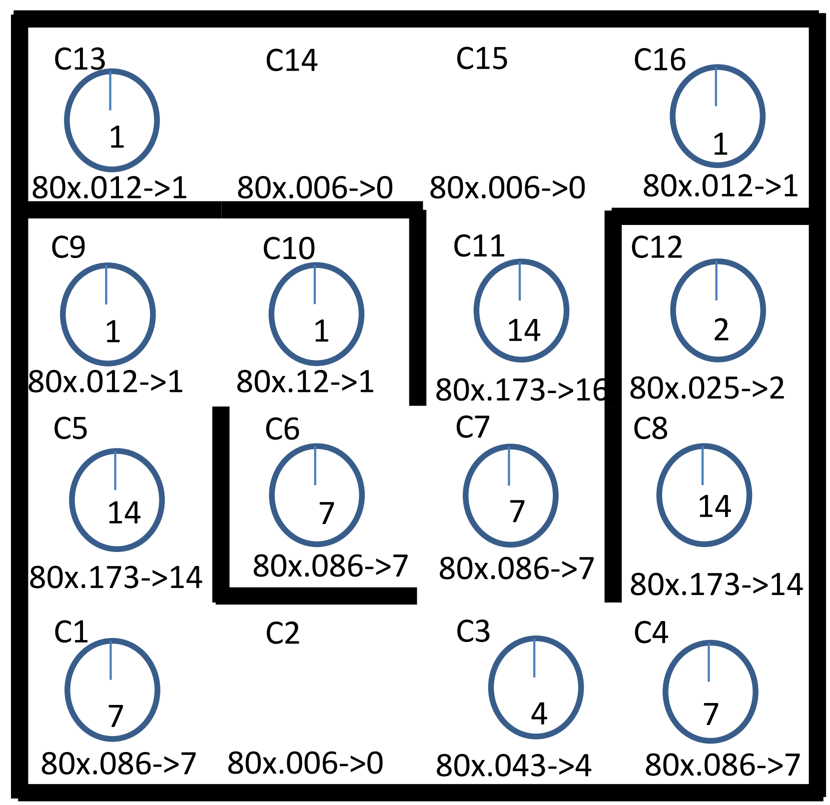
After resampling $s_1^{[m]}$

Resampling (80 particles)

- Resampling or importance sampling replaces particles proportional to their measured likelihood, e.g. draw new samples at cells with highest probabilities.
- Same total number of particles. Normalize particle distribution $s_t^{[m]}$ ($1 \le m \le M$).
- Multiply total number of particles by state probability distribution, at C11: 80x0.173=13.84 ~ 14.



Before resampling $s_1^{[m]}$



After resampling $s_1^{[m]}$

Motion Model

Forward Motion Probability Distribution

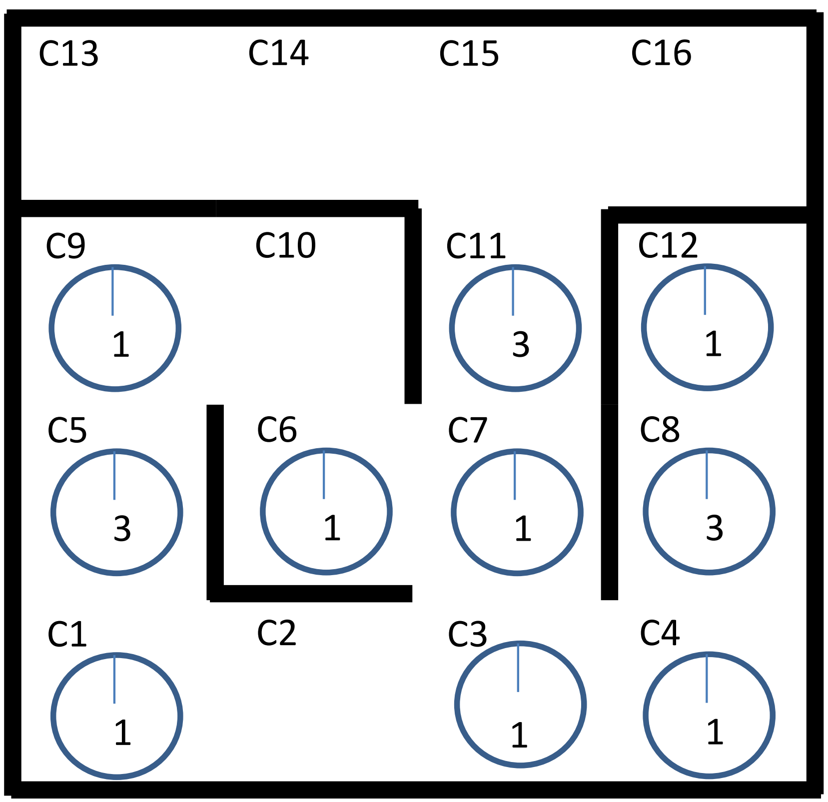
		F_1		
	L_1	S_1	R_1	

		.7		
	.1	.1	.1	

Motion Update (16 particles)

- The algorithm computes a state $s_t^{[m]}$ based on particle $s_{t-1}^{[m]}$ and control u_t corresponding to the prediction

  $\overline{\text{bel}}(s_t)$: $s_t^{[m]} \sim p(s_t \mid s_{t-1}^{[m]}, u_t)$



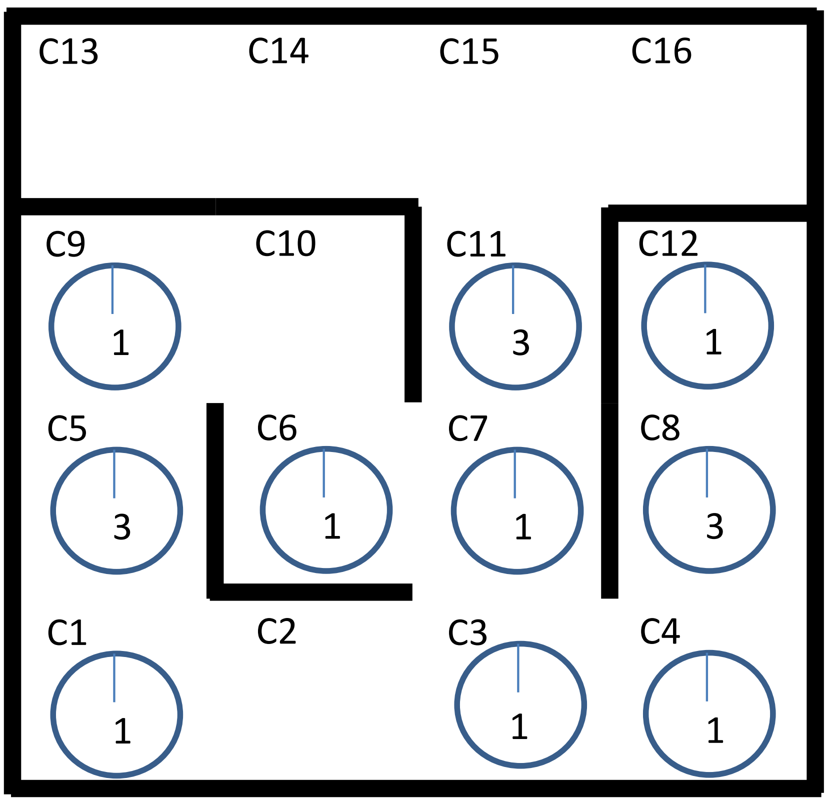
Before motion $s_1^{[m]}$

- C1 1-particle moves with 0.7 prob to C5
- C3 1-particle moves with 0.7 prob to C7
- C4 1-particle moves with 0.7 prob to C8
- C5 3-particles move with 0.7 prob to C9 (2 particles) and stay with 0.3 prob in C5 (1 particle)
- C6 1-particle moves with 0.7 prob to C10
- C7 1-particle moves with 0.7 prob to C11
- C8 3-particles move with 0.7 prob to C12 (2 particles) and stay with 0.3 prob in C8 (1 particle)
- C9 1-particle stays with 0.9 prob in C9
- C11 3-particles move with 0.7 prob to C15 (2 particles) and stay with 0.3 prob in C11 (1 particle)
- C12 1-particle stays with 1.0 prob in C12

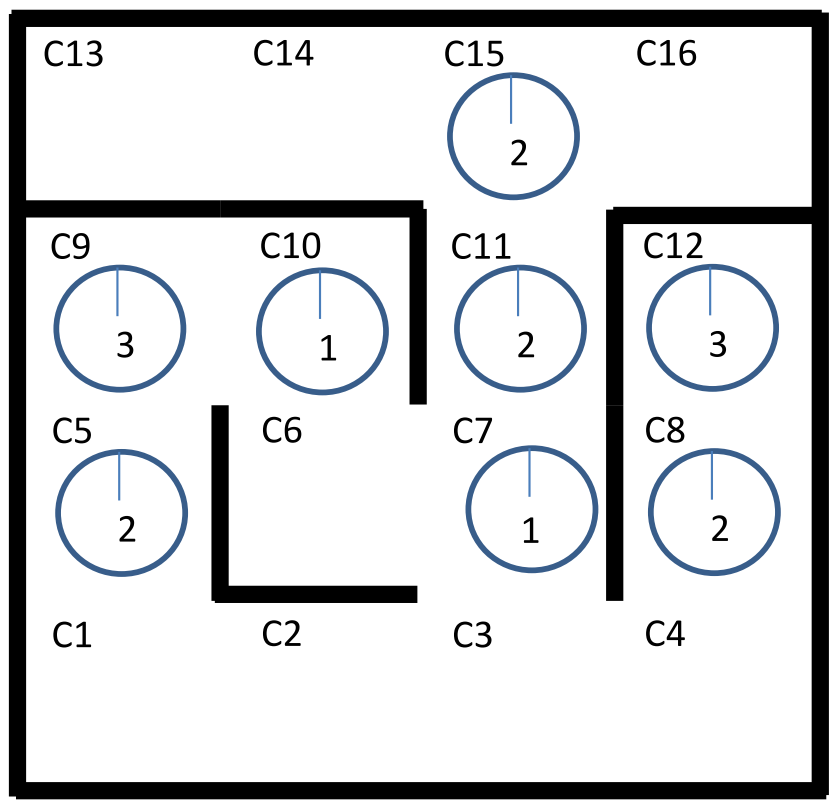
Motion Update (16 particles)

- The algorithm computes a state $s_t^{[m]}$ based on particle $s_{t-1}^{[m]}$ and control u_t corresponding to the prediction

  $\overline{\text{bel}}(s_t)$: $s_t^{[m]} \sim p(s_t \mid s_{t-1}^{[m]}, u_t)$



Before motion $s_1^{[m]}$

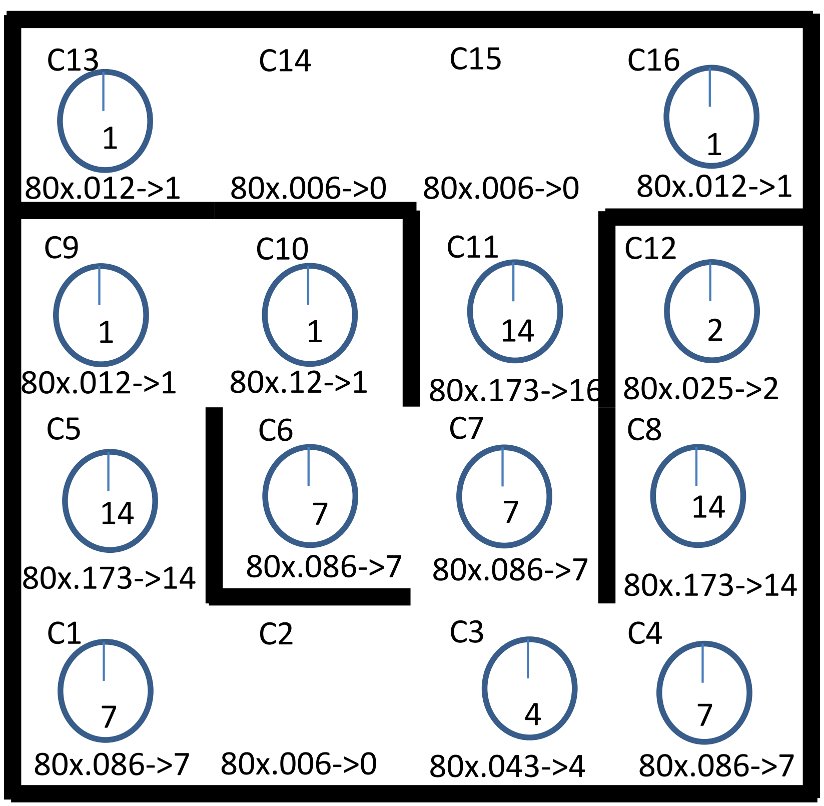


After motion $s_2^{[m]} \sim p(s_2 \mid s_1^{[m]}, u_2)$

Motion Update (80 particles)

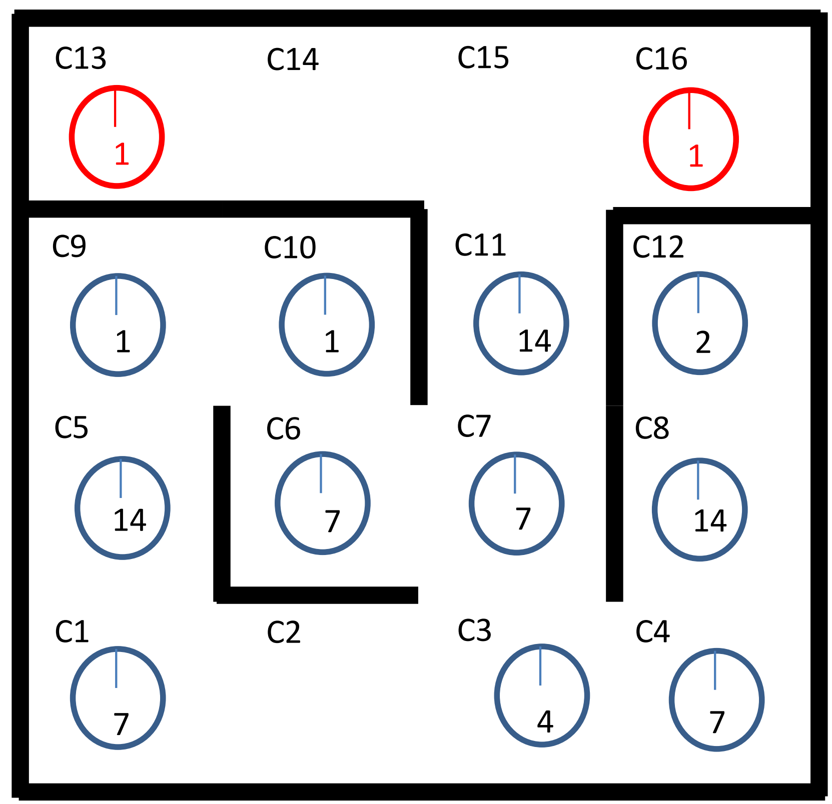
- The algorithm computes a state $s_t^{[m]}$ based on particle $s_{t-1}^{[m]}$ and control u_t corresponding to the prediction

  $\overline{\mathrm{bel}}(s_t)$: $s_t^{[m]} \sim p(s_t \mid s_{t-1}^{[m]}, u_t)$



Before resampling $s_1^{[m]}$

Top 1 Row

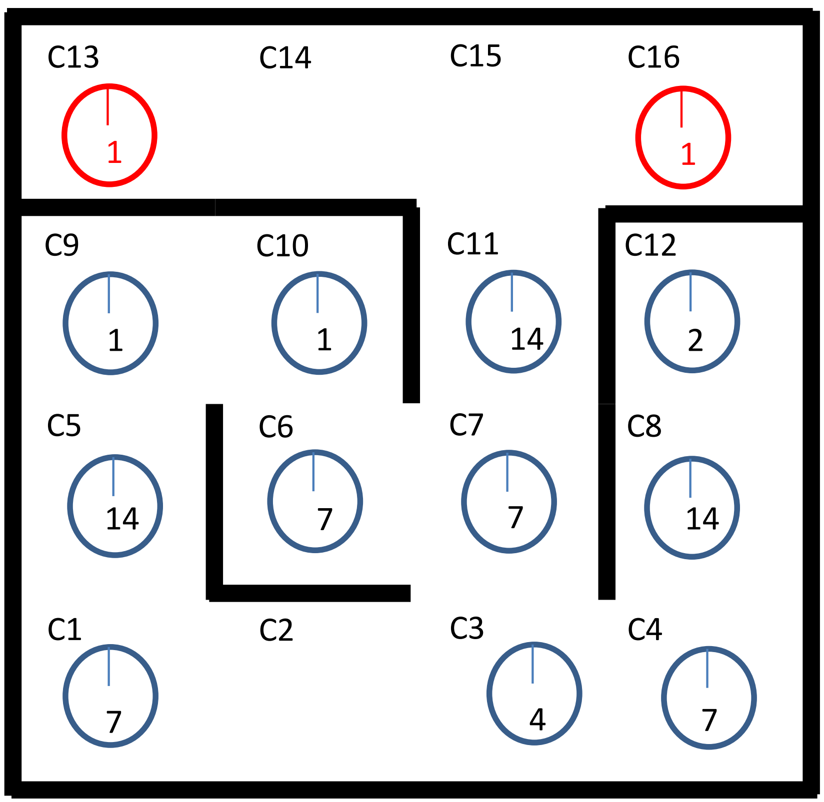


After motion $s_2^{[m]} \sim p(s_2 \mid s_1^{[m]}, u_2)$

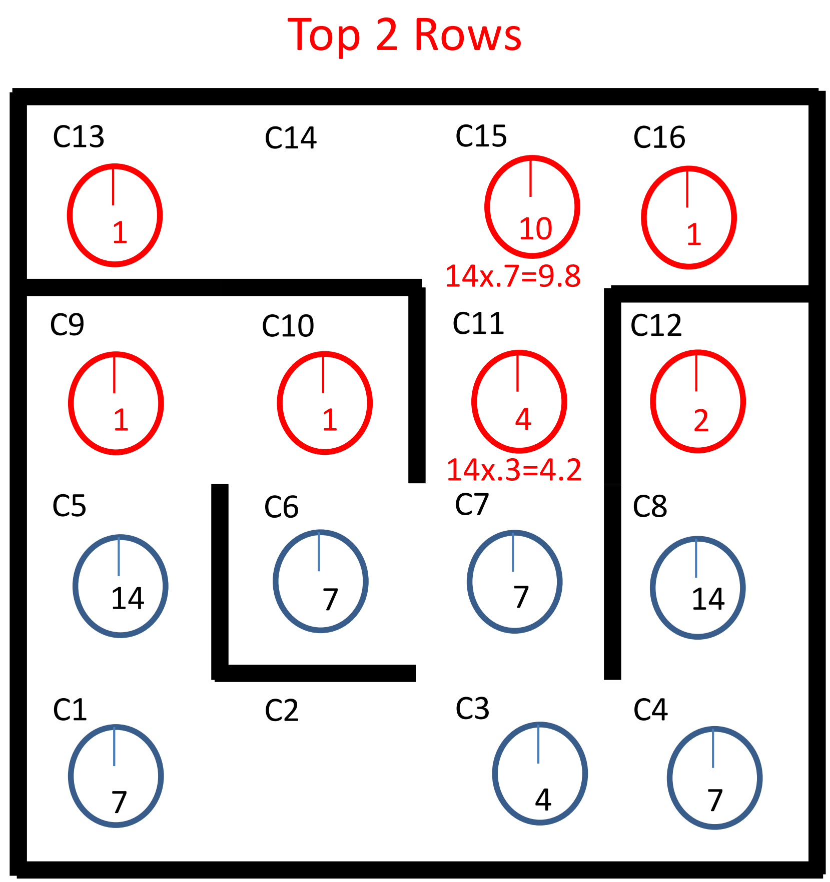
Motion Update (80 particles)

- The algorithm computes a state $s_t^{[m]}$ based on particle $s_{t-1}^{[m]}$ and control u_t corresponding to the prediction

$$\overline{\text{bel}}(s_t):\quad s_t^{[m]} \sim p(\, s_t \mid s_{t-1}^{[m]} \,,\, u_t \,)$$



Before motion $s_1^{[m]}$

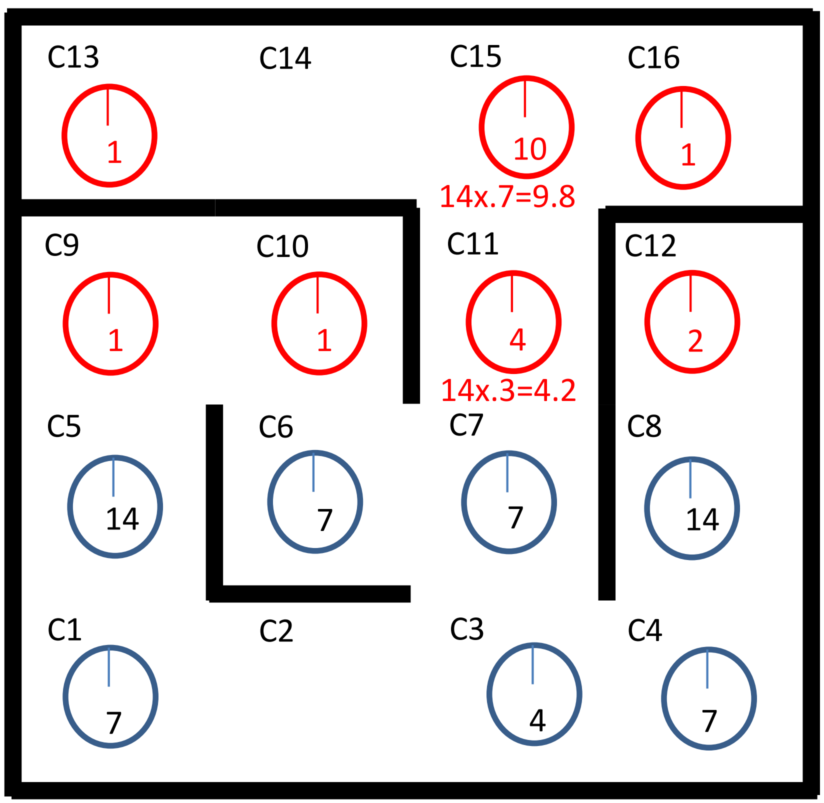


After motion $s_2^{[m]} \sim p(\, s_2 \mid s_1^{[m]} \,,\, u_2 \,)$

Motion Update (80 particles)

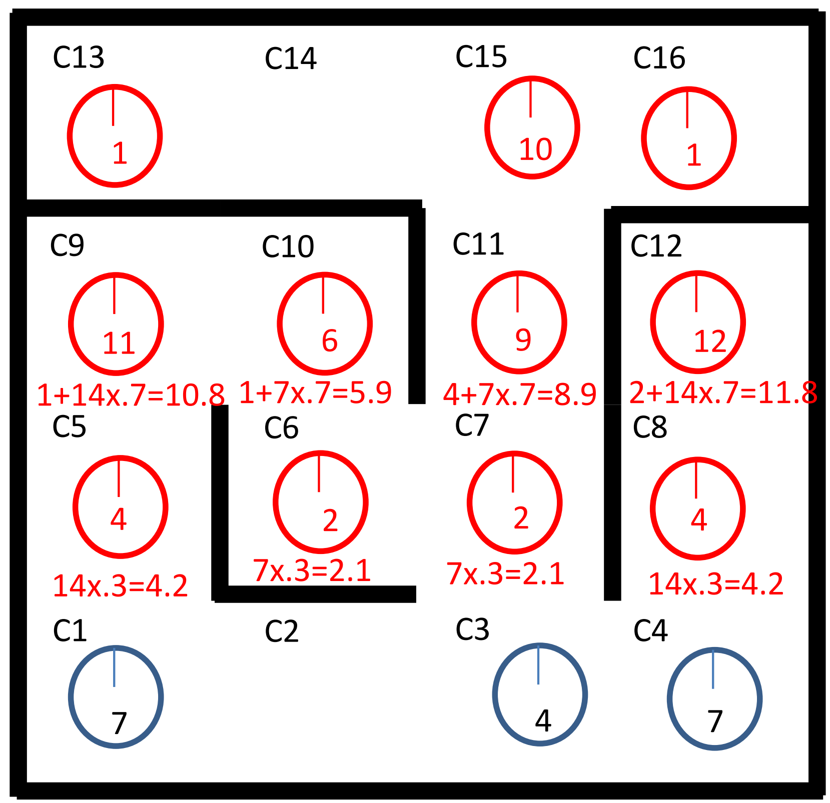
- The algorithm computes a state $s_t^{[m]}$ based on particle $s_{t-1}^{[m]}$ and control u_t corresponding to the prediction

  $\overline{\text{bel}}(s_t)$: $s_t^{[m]} \sim p(s_t \mid s_{t-1}^{[m]}, u_t)$



Before motion $s_1^{[m]}$

Top 3 Rows



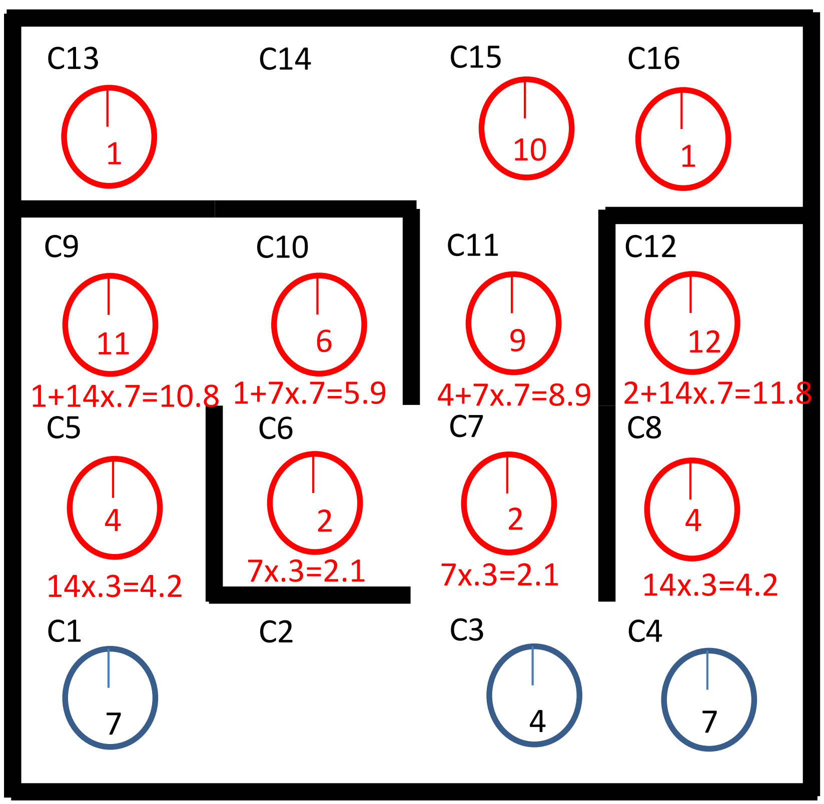
After motion $s_2^{[m]} \sim p(s_2 \mid s_1^{[m]}, u_2)$

Motion Update (80 particles)

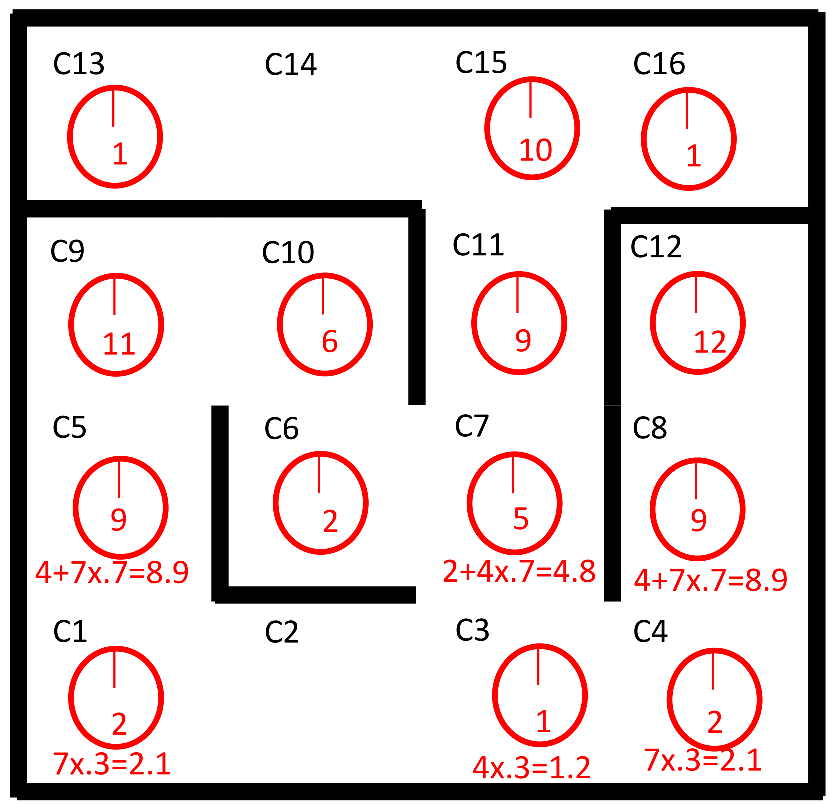
- The algorithm computes a state $s_t^{[m]}$ based on particle $s_{t-1}^{[m]}$ and control u_t corresponding to the prediction

$$\overline{\text{bel}}(s_t):\quad s_t^{[m]} \sim p(\, s_t \mid s_{t-1}^{[m]} \,,\, u_t \,)$$

All Rows



Before motion $s_1^{[m]}$

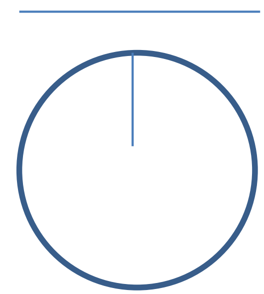


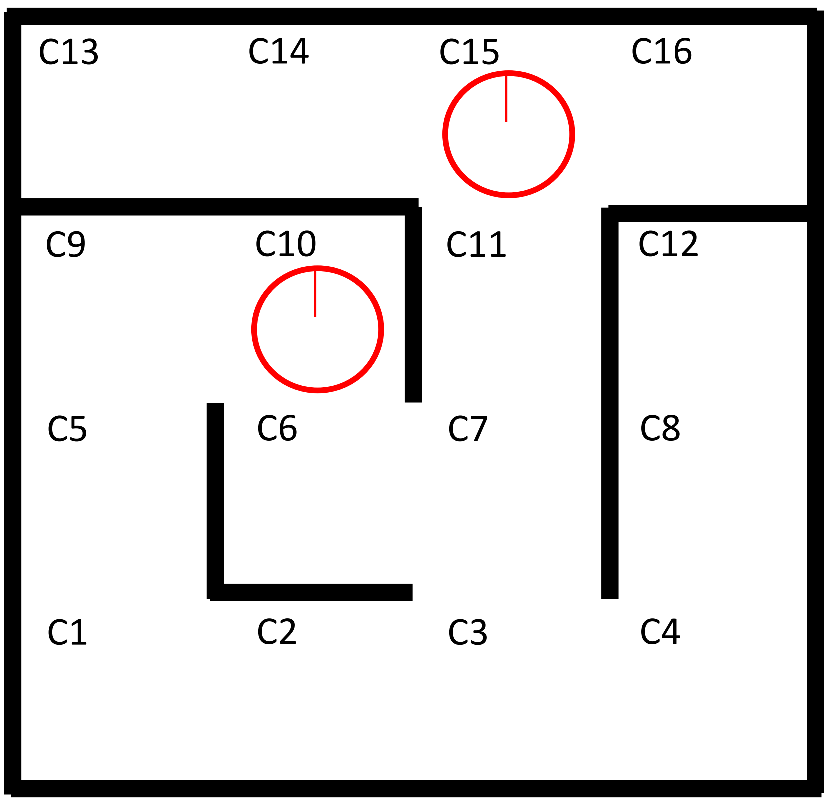
After motion $s_2^{[m]} \sim p(\, s_2 \mid s_1^{[m]} \,,\, u_2 \,)$

Measurement Update

- Estimate probability of measurement z_t given particle state $s_t^{[m]}$:
  p(z_t | $s_t^{[m]}$)

Assume Robot Sensor Readings



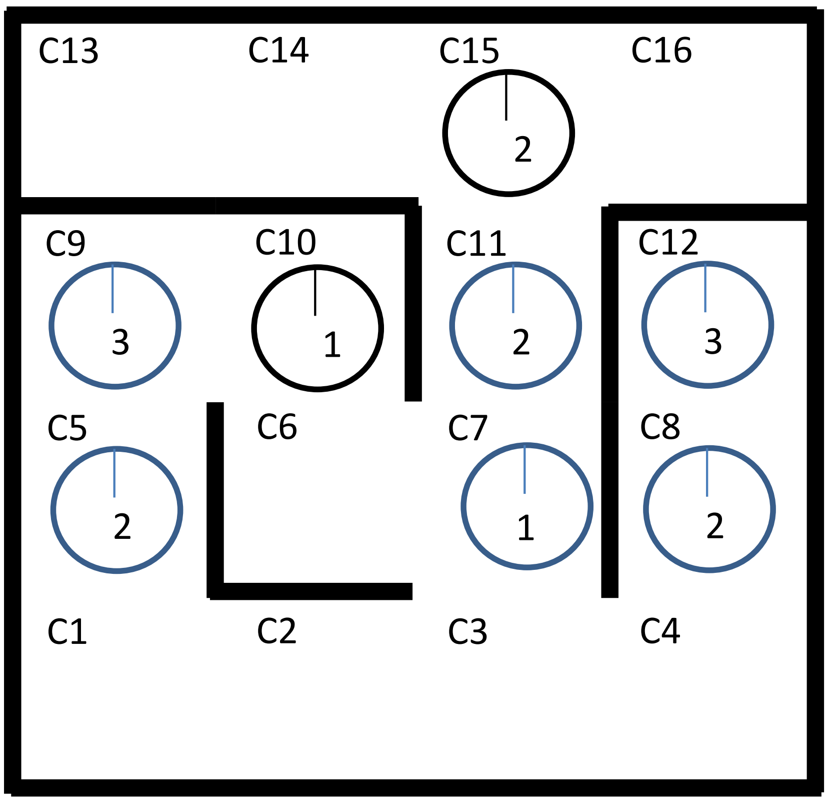


After motion $s_2^{[m]} \sim$ p(s_2 | $s_1^{[m]}$, u_2)

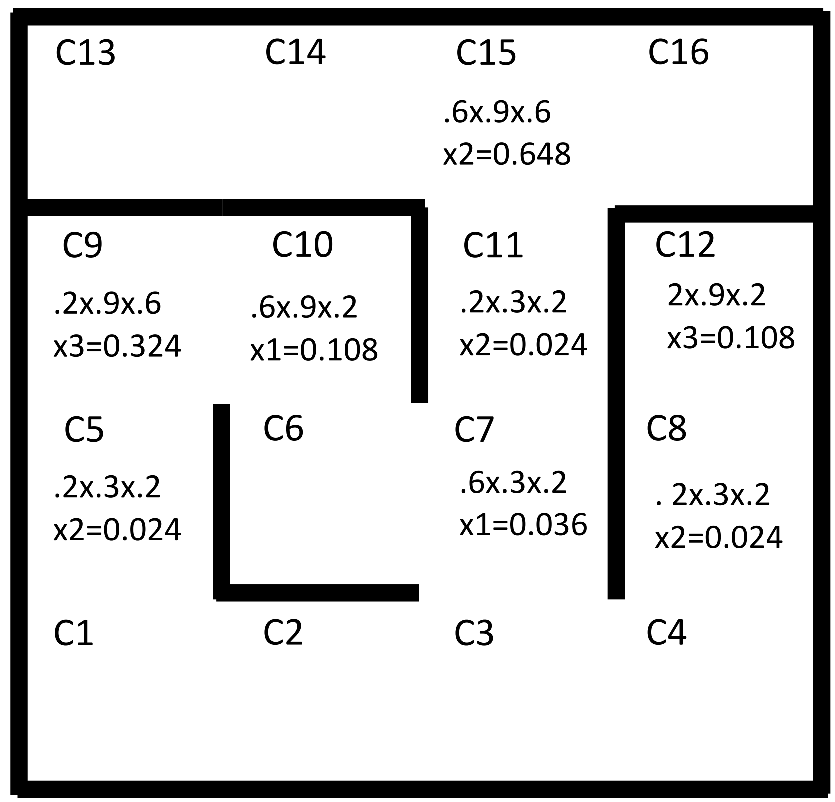
- Assume L(0)F(1)R(0) is current reading "no left wall", "front wall", "no right wall".
- For example, at C15, p(z|s$^{[15]}$):
  p(L(z=0)|s$^{[15]}$)=0.6
  p(F(z=1)|s$^{[15]}$)=0.9
  p(R(z=0)|s$^{[15]}$)=0.6
- For example, at C10, p(z|s$^{[10]}$):
  p(L(z=0)|s$^{[10]}$)=0.6
  p(F(z=1)|s$^{[10]}$)=0.9
  p(R(z=0)|s$^{[10]}$)=0.2

Importance Factor (16 particles)

- The importance factor $w_t^{[m]}$ is computed from measurement estimation z_t given particle state $s_t^{[m]}$ distribution, $w_t^{[m]} = p(z_t \mid s_t^{[m]})$
- Compute importance factors by multiplying measurement probabilities by number of particles in state. For example, at C15: .6x.9x.6*2=0.648)



Measurement Estimation $p(z_2 \mid s_2^{[m]})$

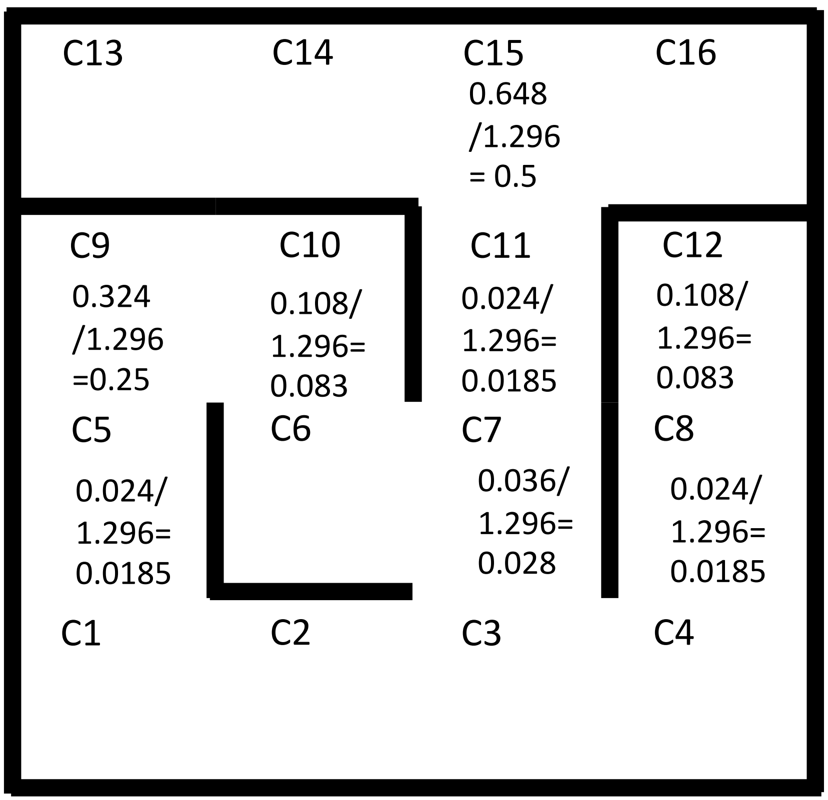


Importance Factor $w_2^{[m]} = p(z_2 \mid s_2^{[m]})$

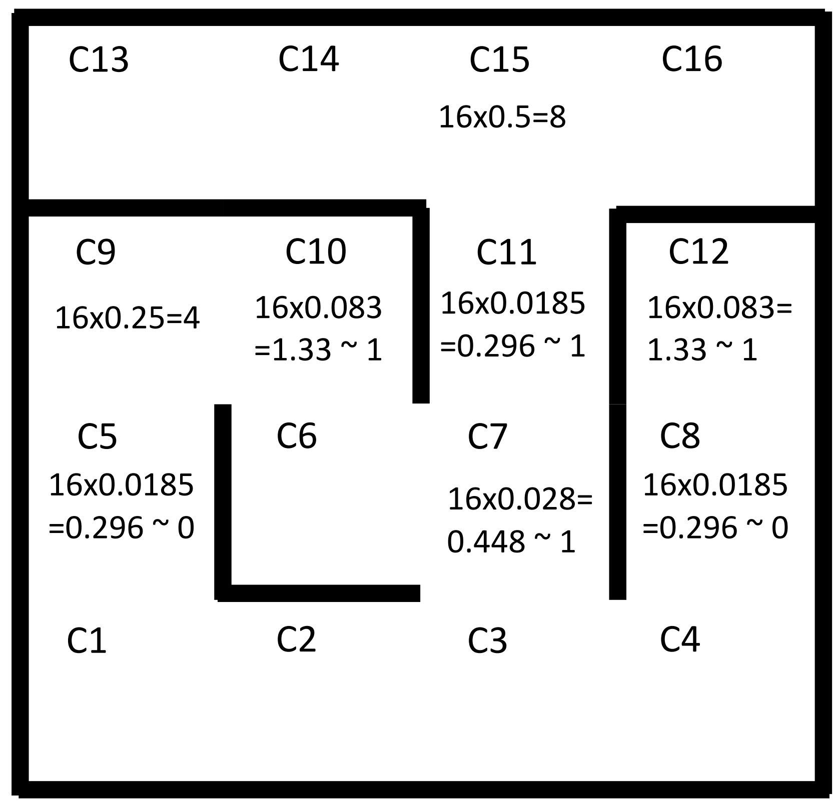
(Total: 1.296)

Resampling (16 particles)

- Resampling or importance sampling replaces particles proportional to their measured likelihood.
- Draw new samples at states with highest probabilities. Keep same total number of particles.
- Normalize particle distribution $s_t^{[m]}$ ($1 \leq m \leq M$).



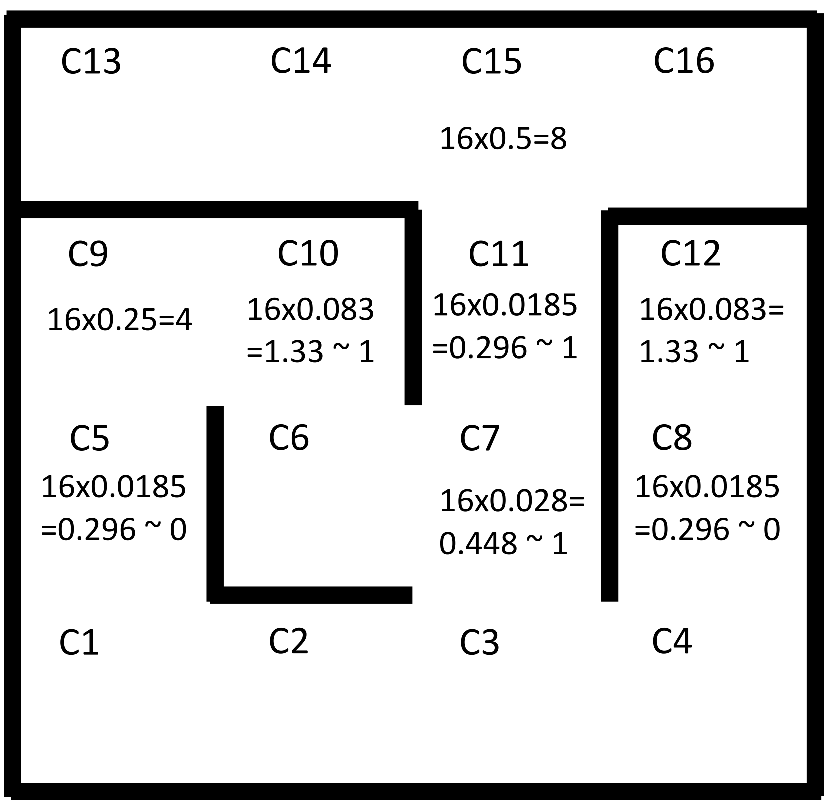
After normalizing by 1.296

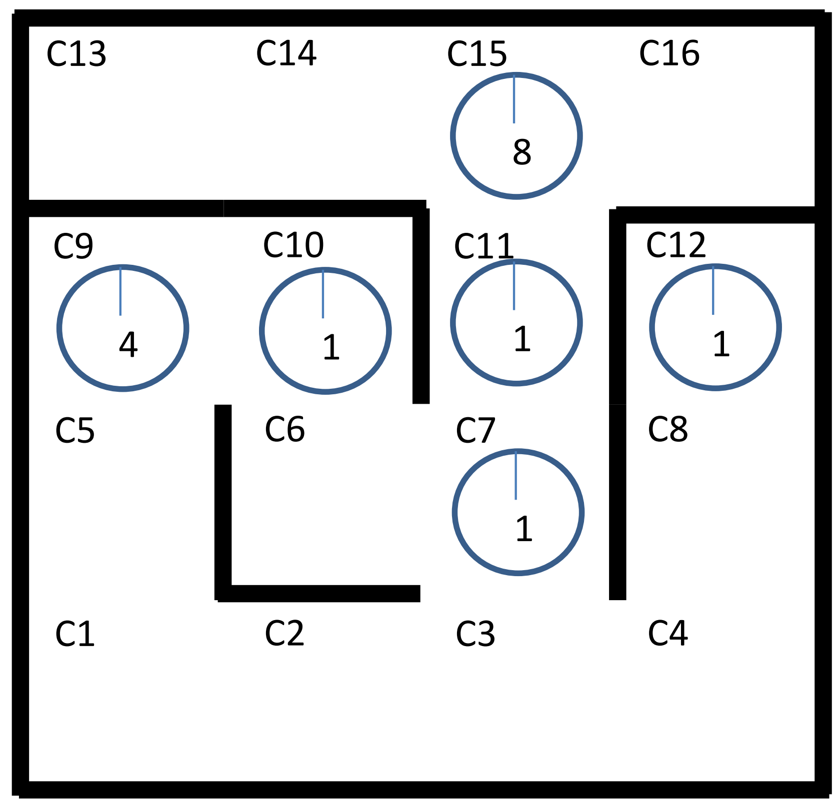


After resampling $s_2^{[m]}$

Resampling (16 particles)

- Resampling or importance sampling replaces particles proportional to their measured likelihood, e.g. draw new samples at poses with highest probabilities. Keep same total number of particles.
- Normalize particle distribution $s_t^{[m]}$ $(1 \leq m \leq M)$.

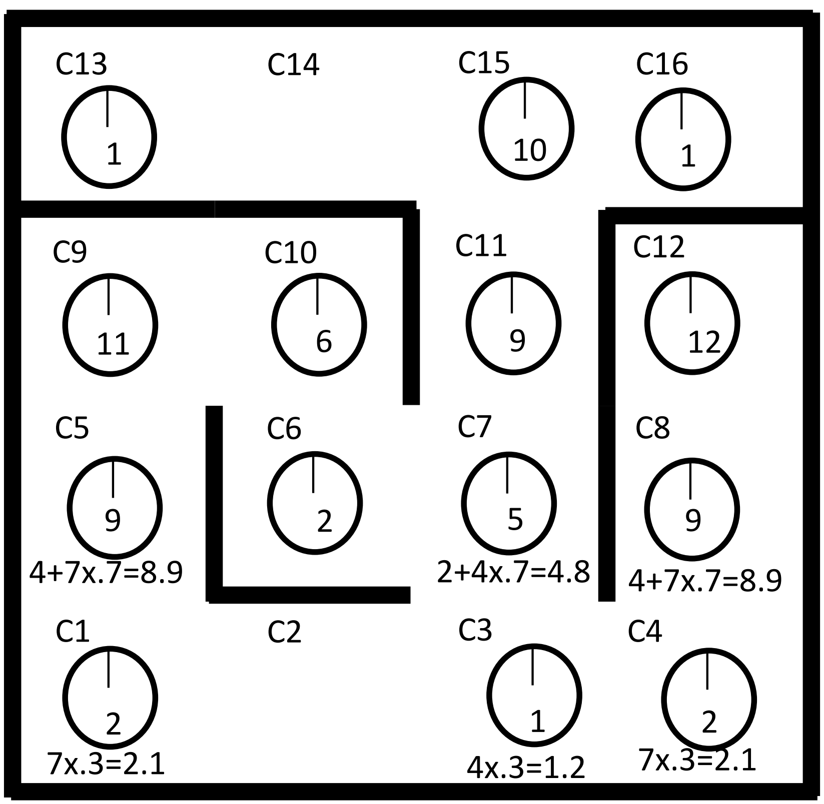




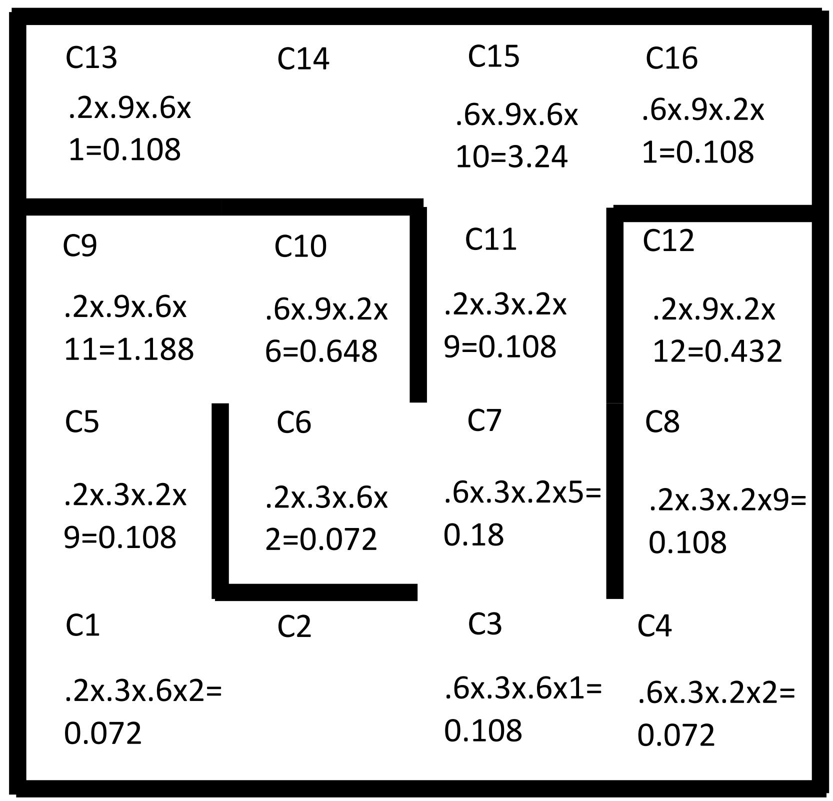
After resampling $s_2^{[m]}$

Importance Factor (80 particles)

- The importance factor $w_t^{[m]}$ is computed from measurement estimation z_t given particle state $s_t^{[m]}$ distribution, $w_t^{[m]} = p(z_t \mid s_t^{[m]})$
- Compute importance factors by multiplying measurement probabilities by number of particles in state. For example, at C15: .6x.9x.6*10=3.24)



Current Distribution

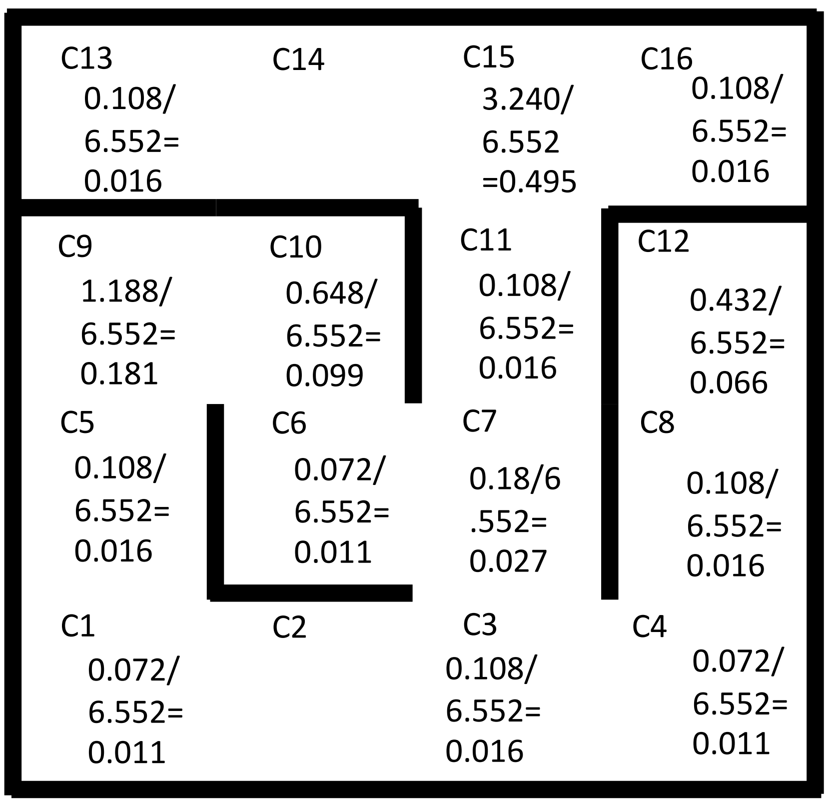


Importance Factor (#particles * z_t)

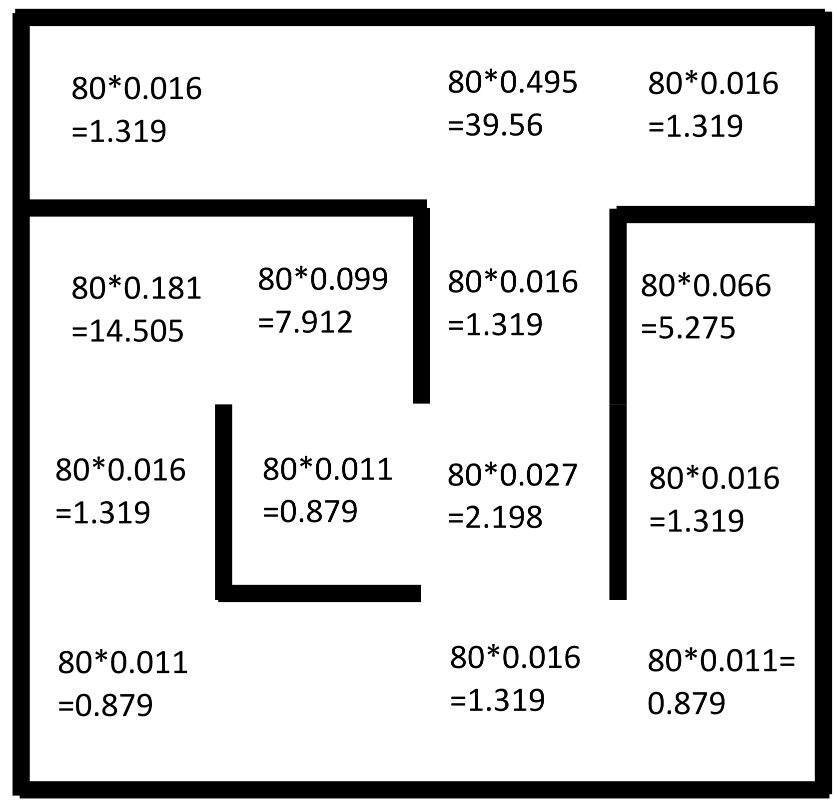
Resampling (80 particles)

- The importance factor $w_t^{[m]}$ is computed from measurement estimation z_t given particle state $s_t^{[m]}$ current distribution, i.e. corresponding to number of particles in each cell.

$$w_t^{[m]} = p(z_t \mid s_t^{[m]})$$



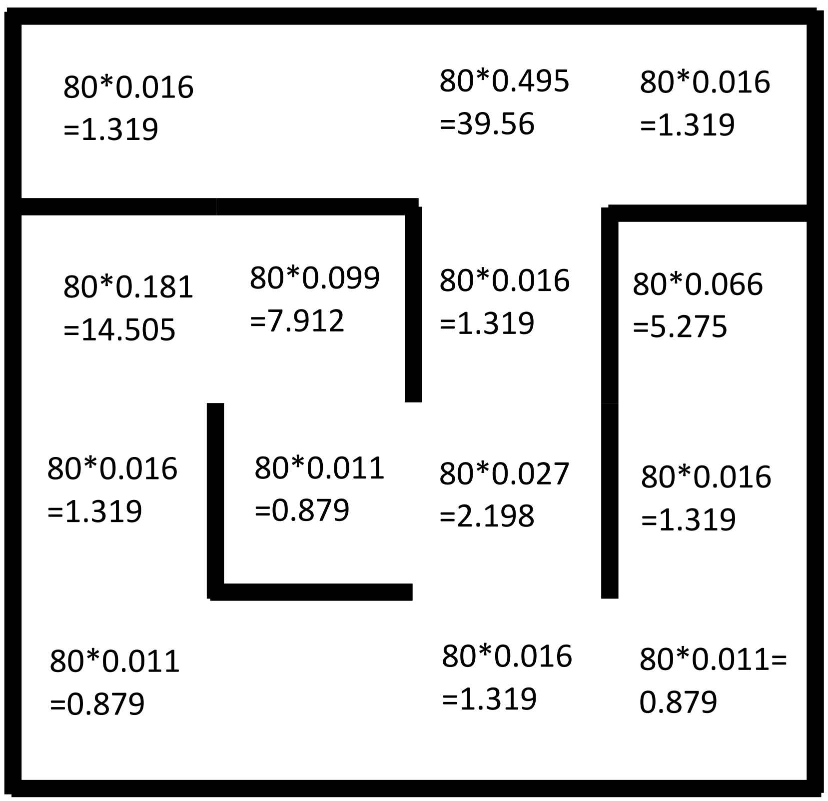
After normalizing by 6.552



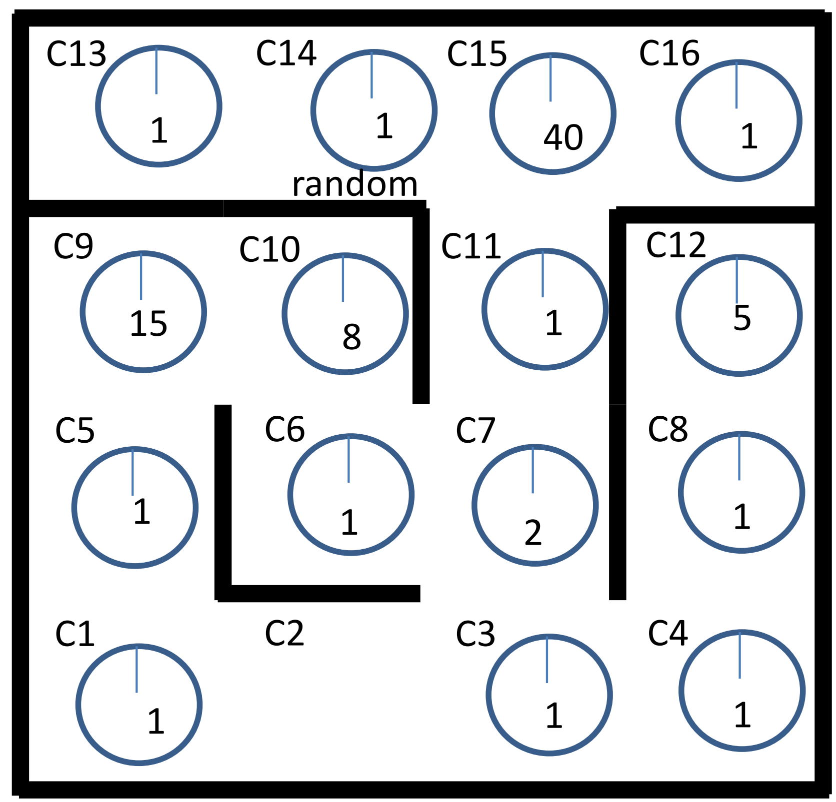
After resampling $s_2^{[m]}$

Resampling (80 particles)

- Resampling or importance sampling replaces particles proportional to their measured likelihood, e.g. draw new samples at poses with highest probabilities. Keep same total number of particles.
- Normalize particle distribution $s_t^{[m]}$ $(1 \le m \le M)$.



After resampling $s_2^{[m]}$



After resampling $s_2^{[m]}$

Particle Filter

AlgorithmParticleFilter(S_{t-1}, u_t , z_t**)**:

$\overline{S}_t = S_t = \emptyset$

for m = 1 to M do

sample $s_t^{[m]} \sim p(s_t \mid s_{t-1}^{[m]}, u_t)$

$w_t^{[m]} = p(z_t \mid s_t^{[m]})$

$\overline{S}_t = \overline{S}_t + \{s_t^{[m]}, w_t^{[m]}\}$

endfor

for m = 1 to M do

draw m with probability $\propto w_t^{[m]}$

add $s_t^{[m]}$ to S_t

endfor

return S_t

Monte Carlo Localization

AlgorithmMCL(S_{t-1}, u_t , z_t, *map***)**:

$\overline{S}_t = S_t = \emptyset$

for m = 1 to M do

$s_t^{[m]}$ = sample_motion_model(u_t ,$s_{t-1}^{[m]}$)

$w_t^{[m]}$ = measurement_model(z_t ,$s_t^{[m]}$,*map*)

$\overline{S}_t = \overline{S}_t + \{s_t^{[m]}, w_t^{[m]}\}$

endfor

for m = 1 to M do

draw m with probability $\propto w_t^{[m]}$

add $s_t^{[m]}$ to S_t

endfor

return S_t

Monte Carlo Localization

- Define a map of the scene, with features that can be sensed by the robot.
- Choose N random particle locations (x,y,θ) to cover the scene.
- Place mobile robot in scene (unknown location).
- Until robot is localized do:
  1. Move robot according to known motion model with noise.
  2. Move each particle with similar motion using known motion model with noise.
  3. Compare physical sensor readings with simulated sensor readings from each particle at the state, given:
     - We know each particle’s location (state).
     - We have a noise model of the sensor.
     - We have a known map with feature locations (walls/obstacles/landmarks).
  4. Use comparison from (3) to generate an “importance weight” for each particle, i.e. how close particle estimated states match the sampled measurement.
  5. Resample the particles (with replacement) according to the new weighted distribution from (4). Higher weights mean more agreement with the physical sensor measurement, and a more likely location estimation for the robot.
  6. Repeat steps (1-5) with the newly sampled particle set until robot particles converge.
- After each motion update, particles that are close to the actual robot location will have their sensor measurements more consistent with the physical readings, reinforcing these particles.
- Particles that were not close to the actual robot location after the movement update will not be consistent with sensor measurements and will be less likely to survive during resampling.

Monte Carlo Localization

- Example
  http://www.hessmer.org/robotics/monte-carlo-location-for-robots/monte-carlo-localization-implementation.html