Lecture 13: Filtering II

Professor Katie Driggs-Campbell

February 29, 2024

ECE484: Principles of Safe Autonomy

Administrivia

sign up to be posted today!

- Upcoming due dates:
  - HW2 and MP2 due Friday 3/01
  - HW3 and MP3 due Friday 3/22 (after break)
  - Project Pitches in class 3/05 and 3/07
    - Presentation template on website
    - Sign-up for ordering will be posted later today (1 extra point for going on Tuesday)
- Bonus MP will be worth 1% of your overall grade
- After break, no formal labs – transition to project support

Today's Plan

- Wrap up Bayes Filters (discrete)
- Particle Filters (non-parametric)
- Kalman Filters (continuous, linear)

Today’s Plan

- Wrap up Bayes Filters (discrete)
- Particle Filters (non-parametric)
- Kalman Filters (continuous, linear)

Bayes’s Formula

Recall: $P(x, y) = P(x|y)P(y) = P(y|x)P(x)$

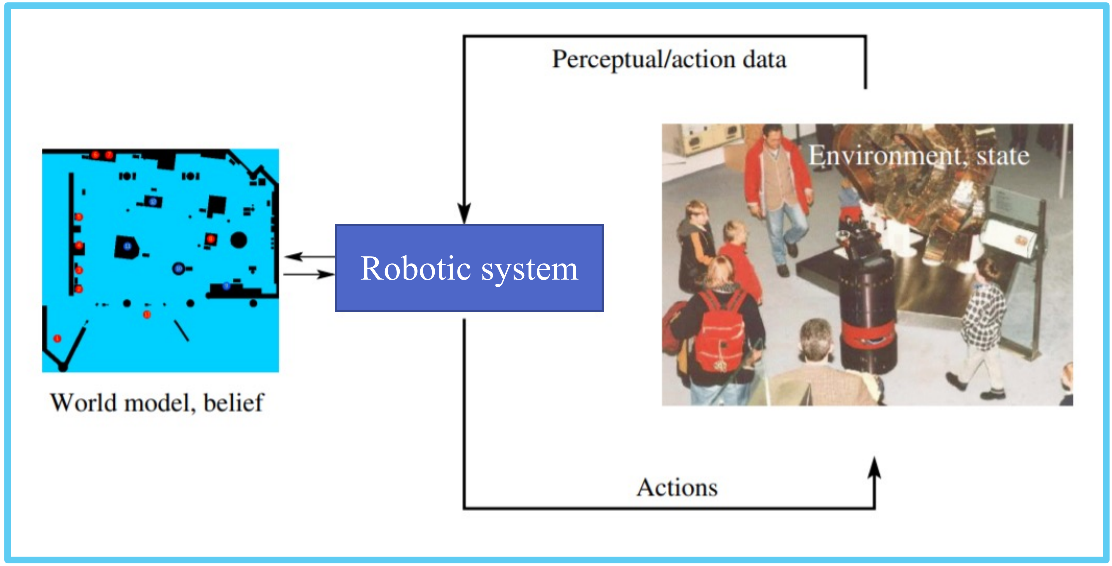
$$P(\overset{\text{state to est}}{x}|\overset{\text{sensor info}}{y}) = \frac{\overbrace{P(y|x)}^{\text{sensor model}}\left.P(x)\right\}\text{prior}}{\left.P(y)\right\}\text{evidence}}$$

Robot’s Belief over States

Belief: Robot’s knowledge about the state of the environment

$bel(x_t) = p(x_t | z_{1:t}, u_{1:t})$

Posterior distribution over state at time *t* given all past measurements and control

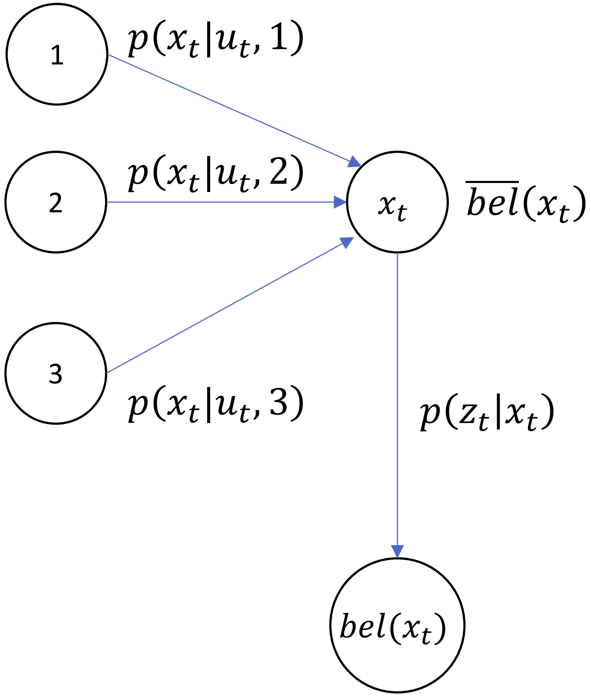


Prediction: $\overline{bel}(x_t) = p(x_t | z_{1:t-1}, u_{1:t})$

Calculating $bel(x_t)$ from $\overline{bel}(x_t)$ is called **correction** **or** **measurement update**

Recursive Bayes Filter

```
**Algorithm Bayes_Filter**($bel(x_{t-1}), u_t, z_t$)
for all $x_t$ do:
        $\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t|u_t, x_{t-1}) bel(x_{t-1})$
        $bel(x_t) = \eta\, p(z_t|x_t)\, \overline{bel}(x_t)$
end for
return $bel(x_t)$
```



Bayes Filters (1)

$bel(x_0 = O) = bel(x_0 = C) = .5$

sensor model

$p(z_t \mid x_t)$

↳ so
sc

z_t \ x_t	O	C
so	.6	.2
sc	.4	.8

dyn model

$p(x_t \mid u_t, x_{t-1})$

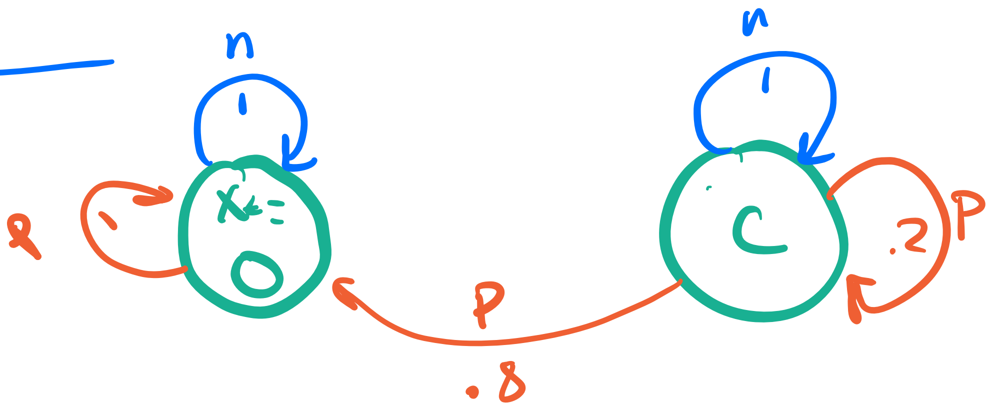
↳ n, P

$u_t = n$

x_t \ x_{t-1}	O	C
O	1	0
C	0	1

$u_t = P$

x_t \ x_{t-1}	O	C
O	1	.8
C	0	.2



Bayes Filters (2)

@ $t=1$, do nothing $u_1 = n$

$$\overline{bel}(x_1) = \sum_{x_0} p(x_1 \mid \overset{=n}{u_1}, x_0)\, bel(x_0)$$

$$= p(x_1 \mid n, o)\, bel(o) + p(x_1 \mid n, c)\, bel(c)$$

suppose $x_1 = o$, plug in:

$$\overline{bel}(x_1 = o) = p(o \mid n, o)\, bel(o) + p(o \mid n, c)\, bel(c)$$

$$= .5$$

suppose $x_1 = c$, plug in!

$$\overline{bel}(x_1 = c) = p(c \mid n, o)\, bel(o) + p(c \mid n, c)\, bel(c)$$

$$= .5$$

Bayes Filters (3)

recall: $bel(x) = \eta \, p(z|x) \, \overline{bel}(x)$

suppose $z_1 = so$

1) $x_1 = o$

$bel(x_1 = o) = \eta \, p(so|o) \, \overline{bel}(o)$

$= \eta \cdot .6 \cdot .5 = \eta \cdot .3 = .75$

2) $x_1 = c$

$bel(x_1 = c) = \eta \, p(so|c) \, \overline{bel}(c)$

$= \eta \cdot .2 \cdot .5 = \eta \cdot .1 = .25$

normalizer: $\eta = \frac{1}{.3 + .1} = 2.5$

Bayes Filters (4)

now push! + get sensor update

$u_2 = p$ $z_2 = so$

$$\overline{bel}(x_2=o) = 1 \cdot .75 + .8 \cdot .25 = .95$$

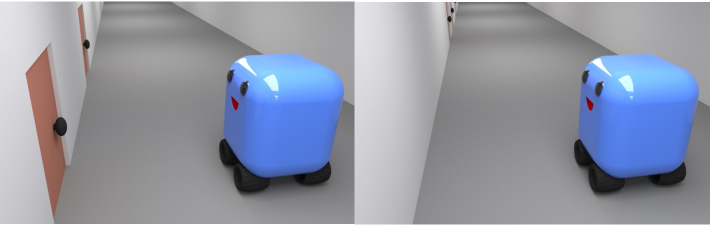
$$\overline{bel}(x_2=c) = 0 \cdot .75 + .2 \cdot .25 = .05$$

$$bel(x_2=o) = \eta \cdot .6 \cdot .95 \approx .983$$

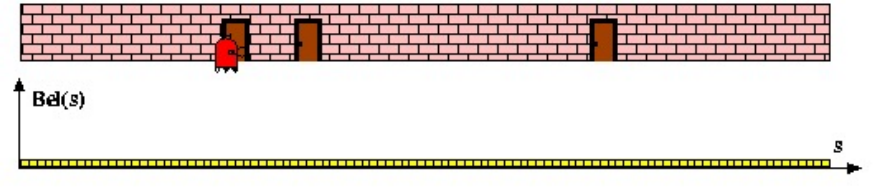
$$bel(x_2=c) = \eta \cdot .2 \cdot .05 \approx .017$$

Bayes Filters (5)

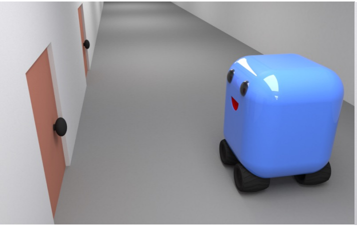
Discrete Bayes Filter - Illustrati

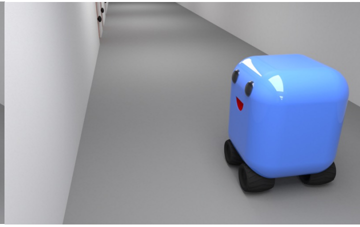


time = 0

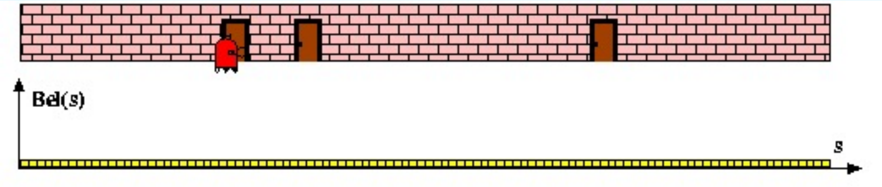


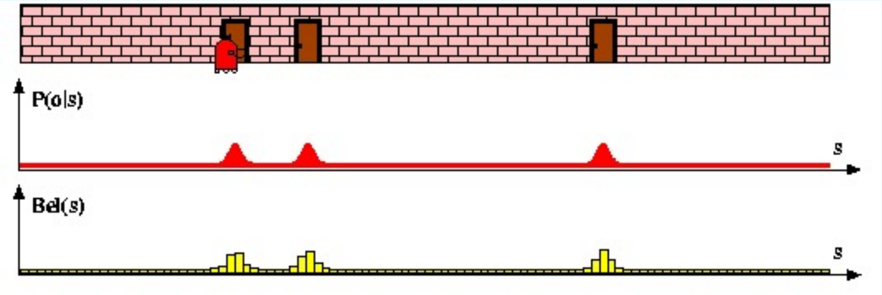
Discrete Bayes Filter - Illustrati



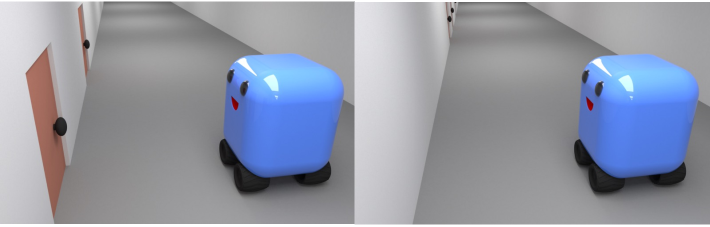


time = 0

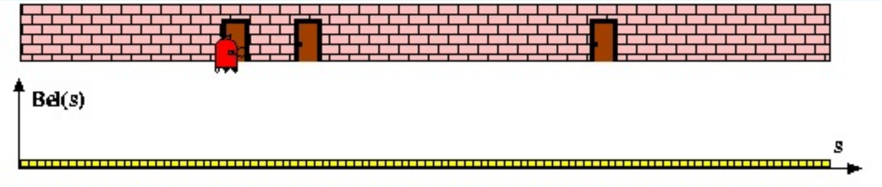


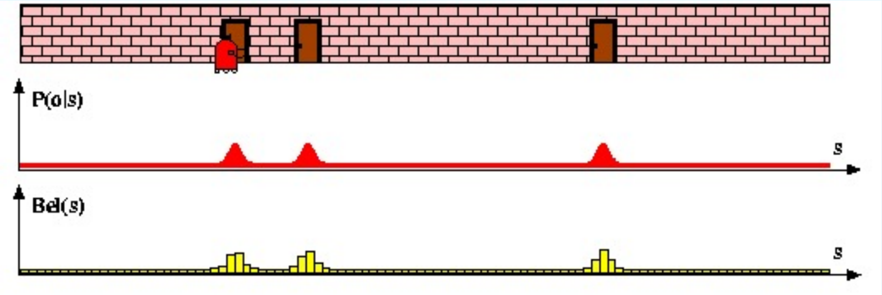


Discrete Bayes Filter - Illustrati

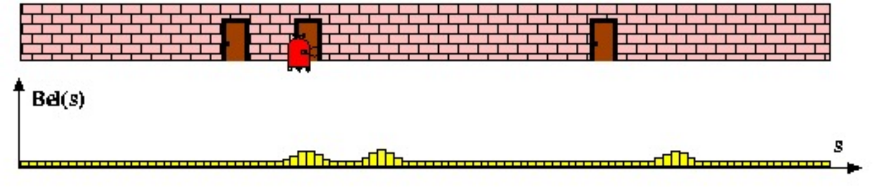


time = 0

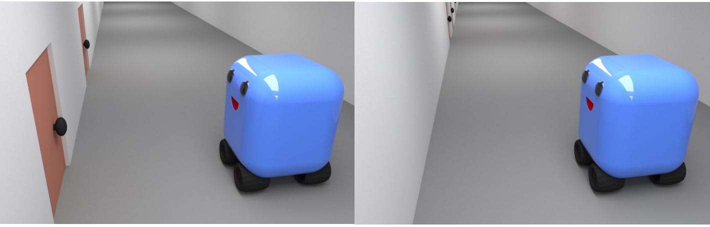




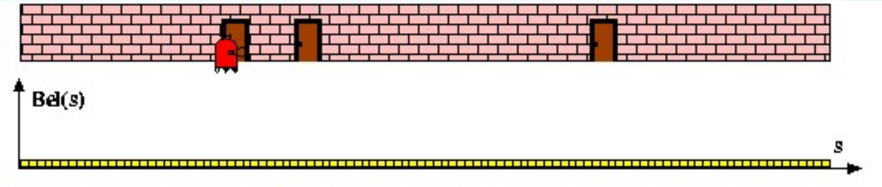
time = 1

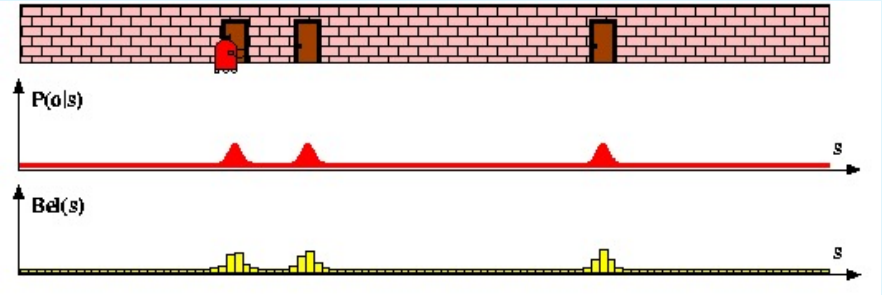


Discrete Bayes Filter - Illustrati

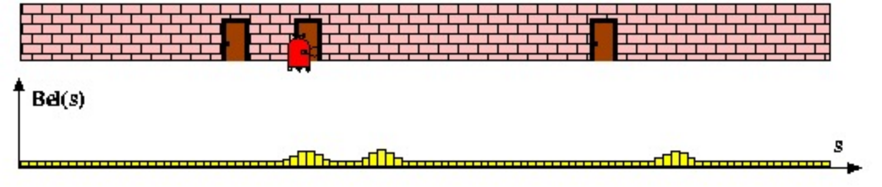


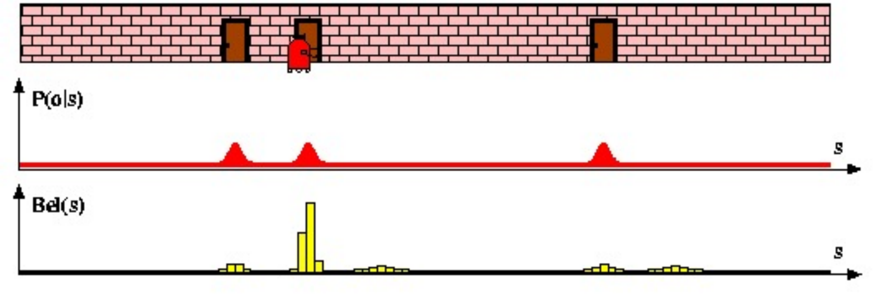
time = 0



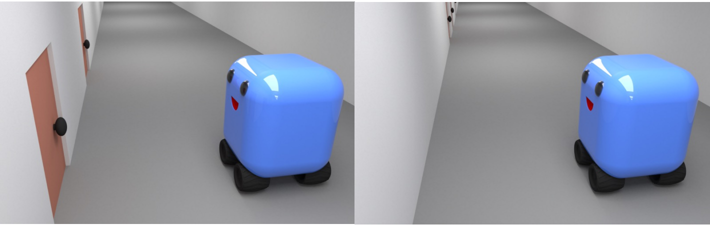


time = 1

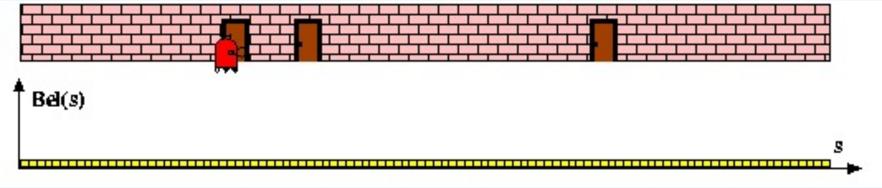


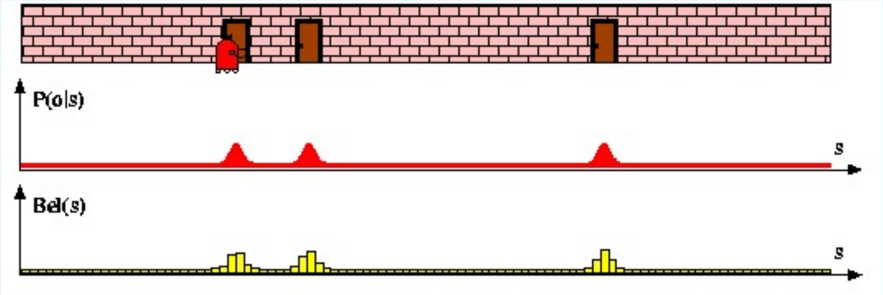


Discrete Bayes Filter - Illustrati

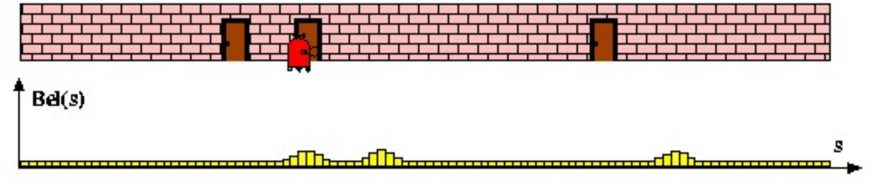


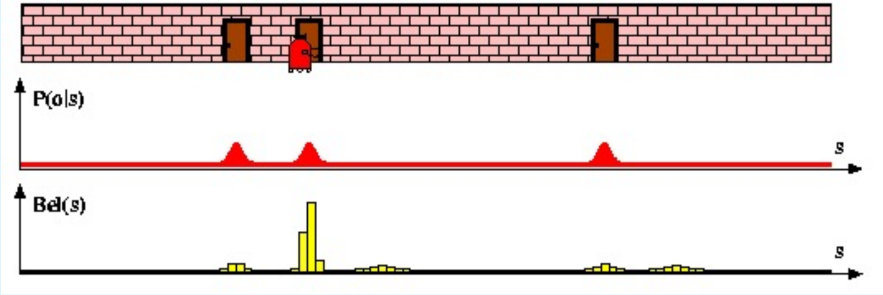
time = 0



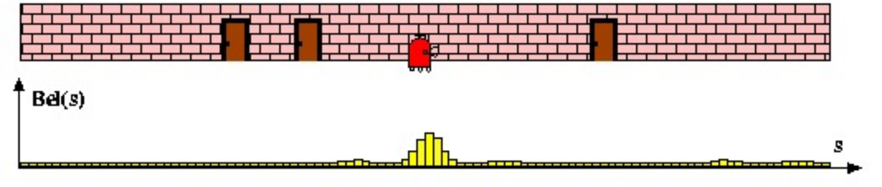


time = 1





time = 2



Bayes Filter Recap

$$\eta p(z_t|x_t)\int p(x_t|\, u_t, x_{t-1})bel(x_{t-1})dx_{t-1}$$

- Prediction

$$\overline{bel}(x_t) = \int p(x_t|\, u_t, x_{t-1})bel(x_{t-1})dx_{t-1}$$

- Correction

$$bel(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t)$$

What if we have a good model of our (continuous) system dynamics and we assume a Gaussian model for our uncertainty?

→ Kalman Filters!

Midway Summary

- Bayes filters are a probabilistic tool for **estimating the state of dynamic systems**
  - **They are everywhere!** Kalman filters, Particle filters, Hidden Markov models, Dynamic Bayesian networks, Partially Observable Markov Decision Processes (POMDPs), …
  - Bayes rule allows us to compute probabilities that are hard to assess otherwise
  - Recursive Bayesian updating can efficiently combine evidence over time

Today's Plan

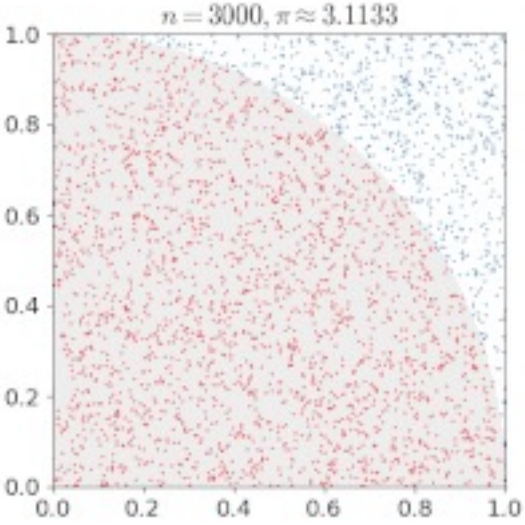
- Wrap up Bayes Filters (discrete)
- Particle Filters (non-parametric)
- Kalman Filters (continuous, linear)

Fun facts about Monte Carlo Methods

… a question which occurred to me in 1946 as I was **convalescing from an illness and playing solitaires. The question was what are the chances that a Canfield solitaire laid out with 52 cards will come out successfully?** After spending a lot of time trying to estimate them by pure combinatorial calculations, I wondered whether a more practical method than "abstract thinking" might not be to **lay it out say one hundred times and simply observe and count the number of successful plays. This was already possible to envisage with the beginning of the new era of fast computers,** and I immediately thought of problems of neutron diffusion and other questions of mathematical physics, and more generally how to change processes described by certain differential equations into an equivalent form interpretable as a succession of random operations. Later [in 1946], I described the idea to John von Neumann, and we began to plan actual calculations. - *Stanislaw Ulam, ~1940s*

Monte Carlo methods vary, but tend to follow this pattern:

1. Define a domain of possible inputs
2. Generate inputs randomly from a probability distribution over the domain
3. Perform a deterministic computation on the inputs
4. Aggregate the results



Monte Carlo methods were central to the **simulations required for the Manhattan Project**, and more recently has been extended to Sequential Monte Carlo in advanced signal processing and Bayesian inference. Also, the extension **MC Tree Search enabled AlphaGo.**

Particle Filters

- Particle filters are an implementation of recursive Bayesian filtering, where the posterior is represented by a set of weighted samples
- Instead of a precise probability distribution, represent belief $bel(x_t)$ by a set of particles, where each particle tracks its own state estimate
- Random sampling used in generation of particles
- Particles are periodically re-sampled, with probability weighted by likelihood of last generation of particles
- Nonparametric filter – can approximate complex probability distributions without explicitly computing the closed form solutions

Particle Filtering Algorithm / / Monte Carlo Localization

```
$X_t = x_t^{[1]}, x_t^{[2]}, \dots x_t^{[M]}$ particles

Algorithm MCL($X_{t-1}, u_t, z_t$,m):
$\bar{X}_{t-1} = X_t = \emptyset$
for all $m$ in [M] do:
        $x_t^{[m]} = \boldsymbol{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$
        $w_t^{[m]} = \boldsymbol{measurement\_model}(z_t, x_t^{[m],m})$
        $\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
end for
for all $m$ in [M] do:
        draw $i$ $\mathit{with\ probability}$ $\propto w_t^{[i]}$
        add $x_t^{[i]}$ $\mathit{to}$ $X_t$
end for
return $X_t$
```

- motion model guides the motion of particles
- $w_t^{[m]}$ is the importance factor or weight of each particle m, which is a function of the measurement model and belief
- Particles are resampled according to weight
- **Survival of the fittest:** moves/adds particles to part of space with higher probability

Particle Filtering Algorithm / / Monte Carlo Localization

$X_t = x_t^{[1]}, x_t^{[2]}, \ldots x_t^{[M]}$ particles

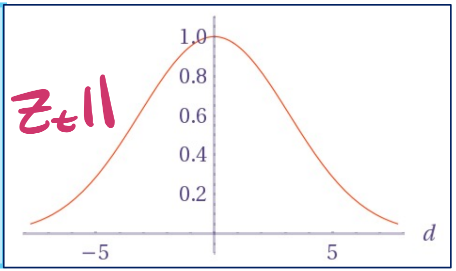
```
Algorithm MCL($X_{t-1}, u_t, z_t$,m):
$\bar{X}_{t-1} = X_t = \emptyset$
for all $m$ in [M] do:
        $x_t^{[m]} = \textbf{\textit{sample\_motion\_model}}(u_t\ x_{t-1}^{[m]})$
        $w_t^{[m]} = \textbf{\textit{measurement\_model}}(z_t, x_t^{[m],m})$
        $\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
end for
for all $m$ in [M] do:
        draw $i$ $with\ probability\ \propto w_t^{[i]}$
        add $x_t^{[i]}$ $to\ X_t$
end for
return $X_t$
```

Step 1: Initialize particles uniformly distribute over space and assign initial weight

Step 2: Sample the motion model to propagate particles

Step 3: Read measurement model and assign (unnormalized) weight: $d = \| z^{[m]} - z_t \|$

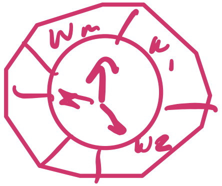
$$w_t^{[m]} = \exp\left(\frac{-d^2}{2\sigma}\right)$$



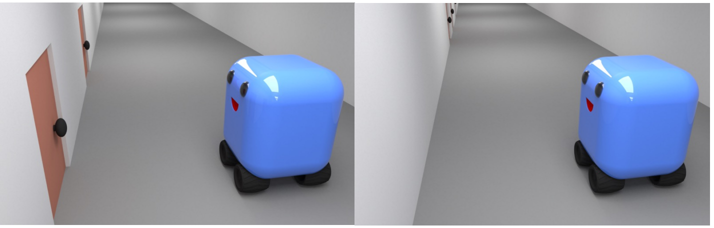
Step 4: Calculate your position update estimate by adding the particle positions scaled by weight

Note that weights must be normalized to sum to 1

Step 5: Choose which particles to *resample* in the next iteration by replacing less likely particles with more likely ones

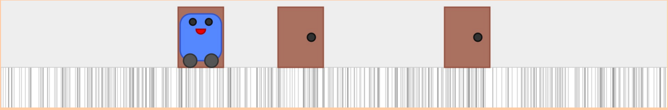


Particle Filter Localization - Illustra



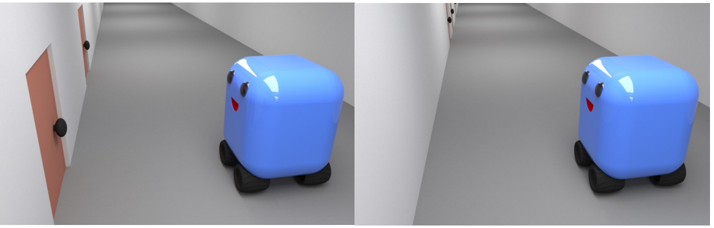
time = 0

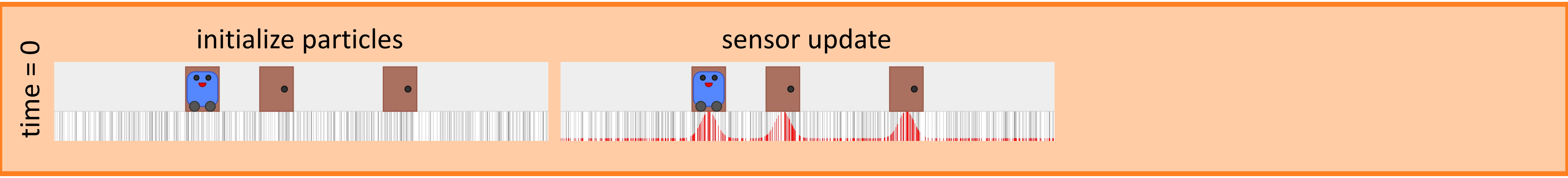
initialize particles



Images By Daniel Lu - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=25252955

Particle Filter Localization - Illustra





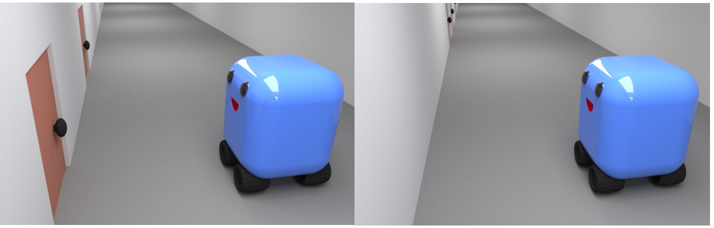
time = 0

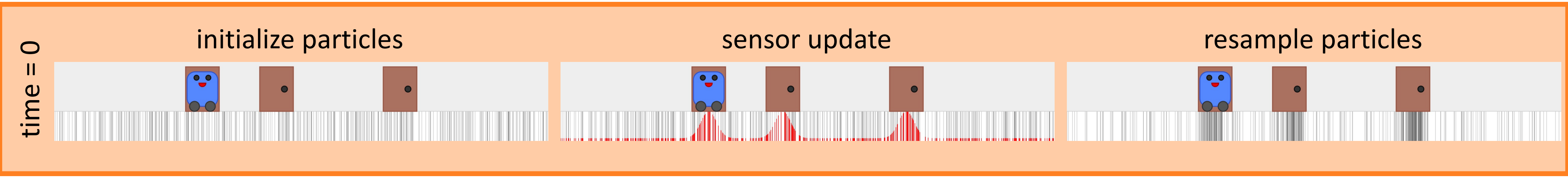
initialize particles

sensor update

Images By Daniel Lu - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=25252955

Particle Filter Localization - Illustra





time = 0

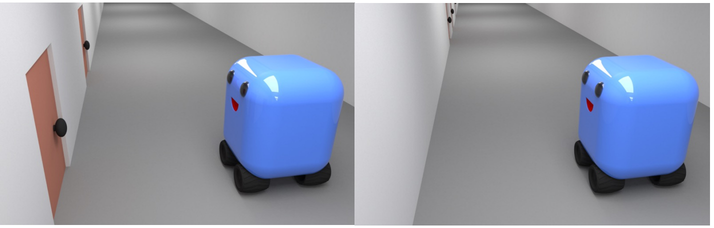
initialize particles

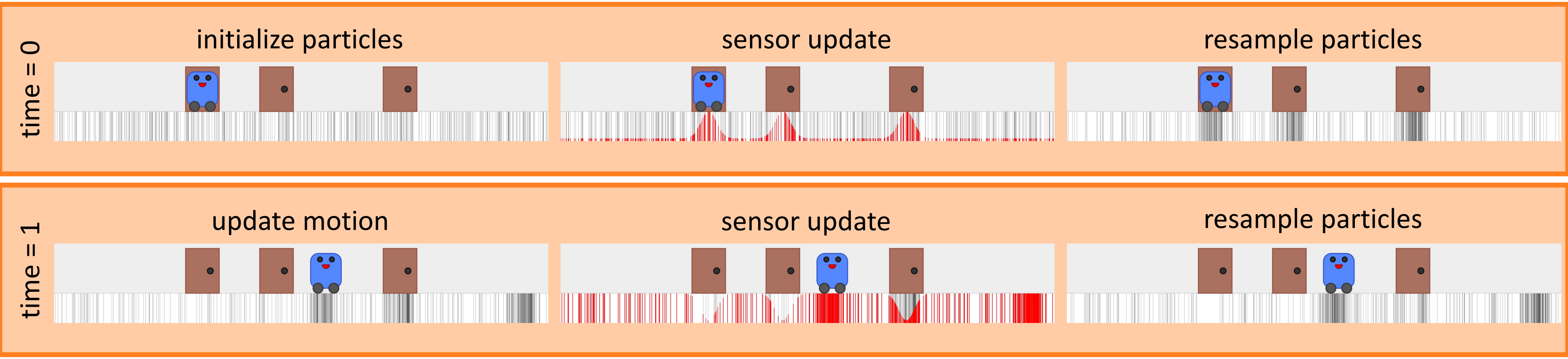
sensor update

resample particles

Images By Daniel Lu - Own work, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=25252955

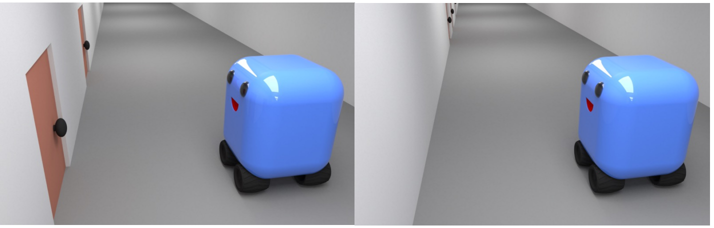
Particle Filter Localization - Illustra

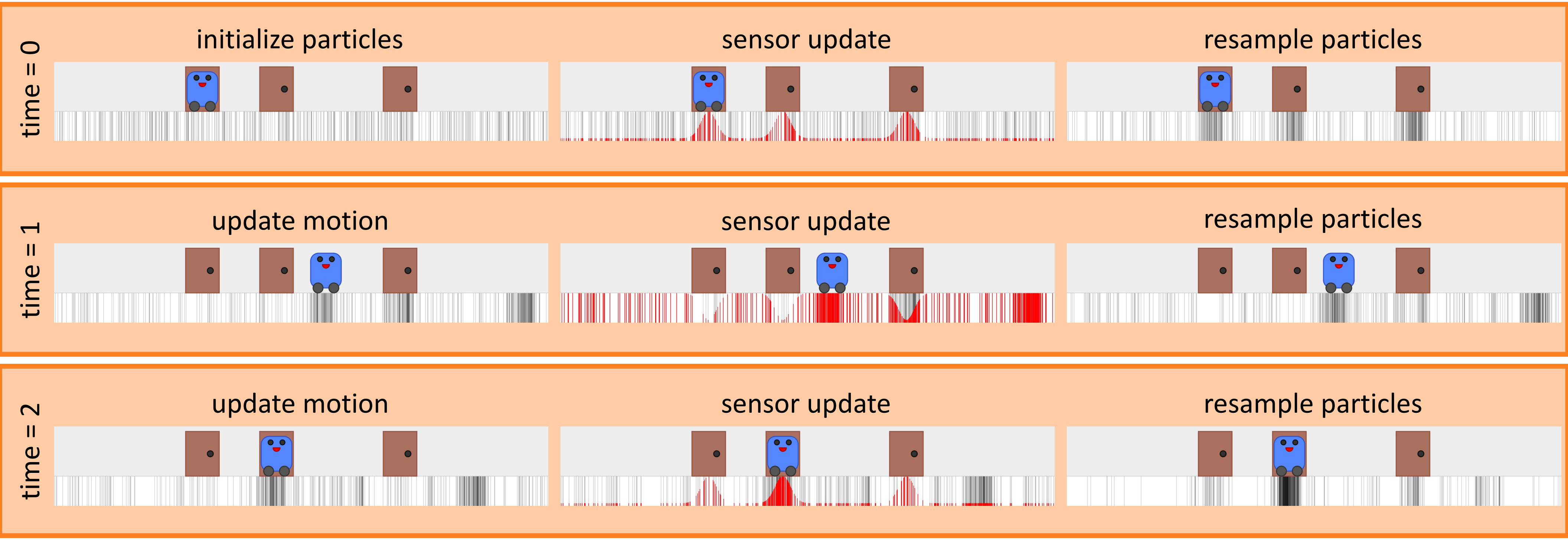




Images By Daniel Lu - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=25252955

Particle Filter Localization - Illustra





time = 0

initialize particles

sensor update

resample particles

time = 1

update motion

sensor update

resample particles

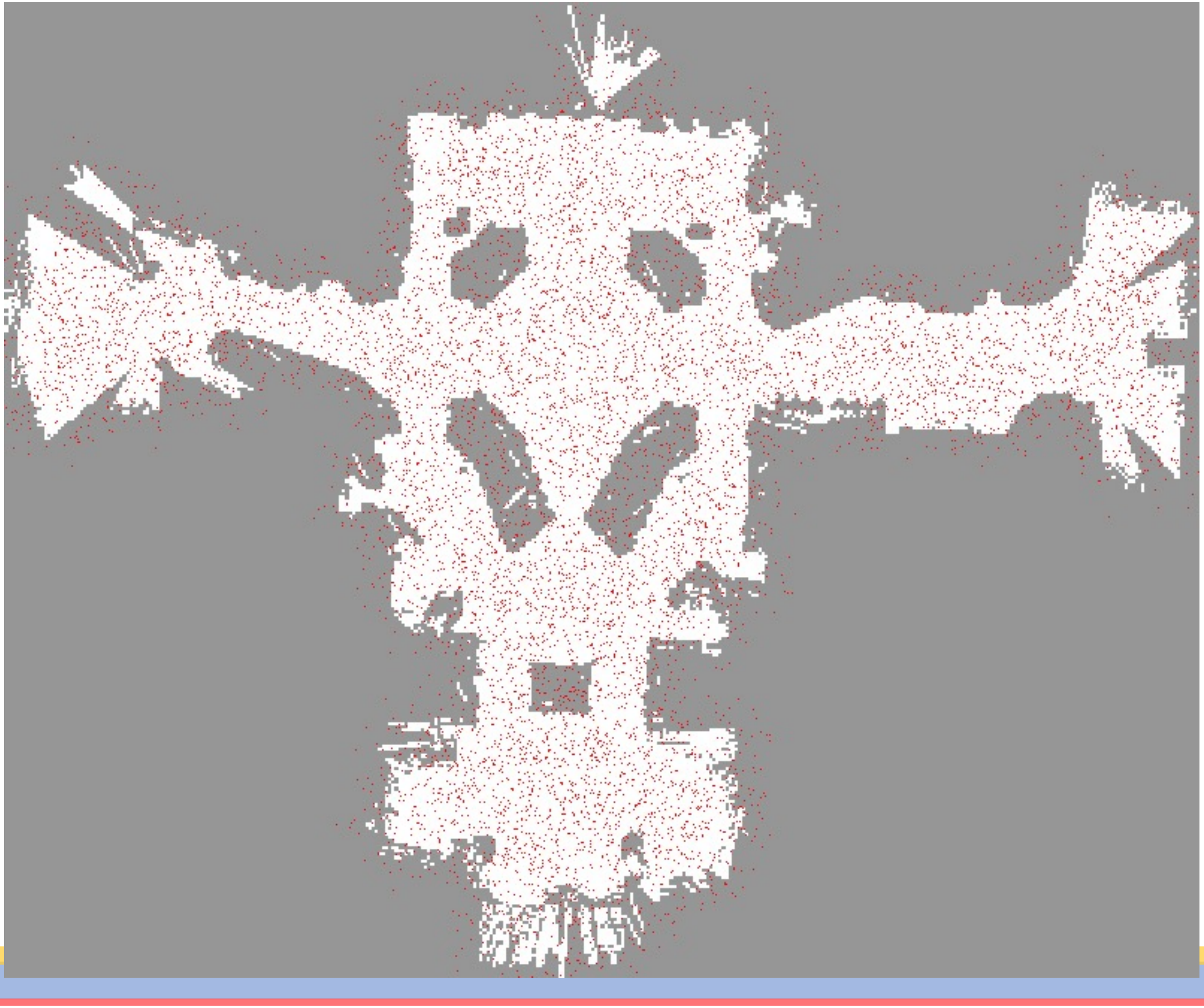
time = 2

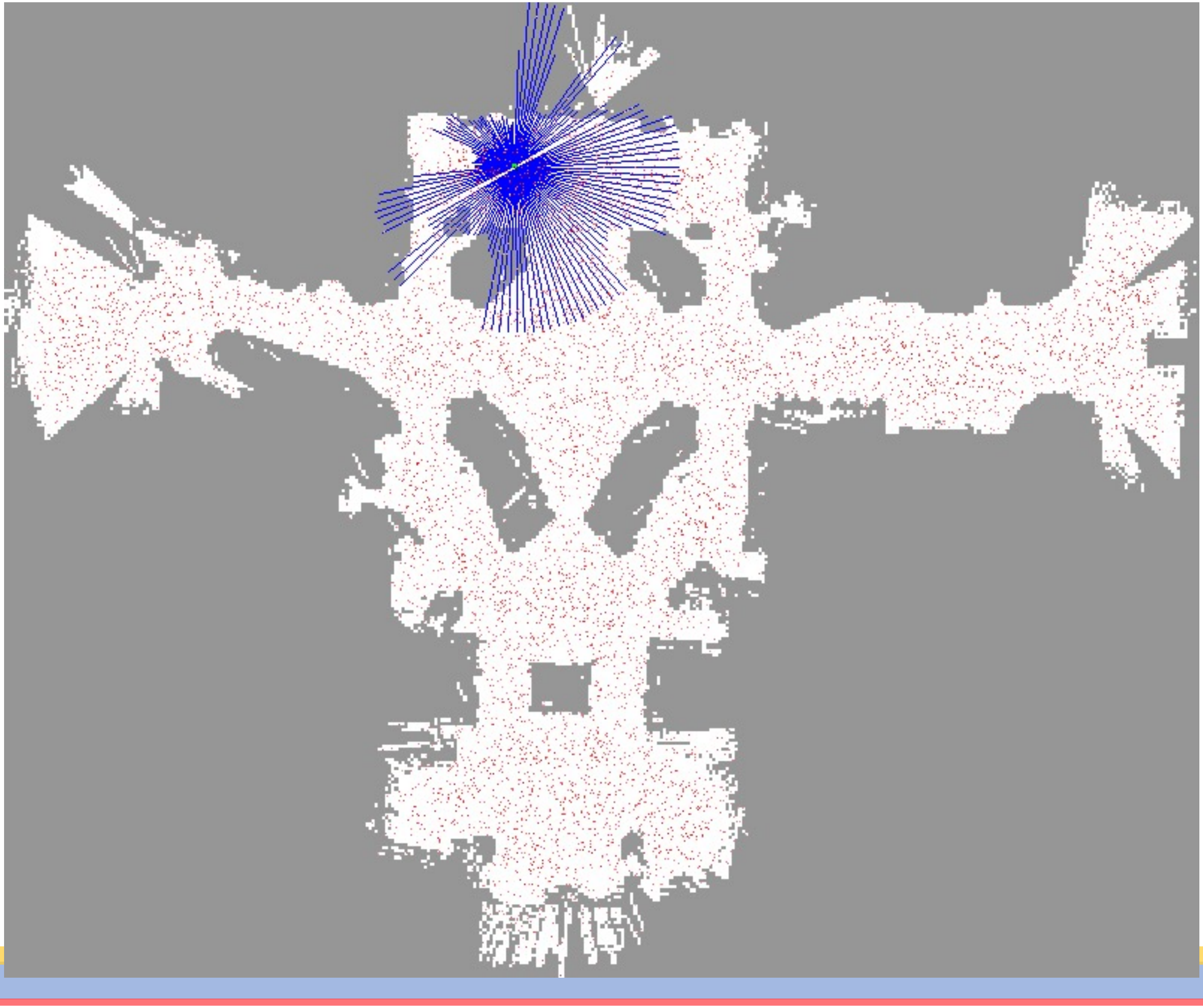
update motion

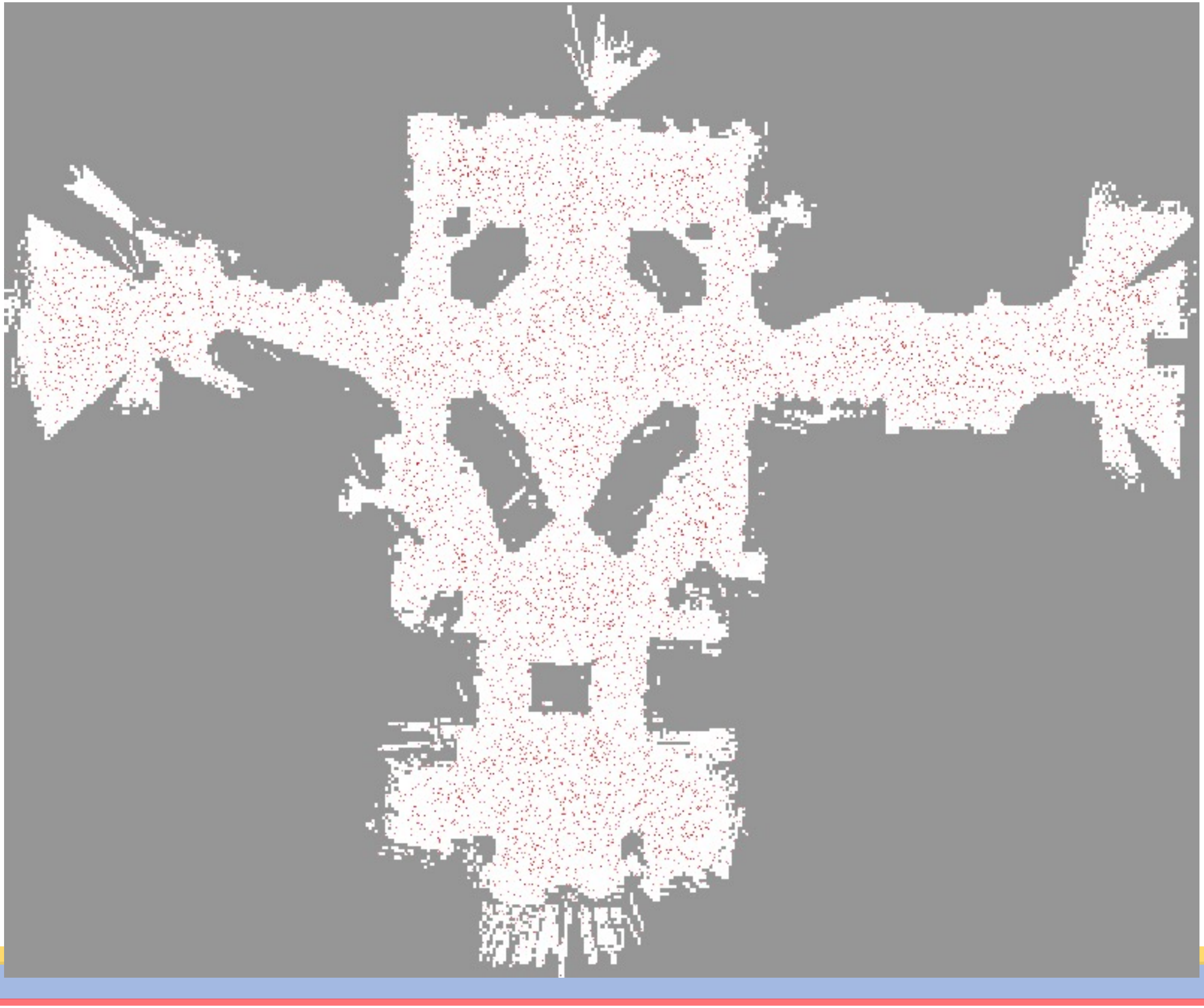
sensor update

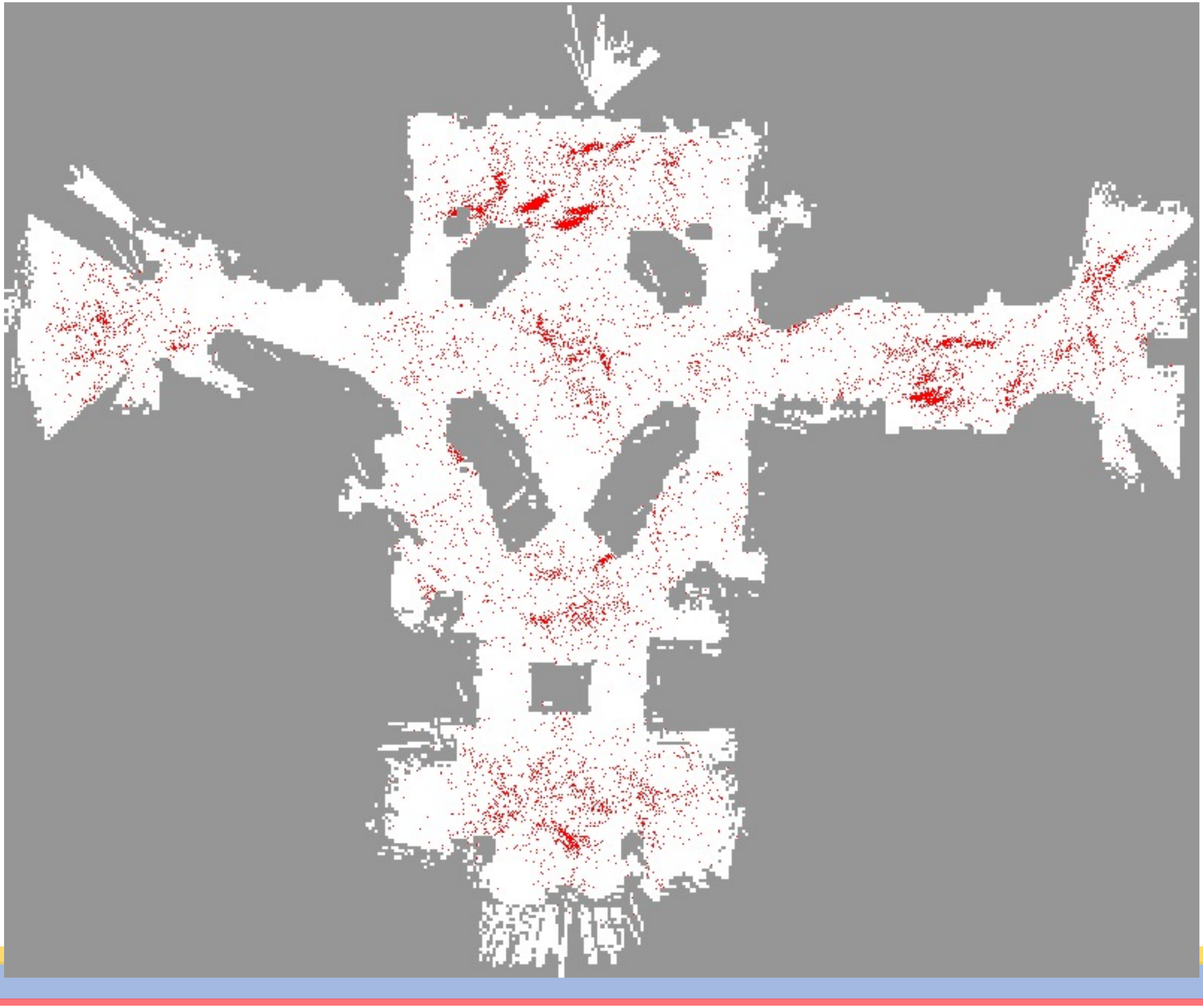
resample particles

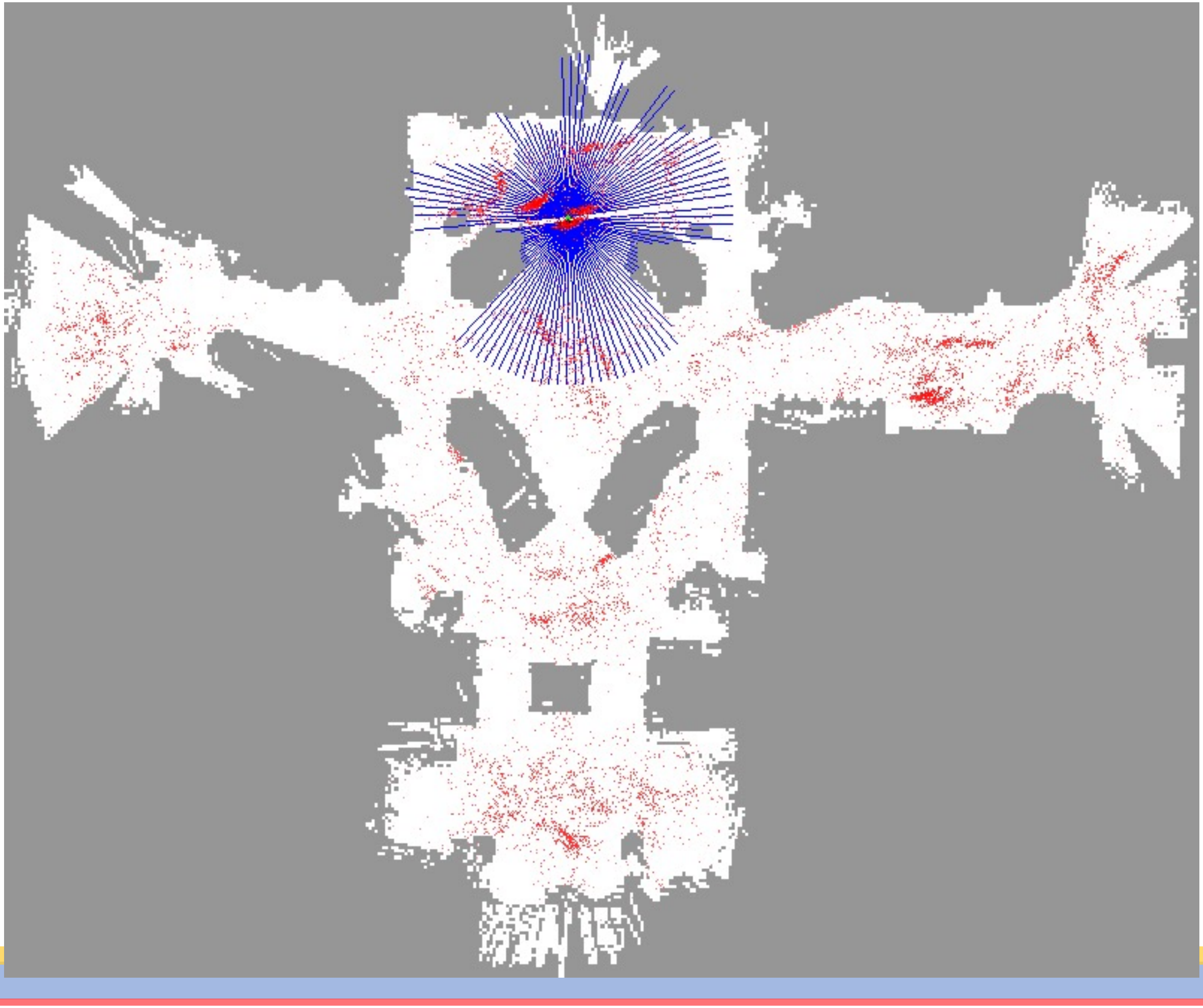
Images By Daniel Lu - Own work, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=25252955

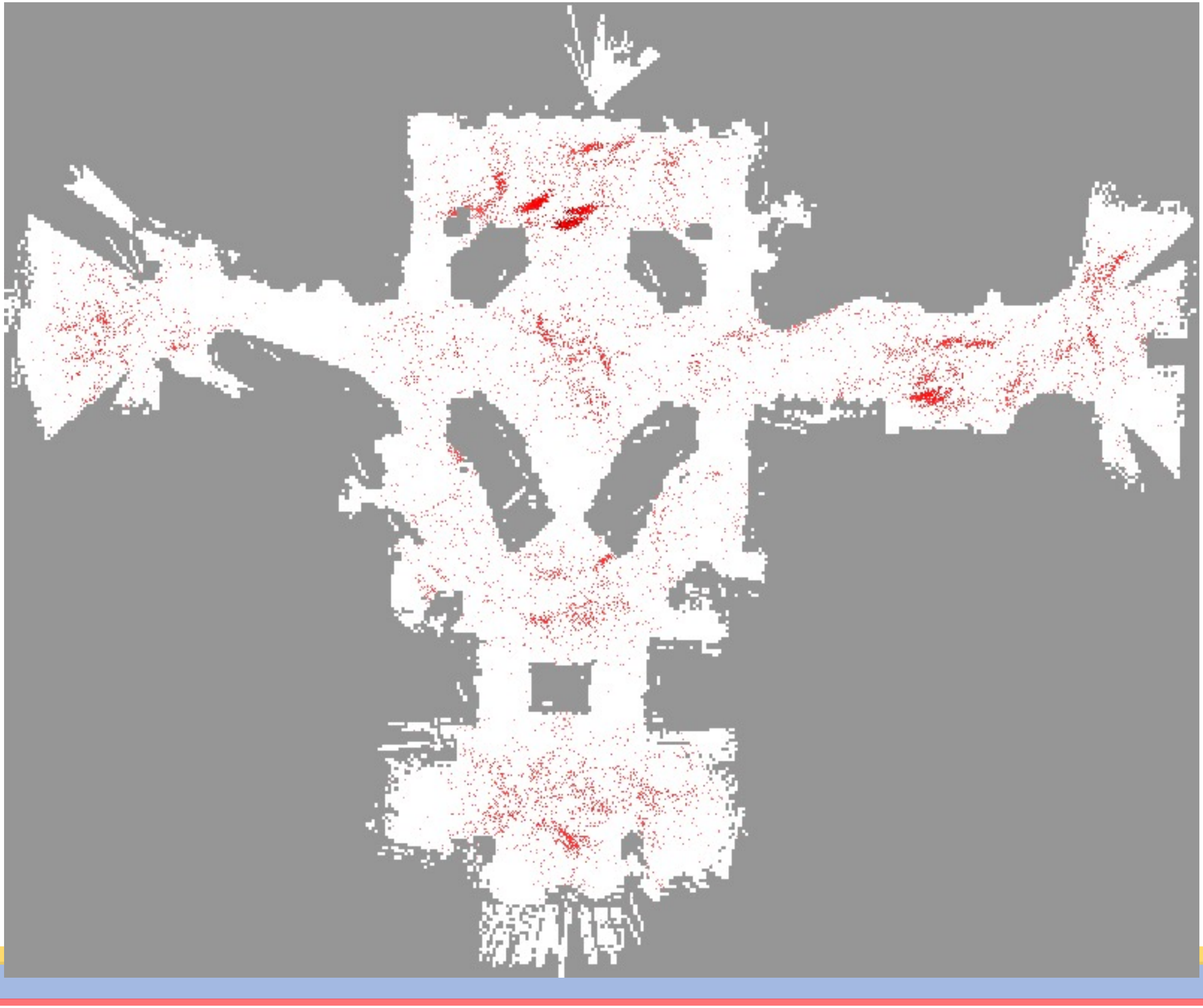


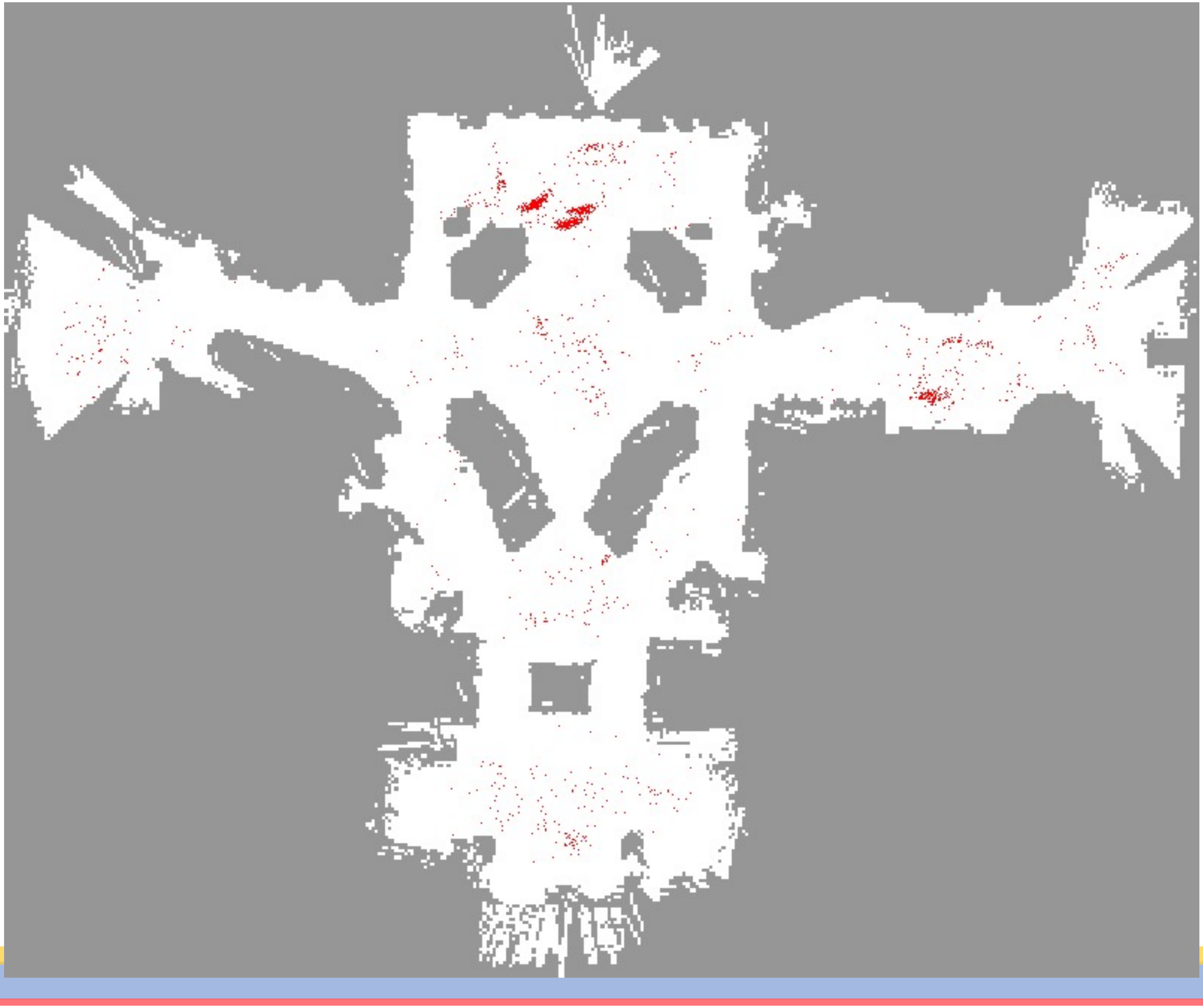


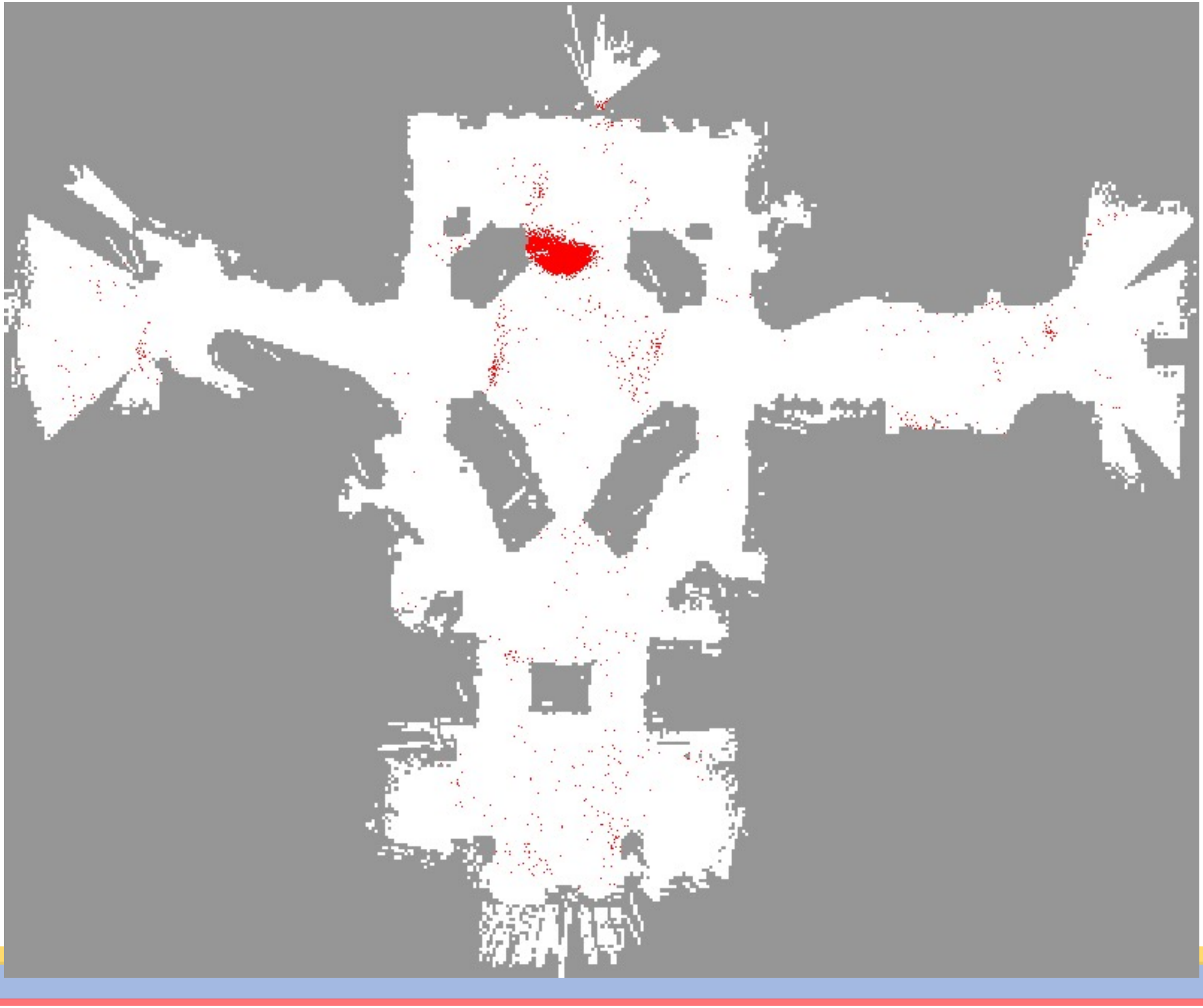


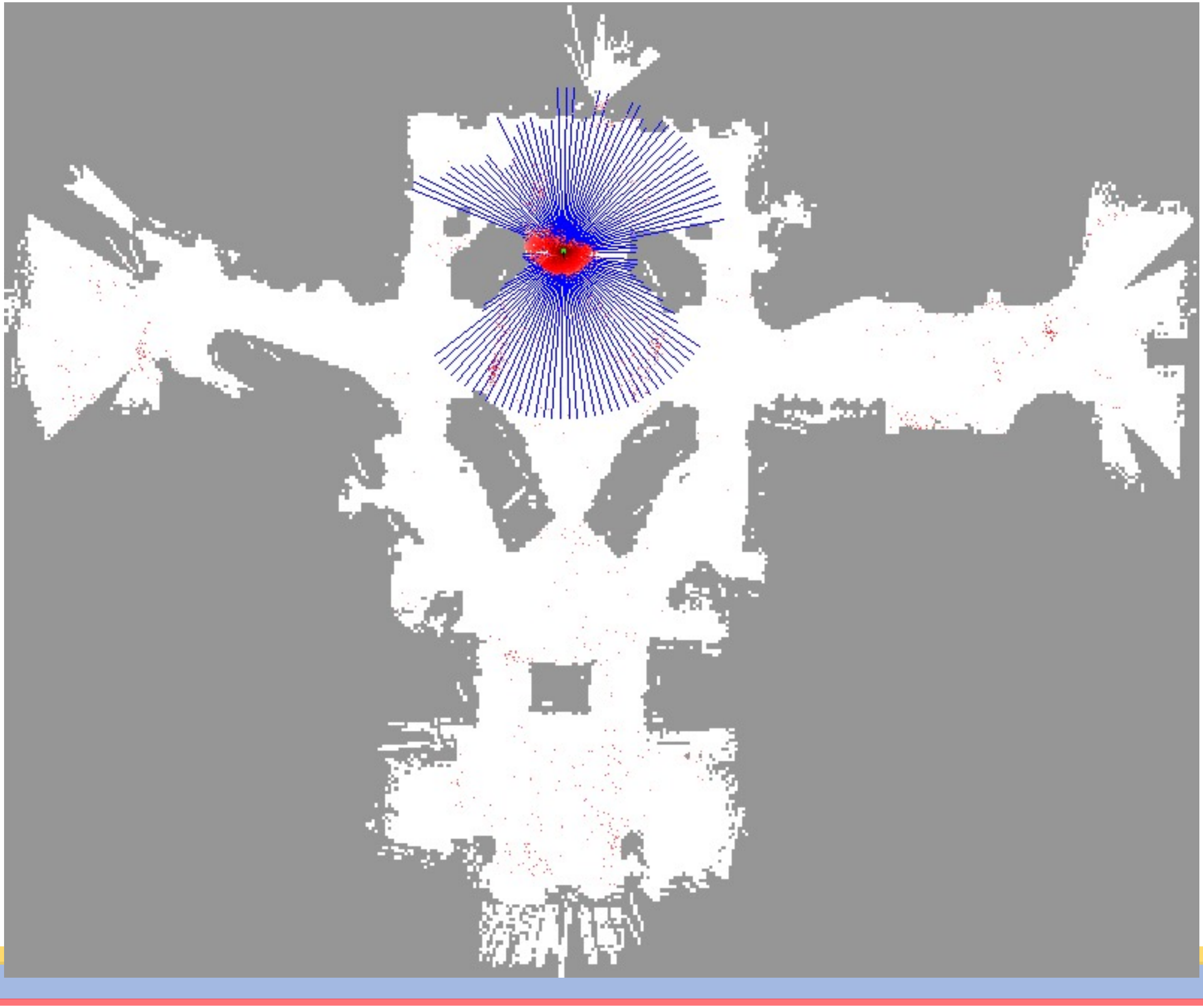


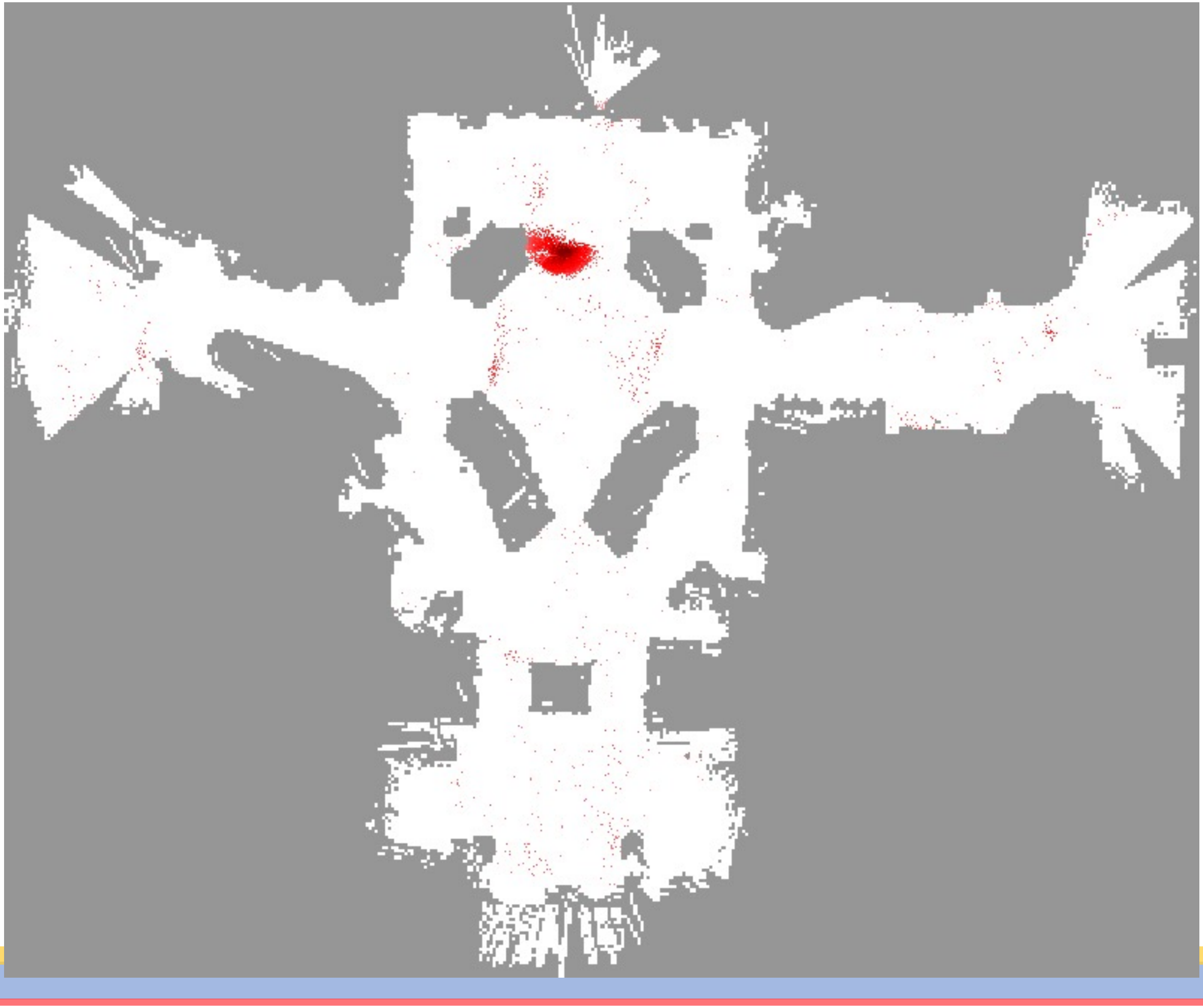


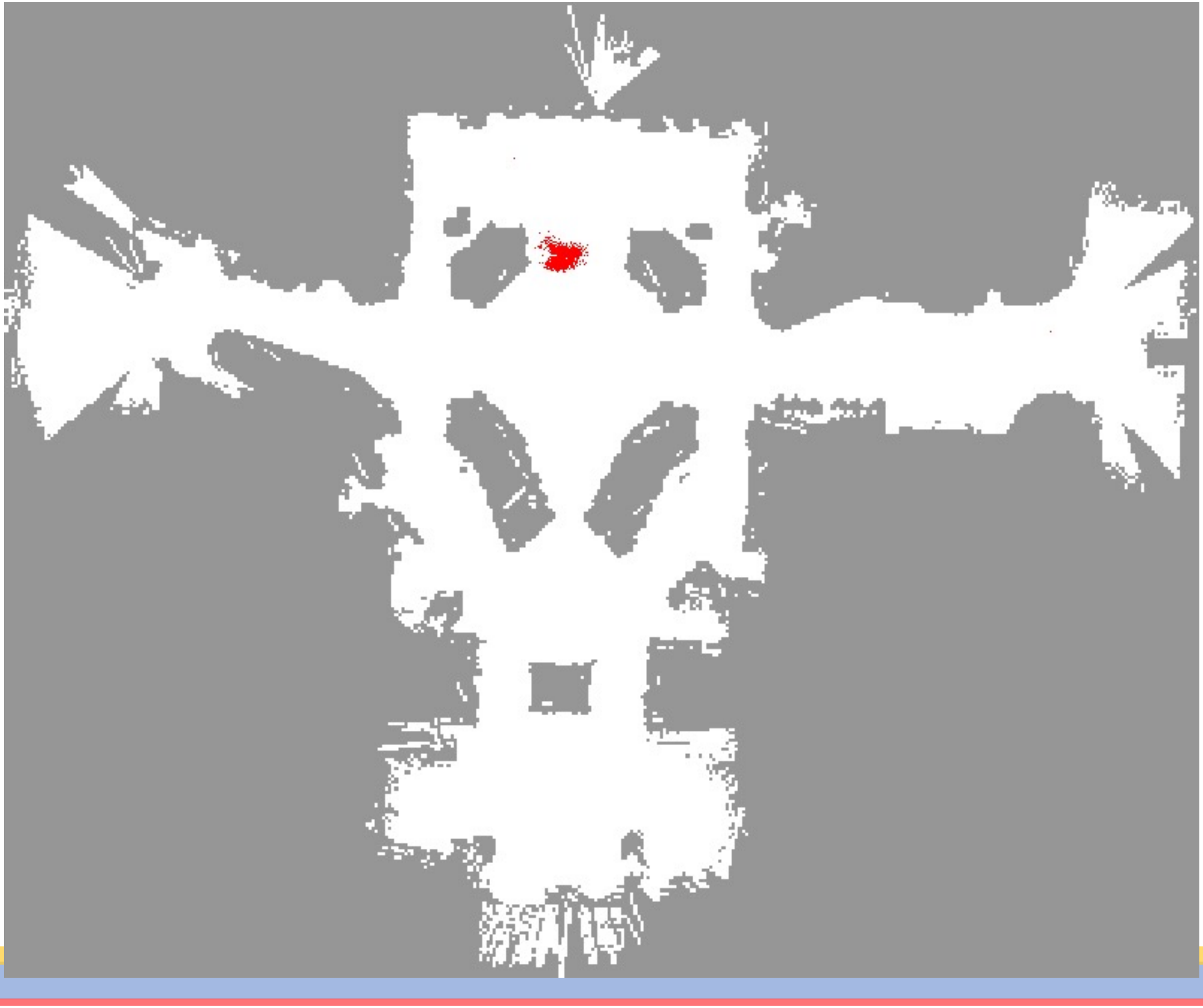


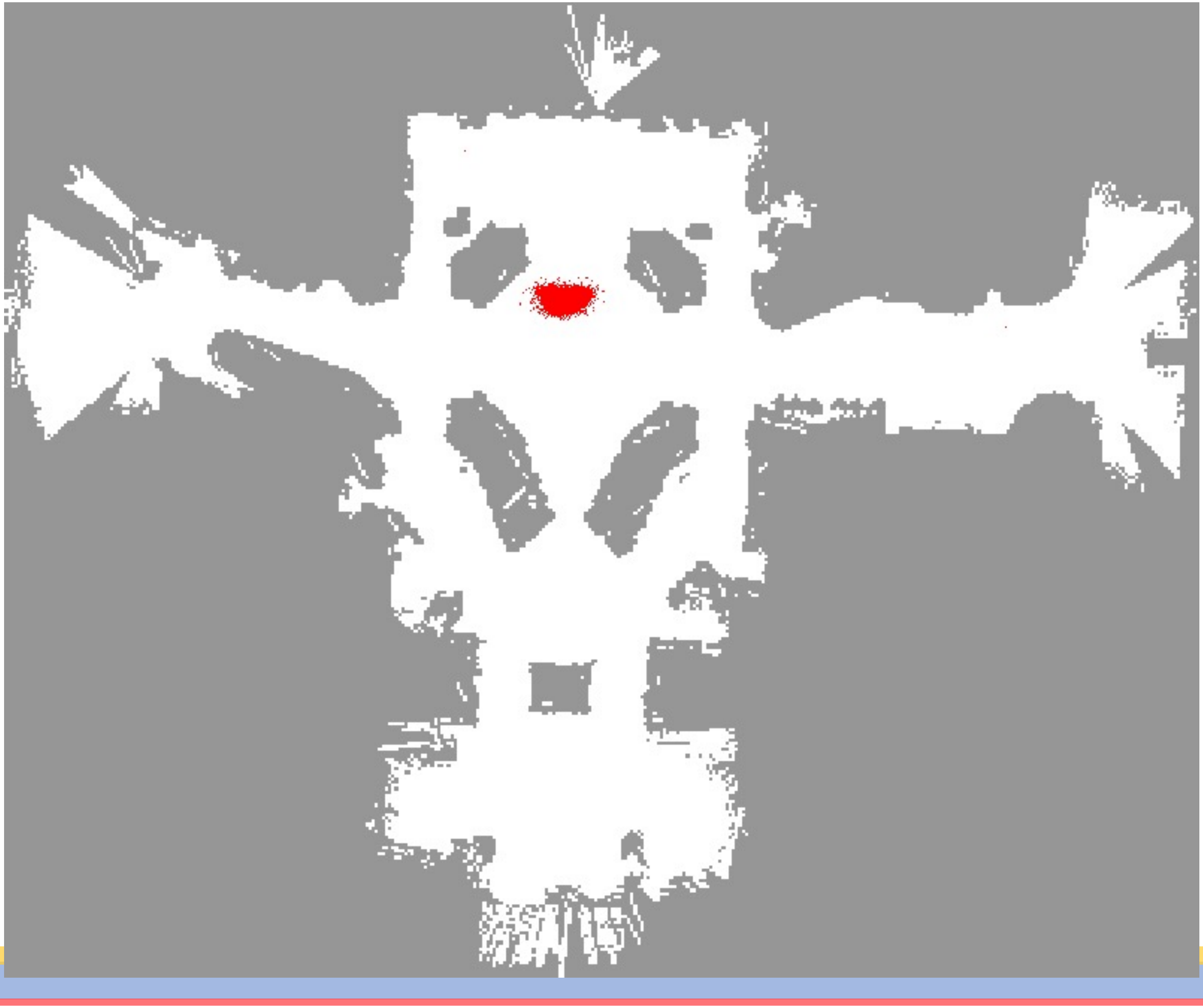


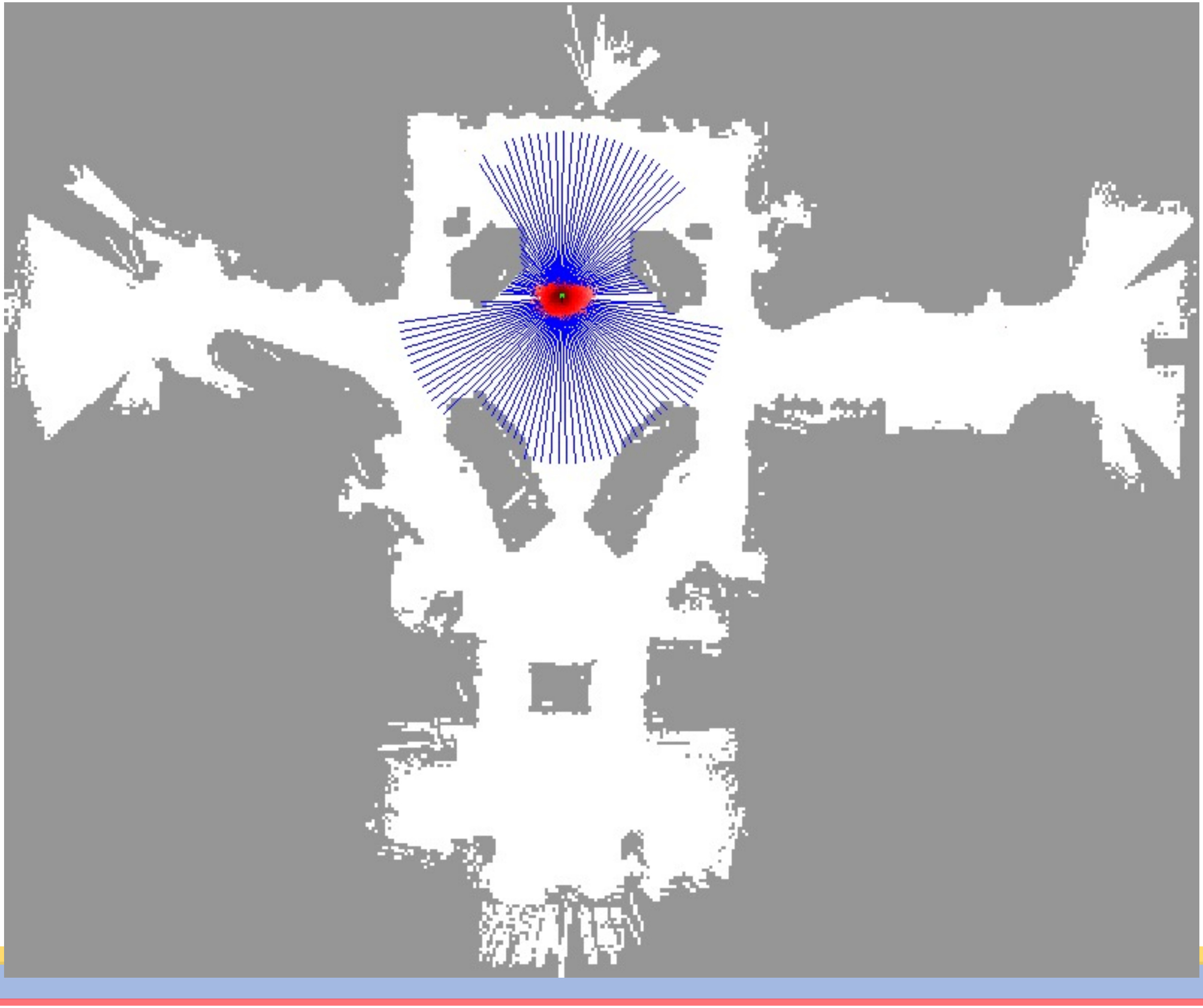


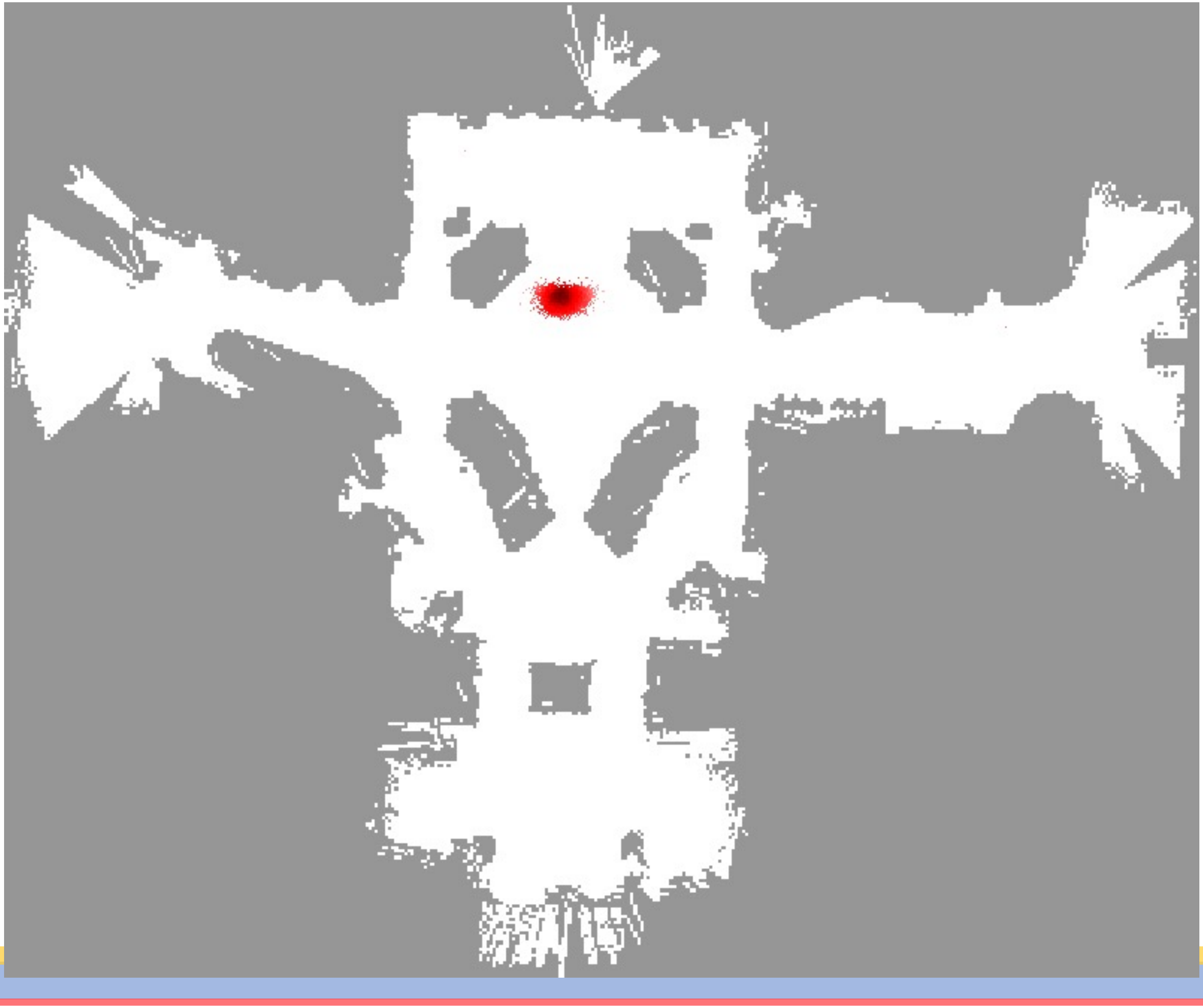


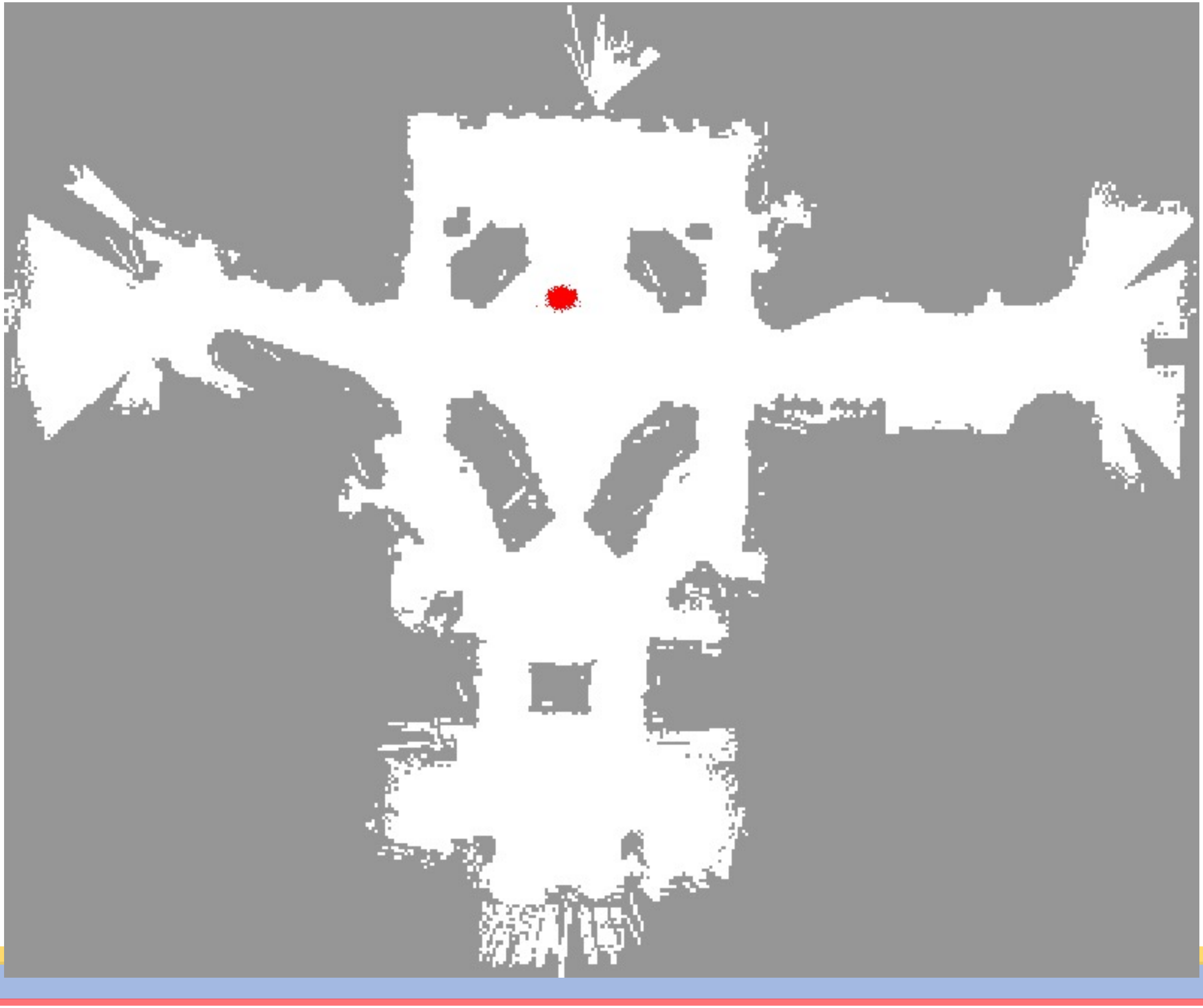


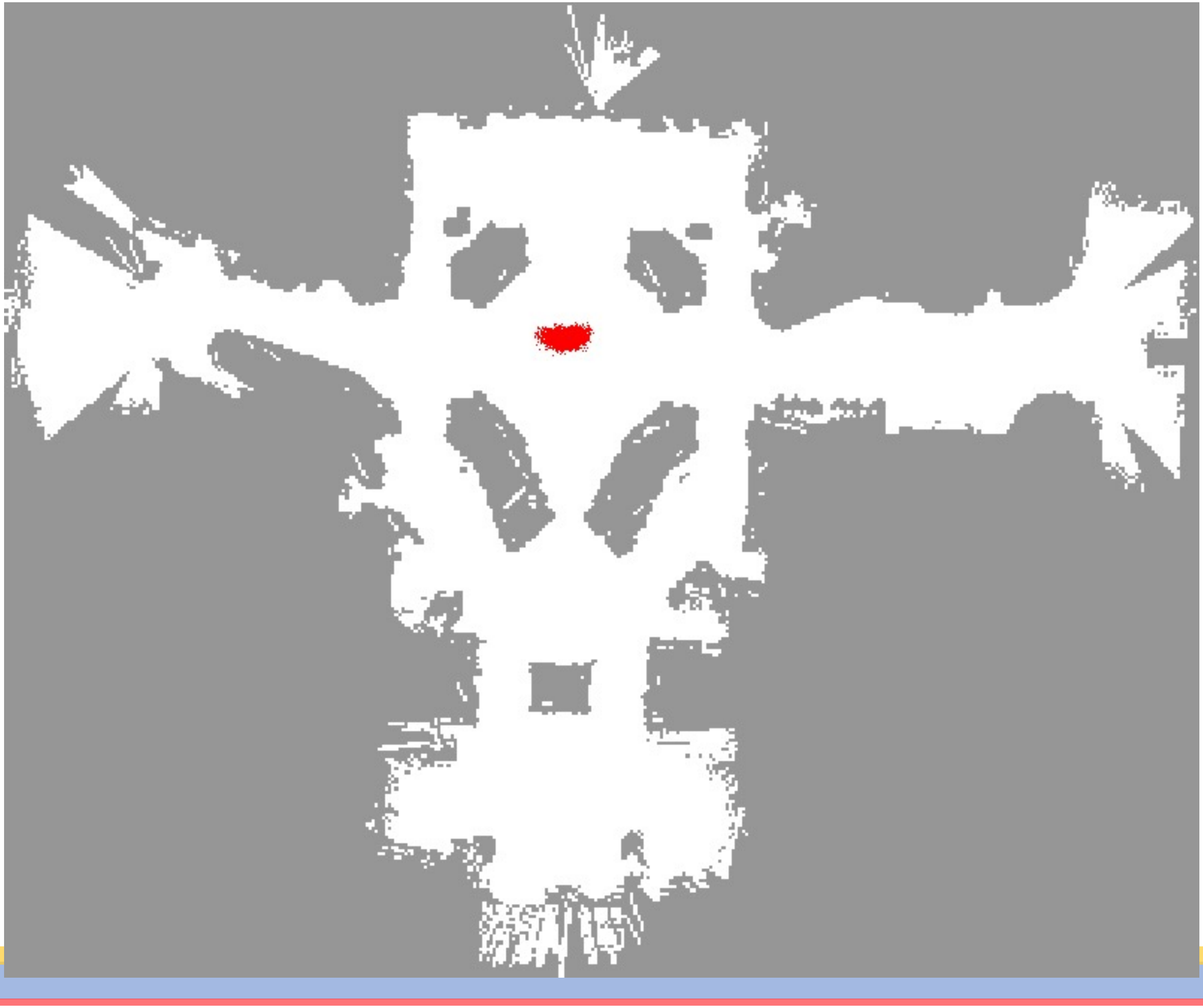


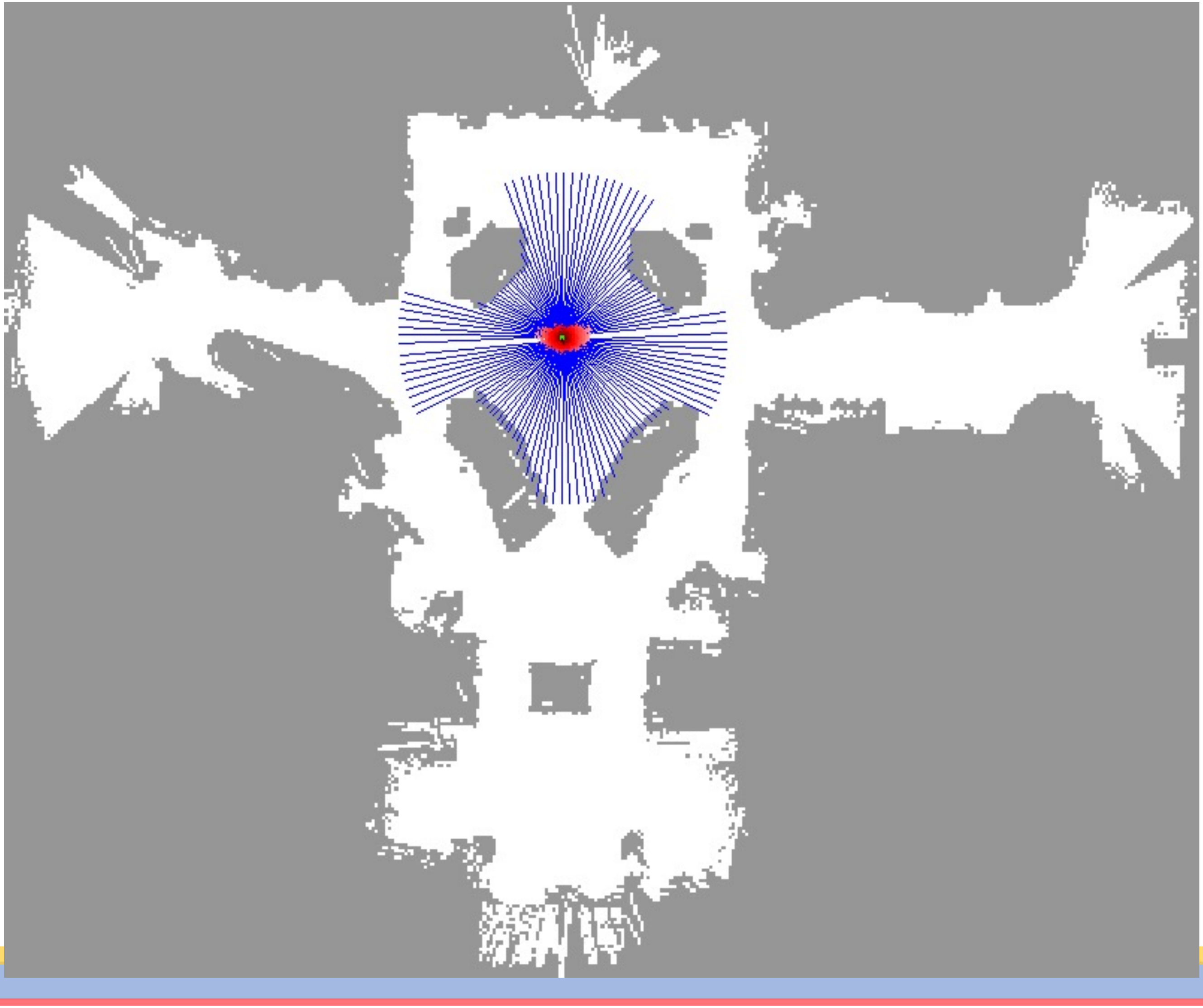


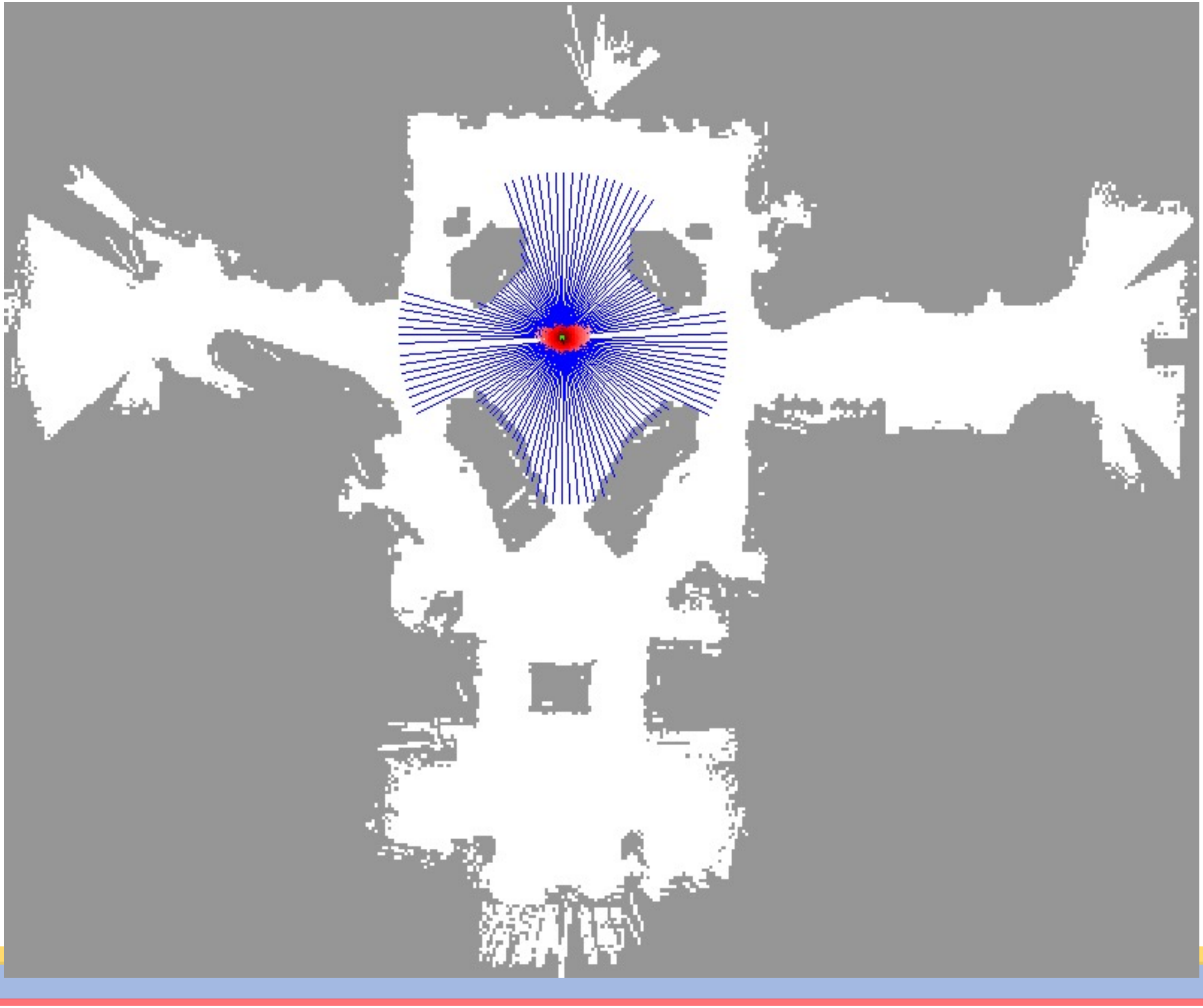




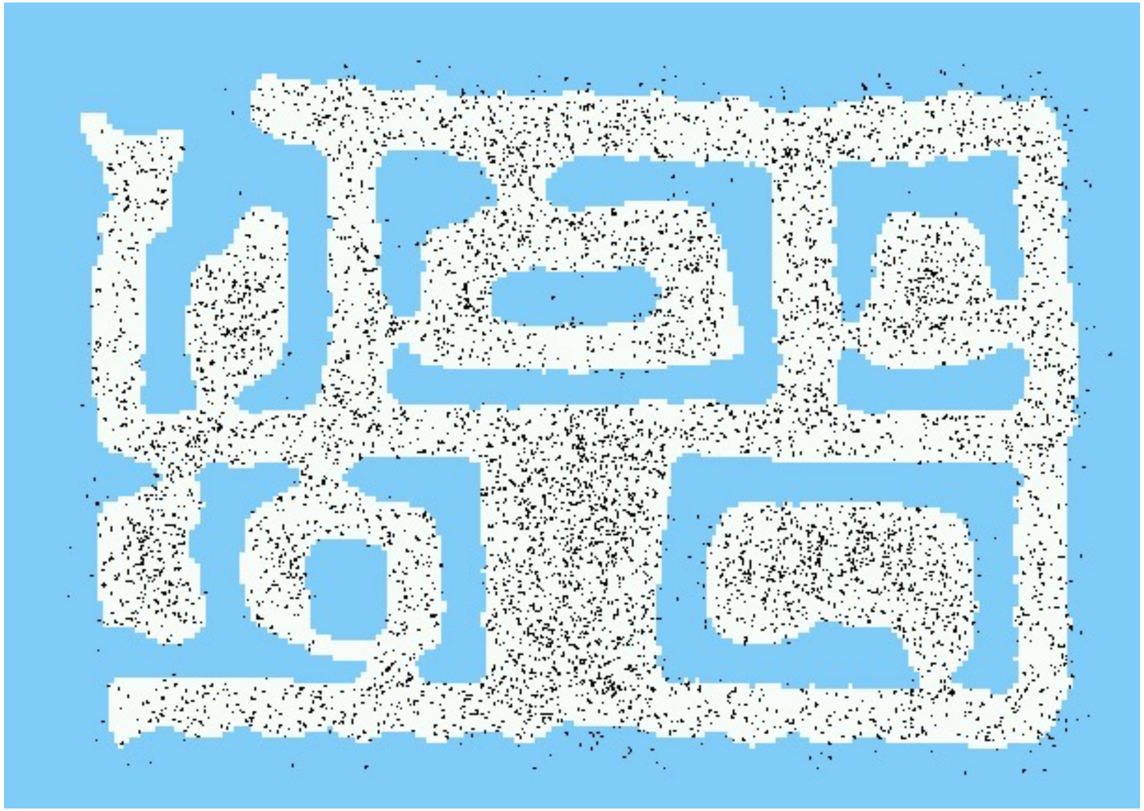




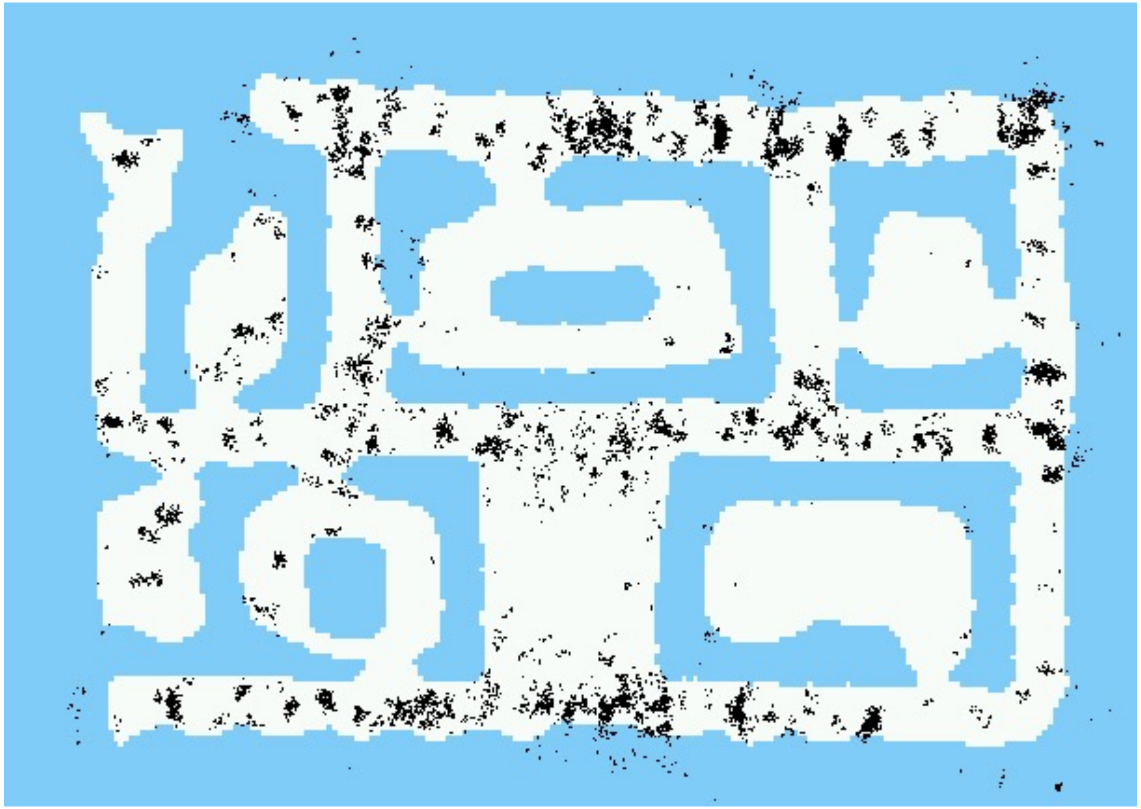




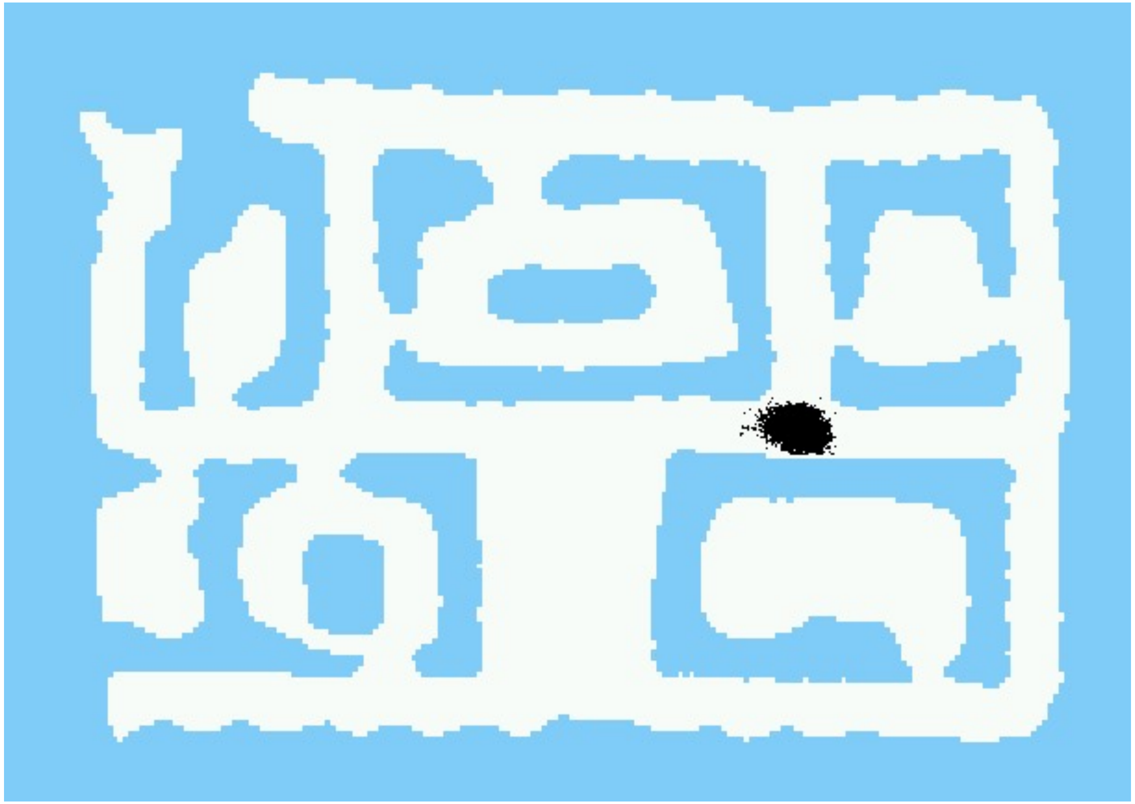
Initial Distribution



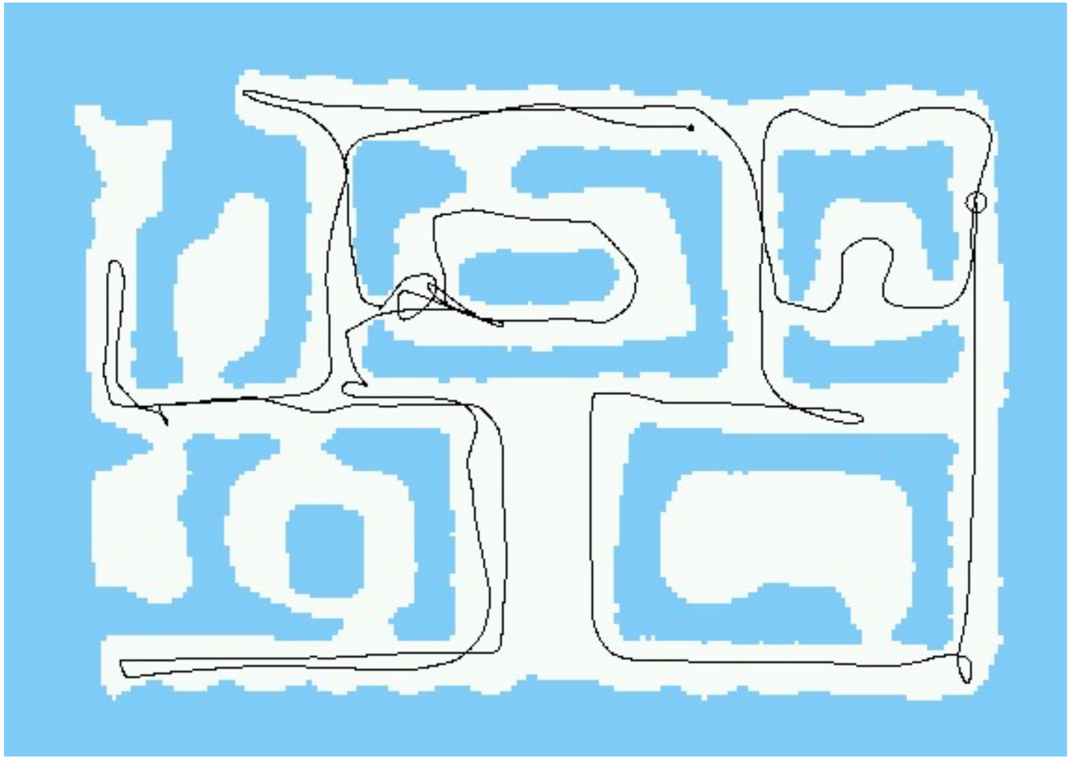
After Incorporating Ten Ultrasound Scans



After Incorporating 65 Ultrasound Scans



Estimated Path



Particle Filtering Algorithm / / Monte Carlo Localization

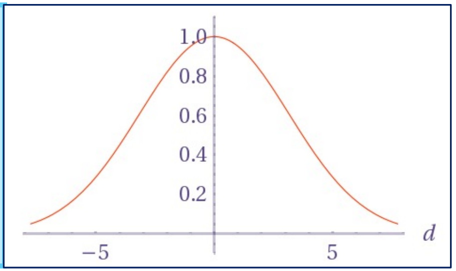
```
$X_t = x_t^{[1]}, x_t^{[2]}, \dots x_t^{[M]}$ particles

Algorithm MCL($X_{t-1}, u_t, z_t$,m):
$\bar{X}_{t-1} = X_t = \emptyset$
for all $m$ in [M] do:
    $x_t^{[m]} = \boldsymbol{sample\_motion\_model}(u_t\ x_{t-1}^{[m]})$
    $w_t^{[m]} = \boldsymbol{measurement\_model}(z_t, x_t^{[m],m})$
    $\bar{X}_t = \bar{X}_t + \langle\, x_t^{[m]}, w_t^{[m]} \rangle$
end for
for all $m$ in [M] do:
    draw $i$ $with\ probability\ \propto w_t^{[i]}$
    add $x_t^{[i]}$ $to\ X_t$
end for
return $X_t$
```

Step 1: Initialize particles uniformly distribute over space and assign initial weight

Step 2: Sample the motion model to propagate particles

Step 3: Read measurement model and assign (unnormalized) weight:

$$w_t^{[m]} = \exp\left(\frac{-d^2}{2\sigma}\right)$$



Step 4: Calculate your position update estimate by adding the particle positions scaled by weight
Note that weights must be normalized to sum to 1

Step 5: Choose which particles to *resample* in the next iteration by replacing less likely particles with more likely ones

Today's Plan

- Wrap up Bayes Filters (discrete)
- Particle Filters (non-parametric)
- Kalman Filters (continuous, linear)

Bayes Filter Reminder

$$\eta p(z_t|x_t)\int p(x_t|\, u_t, x_{t-1})bel(x_{t-1})dx_{t-1}$$

- Prediction

$$\overline{bel}(x_t) = \int p(x_t|\, u_t, x_{t-1})bel(x_{t-1})dx_{t-1}$$

- Correction

$$bel(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t)$$

What if we have a good model of our (continuous) system dynamics and we assume a Gaussian model for our uncertainty?

→ Kalman Filters!

What is a Kalman Filter?

Suppose we have a system that is governed by a linear difference equation:

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$

with measurement

$$z_t = C_t x_t + \delta_t$$

What is a Kalman Filter?

Suppose we have a system that is governed by a linear difference equation:

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$

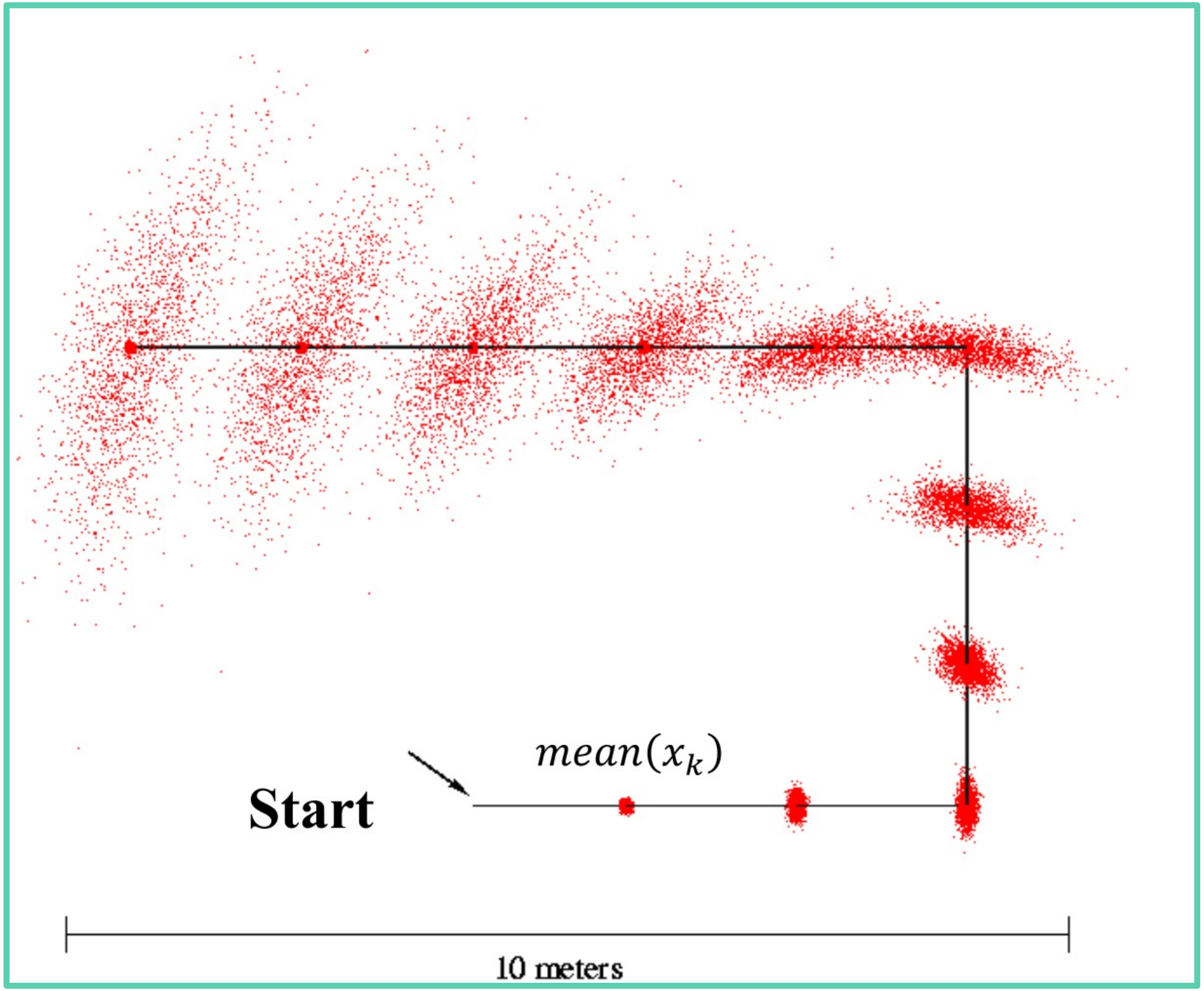
with measurement

$$z_t = C_t x_t + \delta_t$$

- Tracks the estimated state of the system by the mean and variance of its state variables -- minimum mean-square error estimator
- Computes the ***Kalman gain,*** which is used to weight the impact of new measurements on the system state estimate against the predicted value from the process model
- Note that we no longer have discrete states or measurements!

Linear Gaussian Systems: Dynamics

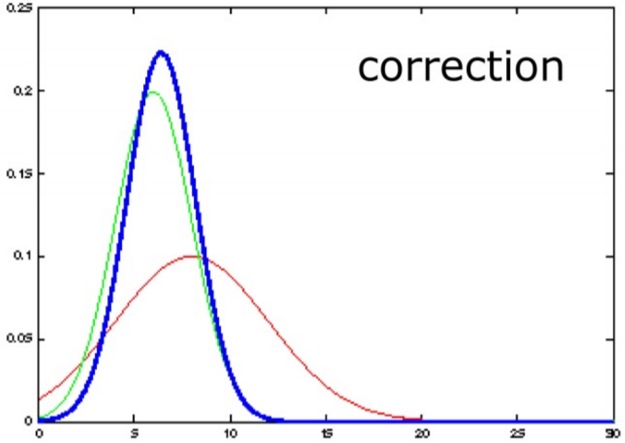
Linear Gaussian Systems: Prediction


$mean(x_k)$
Start
10 meters

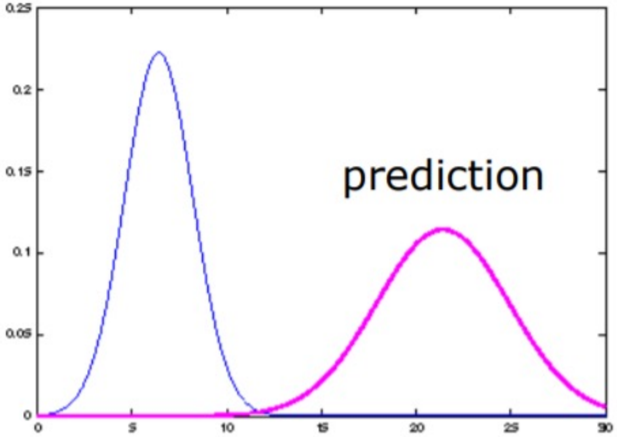
Linear Gaussian Systems: Observations

Kalman Filter Algorithm

1. Algorithm Kalman_Filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2. Prediction
   1. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
   2. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + Q_t$
3. Correction:
   1. $K_t = \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + R_t)^{-1}$
   2. $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$
   3. $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$
4. Return μ_t,Σ_t

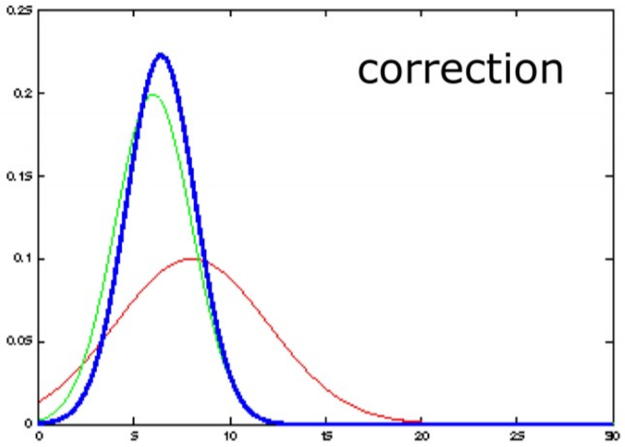


Apply control action

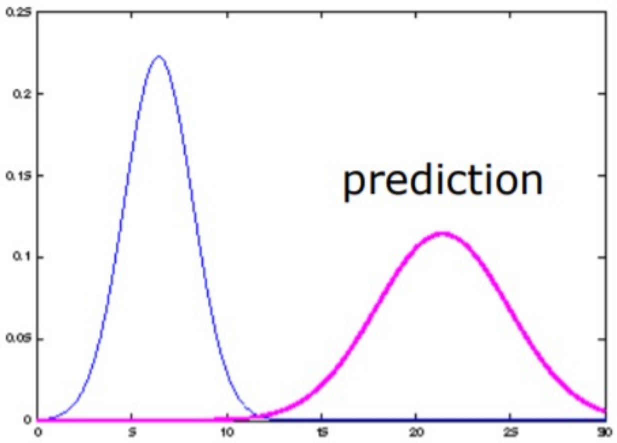


Prediction:

1. $\bar{\mu}_t = A_t\mu_{t-1} + B_t u_t$
2. $\bar{\Sigma}_t = A_t\Sigma_{t-1}A_t^{\top} + Q_t$



Apply control action

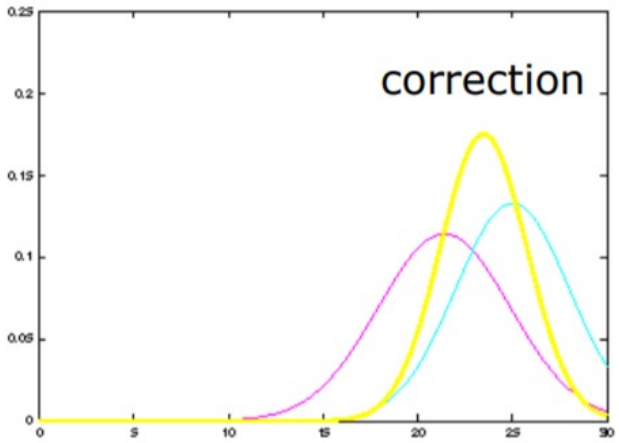


Correction:

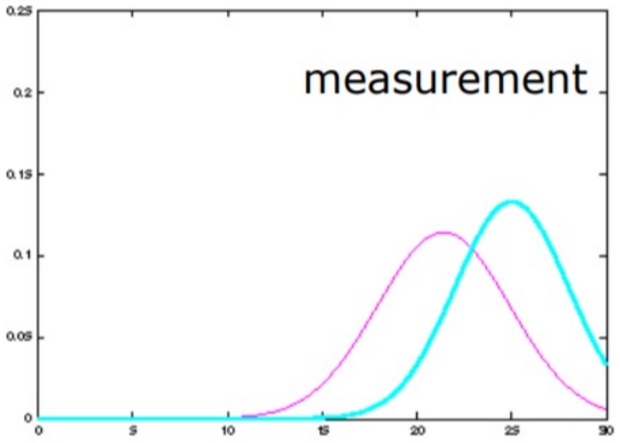
1. $K_t = \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + R_t)^{-1}$
2. $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$
3. $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$

Prediction:

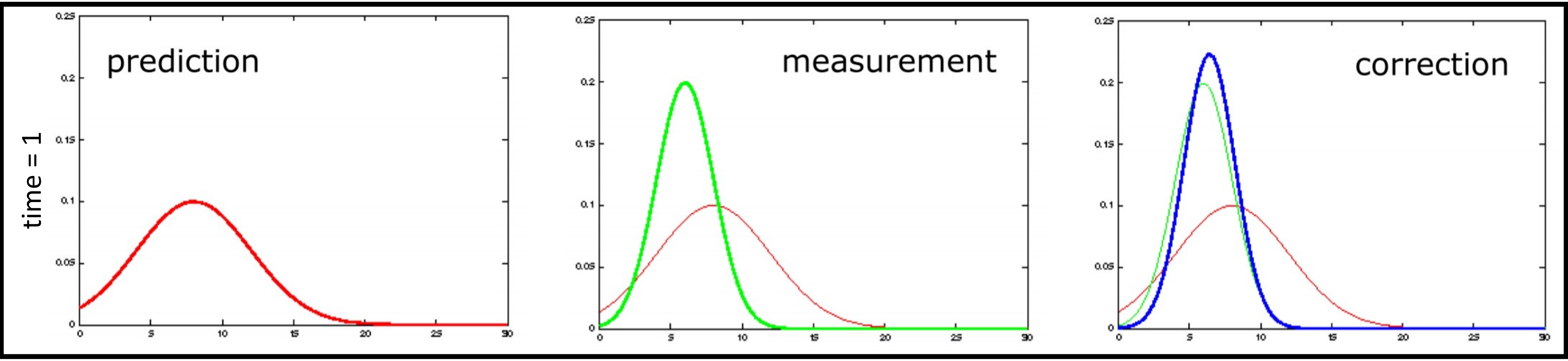
1. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
2. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + Q_t$

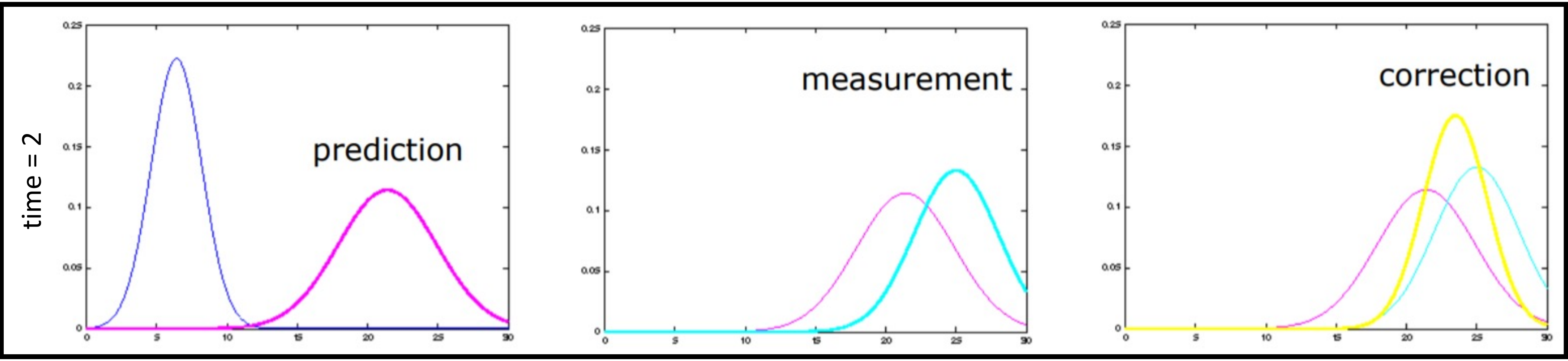


Get sensor measurement



Kalman Filter Example





Summary

- Kalman filters give you the optimal estimate for linear Gaussian systems
  - Although most systems aren’t linear ☹, this filter (and extensions) is highly efficient and widely used in practice
  - For a nice 2D example, check out: How a Kalman filter works, in pictures by Tim Babb
- Particle filters are an implementation of recursive Bayesian filtering, where the posterior is represented by a set of weighted samples
  - The particles are propagated according to the motion model and are then weighted according to the likelihood of the observations
  - In a re-sampling step, new particles are drawn with a probability proportional to the likelihood of the observation.
  - Limitations:
    - Some errors are particularly hard to deal with (e,g., the kidnapped robot problem)
    - The success of your filter is highly reliant on the number of particles → PF can be very memory and compute intensive

Extra Slides

Who was Rudolf Kalman?

Image Credit: Wikipedia

- Kálmán was one of the most influential people on control theory today and is most known for his co-invention of the Kalman filter (or Kalman-Bucy Filter)
- The filtering approach was initially met with vast skepticism, so much so that he was forced to do the first publication of his results in mechanical engineering, rather than in electrical engineering or systems engineering
  - This worked out find as some of the first use cases was with NASA on the Apollo spacecraft
- *Kalman filters are inside every robot, commercial airplanes, uses in seismic data processing, nuclear power plant instrumentation, and demographic models, as well as applications in econometrics*