COLUMBIA UNIVERSITY EEME E6911 FALL ‘25

TOPICS IN CONTROL : PROBABILISTIC ROBOTICS

MONTE CARLO LOCALIZATION (MCL)

Instructor: Ilija Hadzic

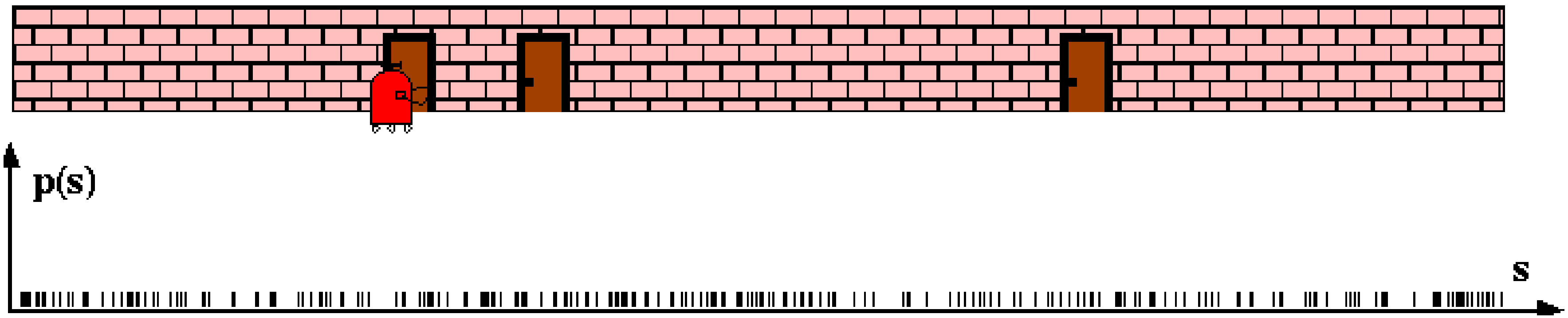
What is MCL?

- Localization using Particle Filter
- Motion Model:
  - Moves particles according to robot kinematics
  - Adds randomness according to the error model
- Measurement Model:
  - Scores particles using landmarks or raw measurement
  - Resamples according to importance
- Performance Functions:
  - Decide how many particles to use
  - Decide when to reseed the particle

Why MCL?

- Advantages over EKF:
  - Handles any error model
  - Works for multi-modal distributions
  - Can solve kidnap
  - Can do global localization
- Disadvantages:
  - Computationally complex (needs many particles)
  - Missing the deadline vs. degrading the performance
  - A few surprising effects (to be discussed)

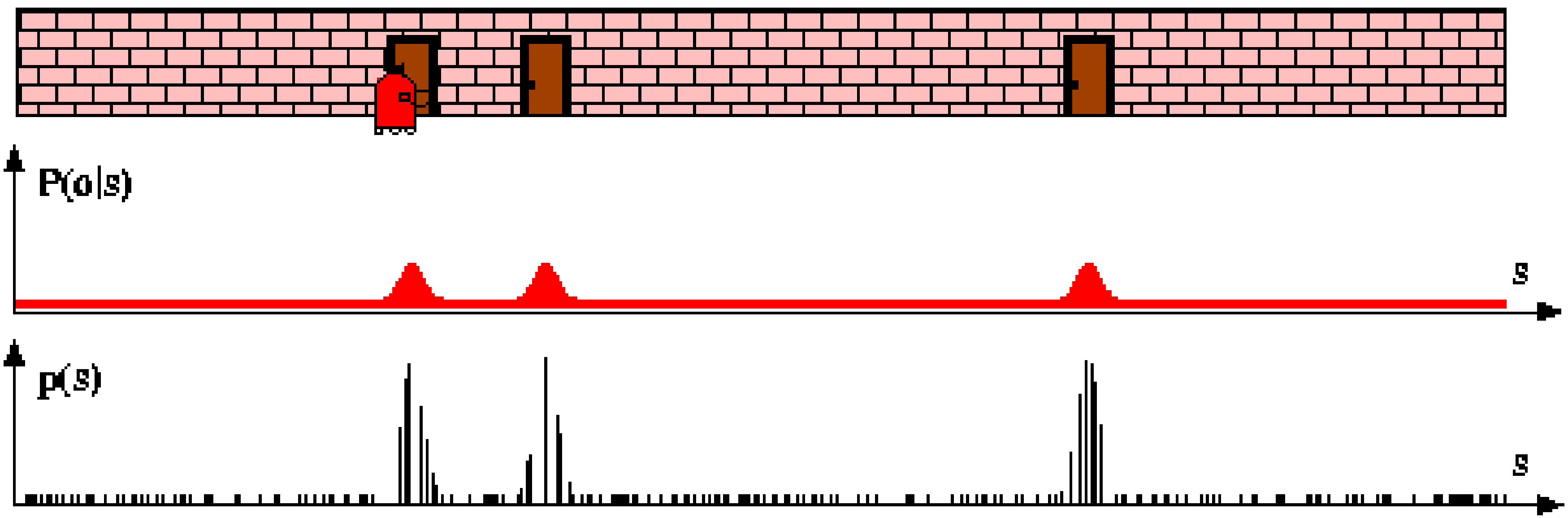
1D Example From the Textbook



- Particles scattered at random (uniform distribution)
- Robot has no idea where it is

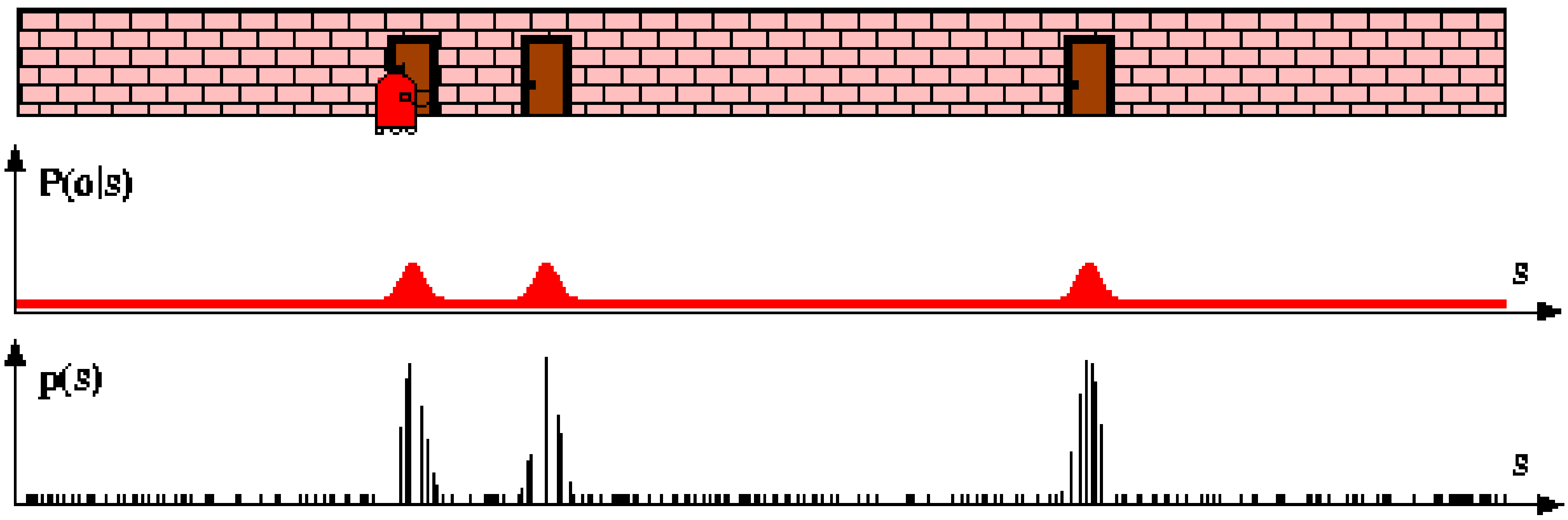
Source: `www.probabilistic-robotics.org`

1D Example From the Textbook



- Particles scored after observing the door
- Some particles are more important than others
- Unless the robot moves, it cannot tell anything more

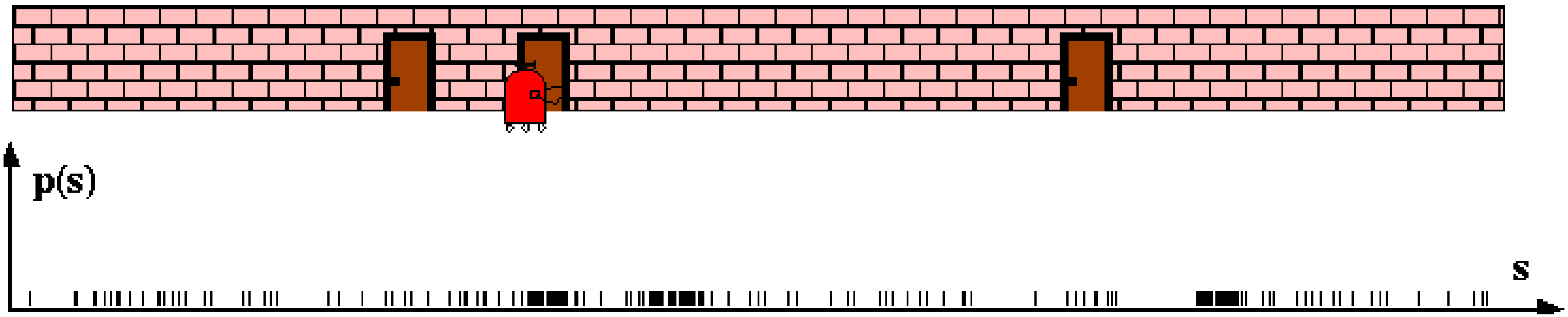
1D Example From the Textbook



- What would the scoring look like if robot were to observe the wall instead of the door?

Source: `www.probabilistic-robotics.org`

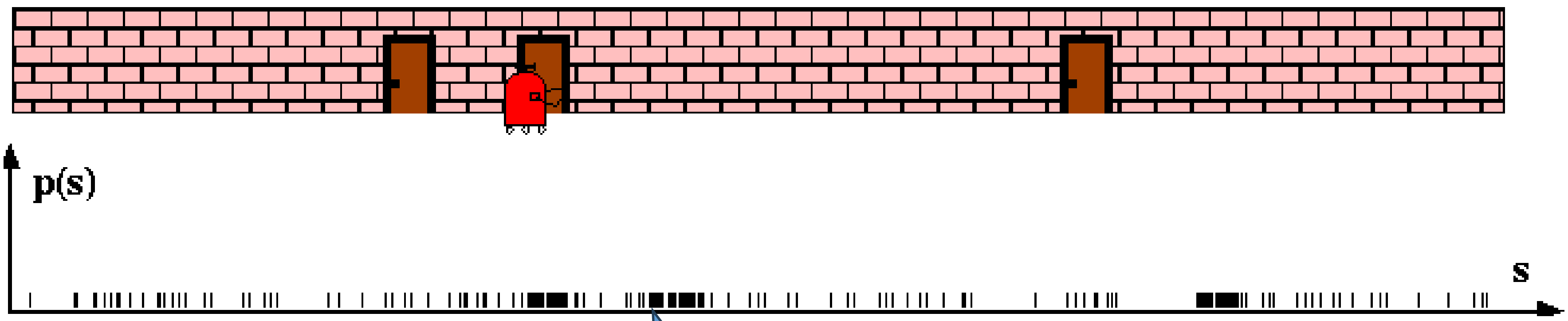
1D Example From the Textbook



- Particles resampled
- Robot moved; particles scored again
- Resampled
- Moved again, encountered the second door

Source: `www.probabilistic-robotics.org`

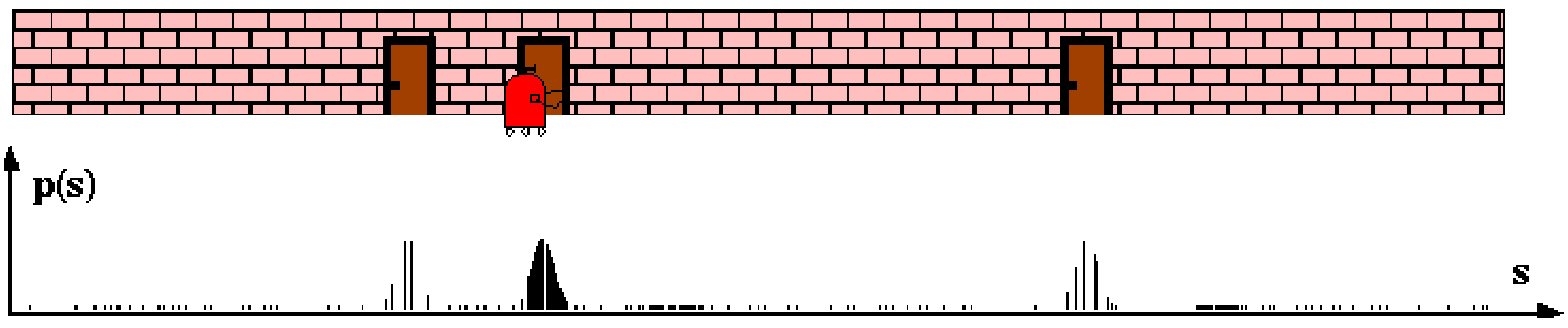
1D Example From the Textbook



Why are particles denser here?

Source: `www.probabilistic-robotics.org`

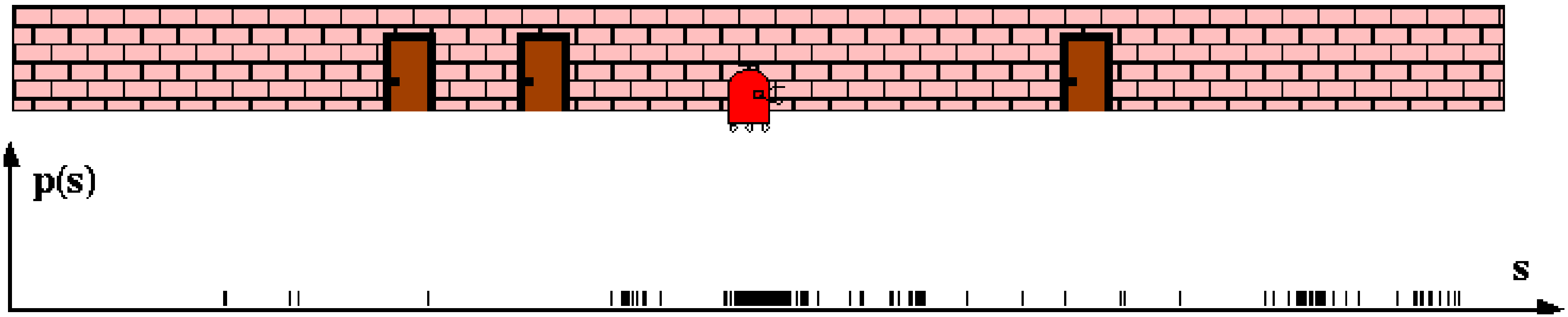
1D Example From the Textbook



- Particles scored
- Resampled
- Robot moves again

Source: `www.probabilistic-robotics.org`

1D Example From the Textbook



- Distribution is now (almost) unimodal
- All particles bunched around robot's true position
- Other particles died off

Source: `www.probabilistic-robotics.org`

2D Motion Model Example

1: **Algorithm sample_motion_model_velocity**(u_t, x_{t-1})**:**

2: $\hat{v} = v + \mathbf{sample}(\alpha_1|v| + \alpha_2|\omega|)$

3: $\hat{\omega} = \omega + \mathbf{sample}(\alpha_3|v| + \alpha_4|\omega|)$

4: $\hat{\gamma} = \mathbf{sample}(\alpha_5|v| + \alpha_6|\omega|)$

5: $x' = x - \frac{\hat{v}}{\hat{\omega}} \sin\theta + \frac{\hat{v}}{\hat{\omega}}\ sin(\theta + \hat{\omega}\Delta t)$

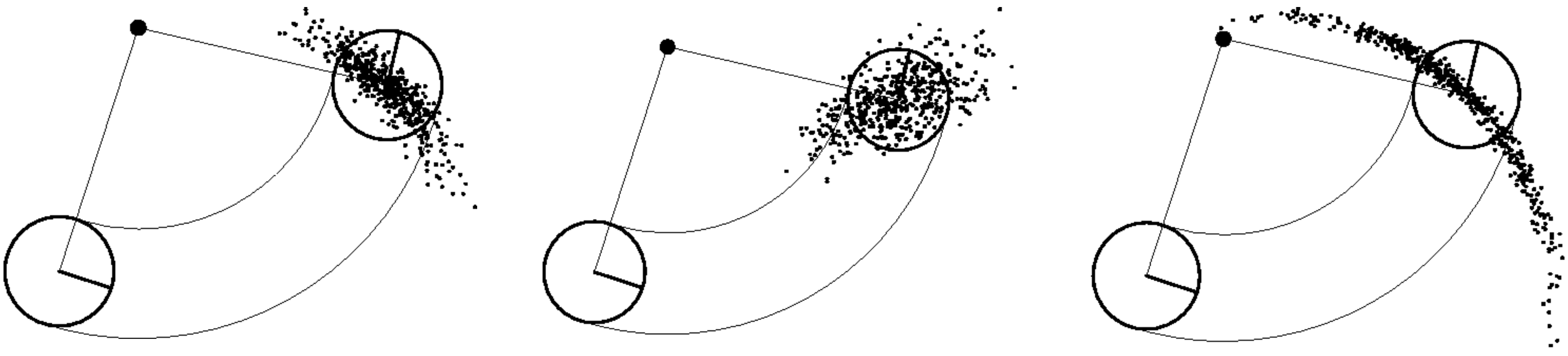
6: $y' = y + \frac{\hat{v}}{\hat{\omega}} \cos\theta - \frac{\hat{v}}{\hat{\omega}}\ cos(\theta + \hat{\omega}\Delta t)$

7: $\theta' = \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t$

8: ***return*** $x_t = (x', y', \theta')^T$

Source: `www.probabilistic-robotics.org`

2D Motion Model Example

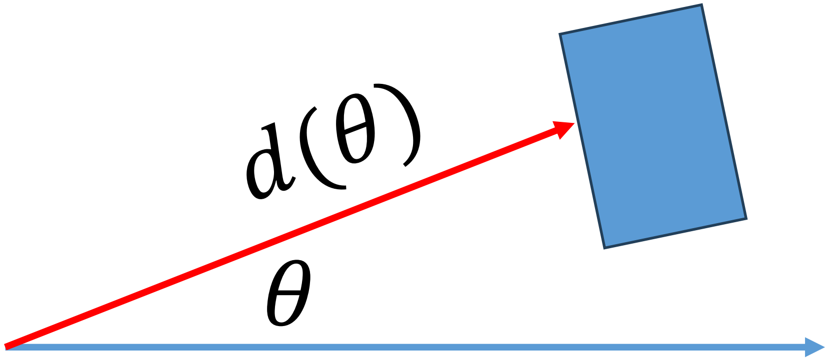


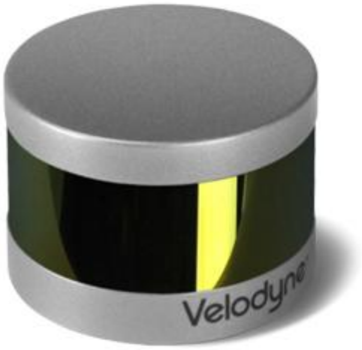
- Thrun's model we saw before, but this time using particles
- Only looks at velocity command

Source: `www.probabilistic-robotics.org`

Range Finder Measurement Model

- Senses the distance to a fixed object (e.g. walls):
  - LIDAR, Radar, Sonar
- Output is the distance in polar coordinates

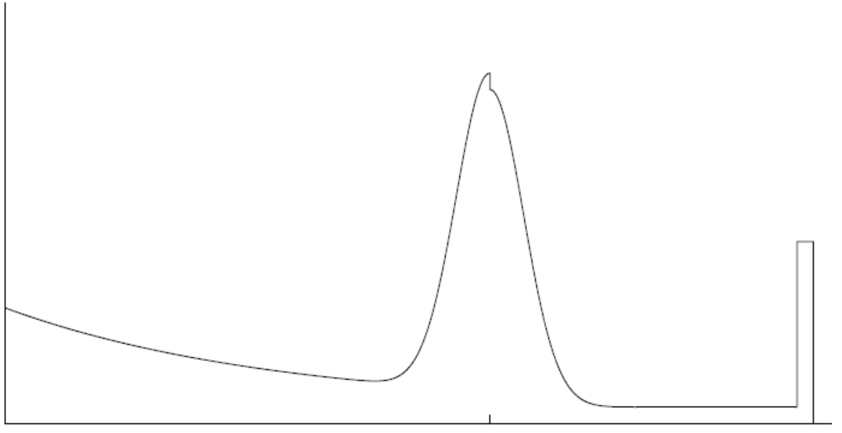




Beam Model

- What are we modeling?
  - $p(z_i[n] \mid \bar{x}[n], m)$ for each beam i.

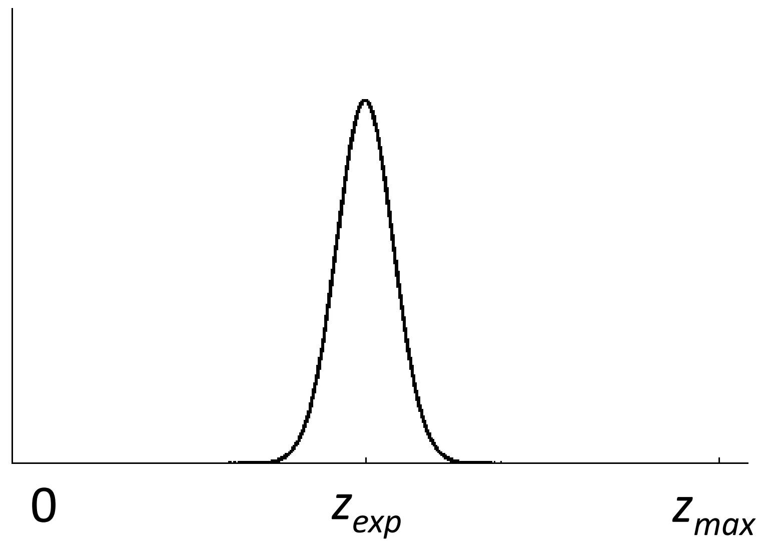
$$P(z \mid x, m) = \begin{pmatrix} \alpha_{\text{hit}} \\ \alpha_{\text{unexp}} \\ \alpha_{\text{max}} \\ \alpha_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} P_{\text{hit}}(z \mid x, m) \\ P_{\text{unexp}}(z \mid x, m) \\ P_{\text{max}}(z \mid x, m) \\ P_{\text{rand}}(z \mid x, m) \end{pmatrix}$$



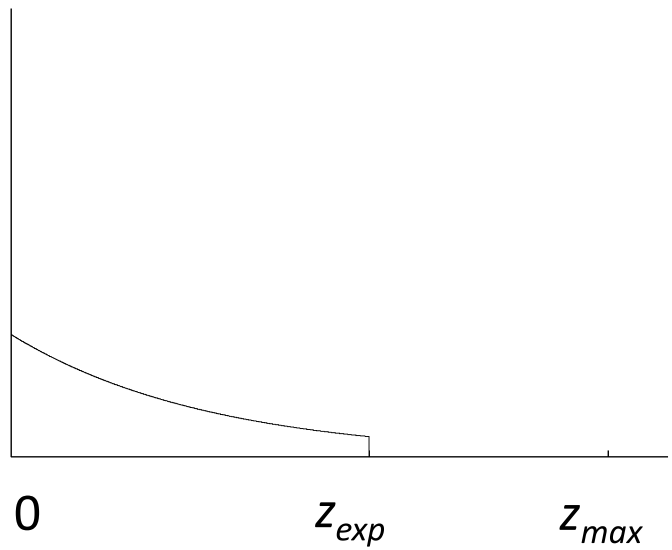
Source: `www.probabilistic-robotics.org`

Beam Model

Measurement noise



Unexpected obstacles



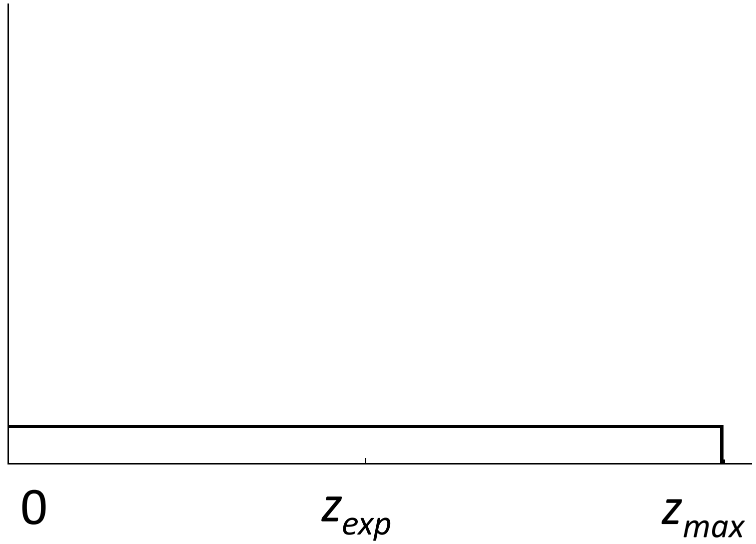
$$P_{hit}(z \mid x, m) = \eta \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2}\frac{(z - z_{\exp})^2}{b}}$$

$$P_{\text{unexp}}(z \mid x, m) = \begin{Bmatrix} \eta \; \lambda \; \mathrm{e}^{-\lambda z} & z < z_{\exp} \\ 0 & otherwise \end{Bmatrix}$$

Source: `www.probabilistic-robotics.org`

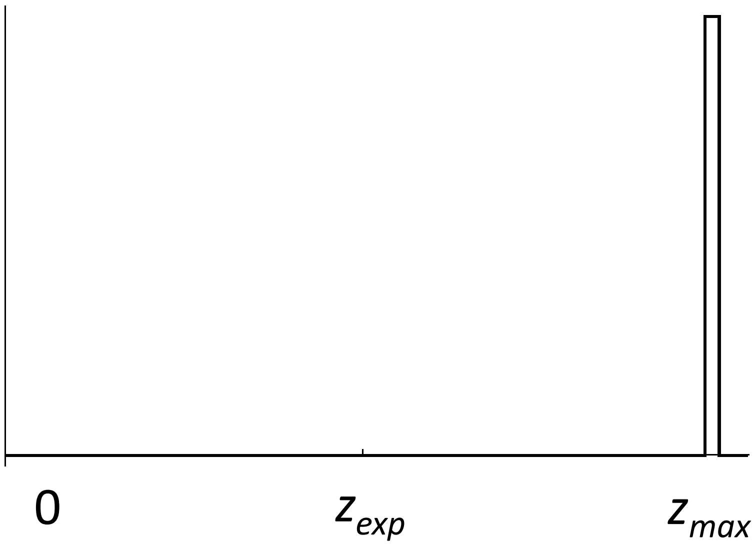
Beam Model

Random measurement



$$P_{rand}(z \mid x, m) = \eta \frac{1}{z_{\max}}$$

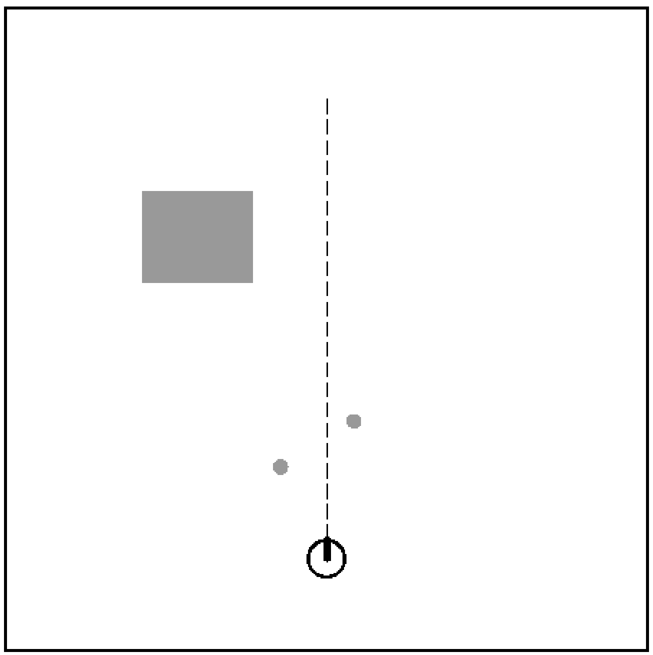
Max range



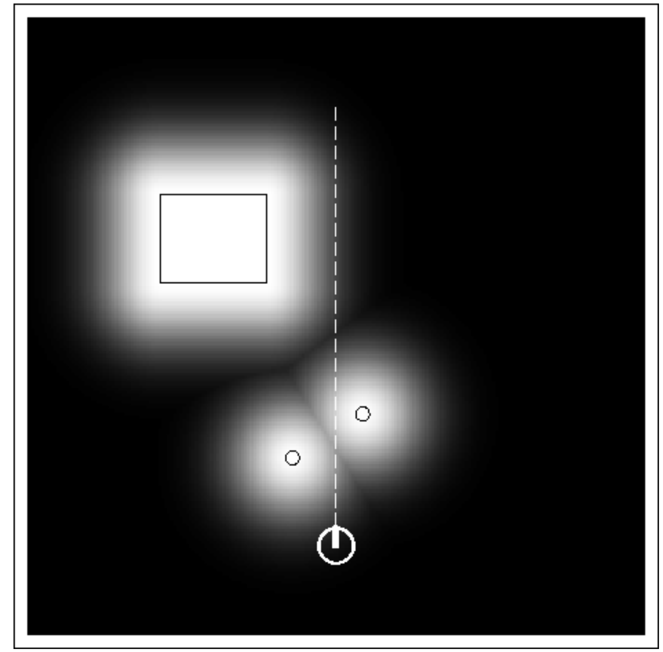
$$P_{\max}(z \mid x, m) = \eta \frac{1}{z_{small}}$$

Source: `www.probabilistic-robotics.org`

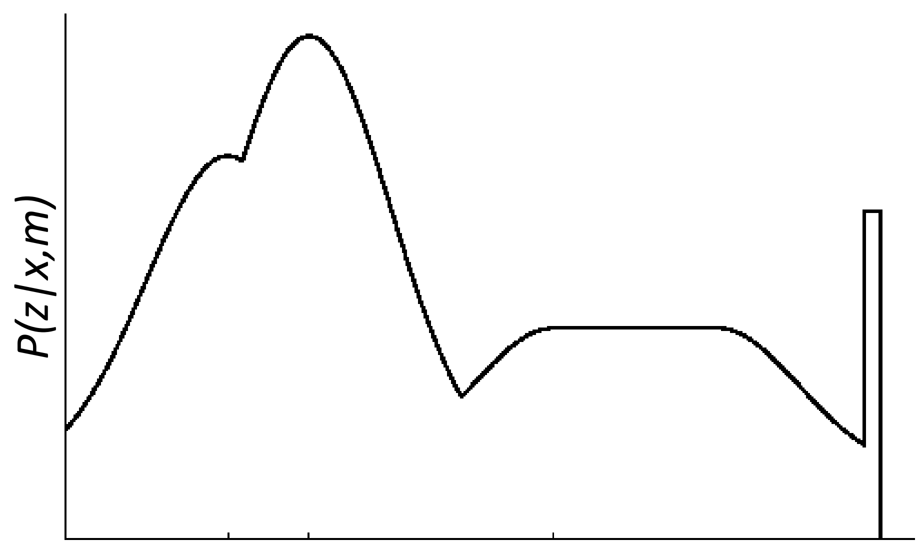
Likelihood Fields



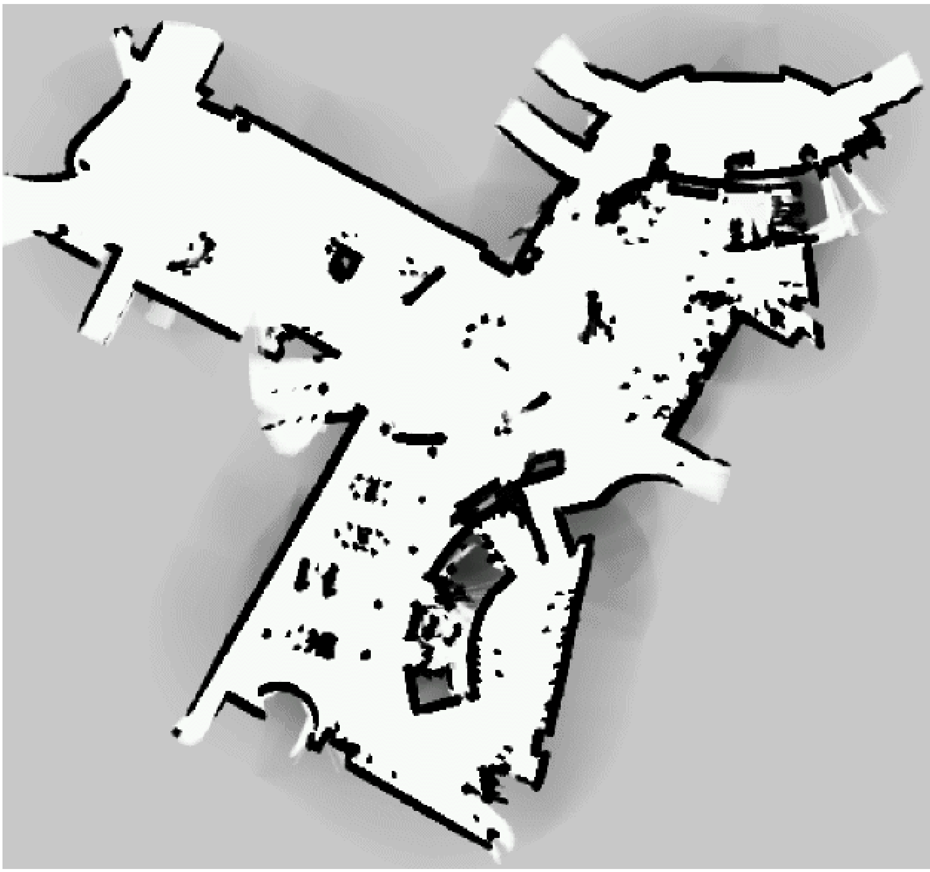
Map m



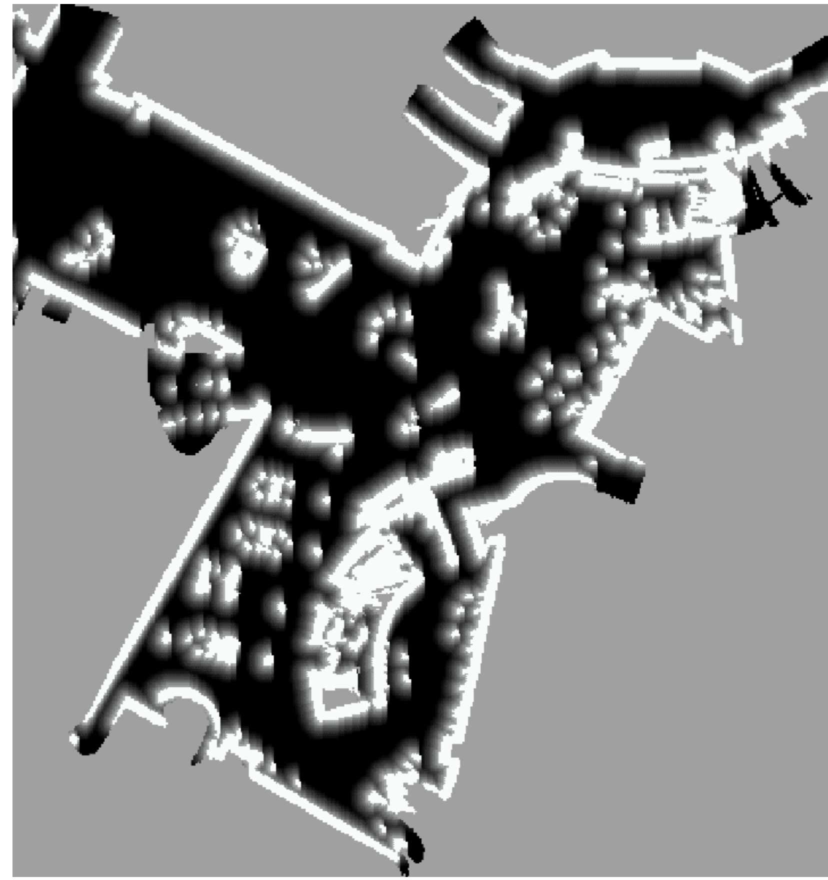
Likelihood field



San Jose Tech Museum

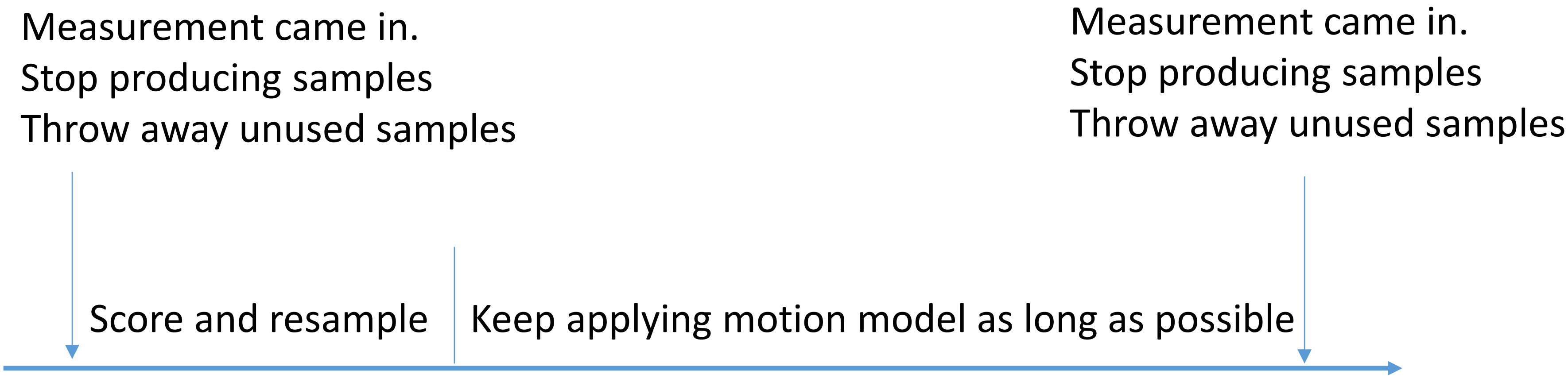


Occupancy grid map



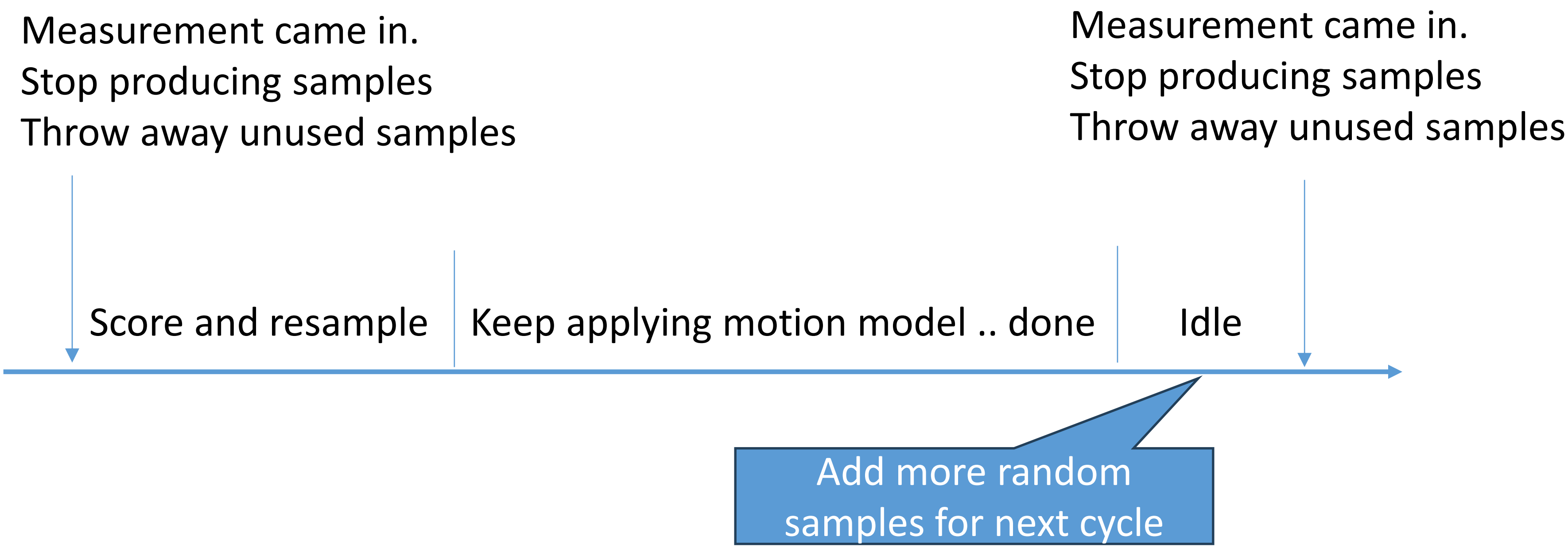
Likelihood field

Sampling Against the Deadline



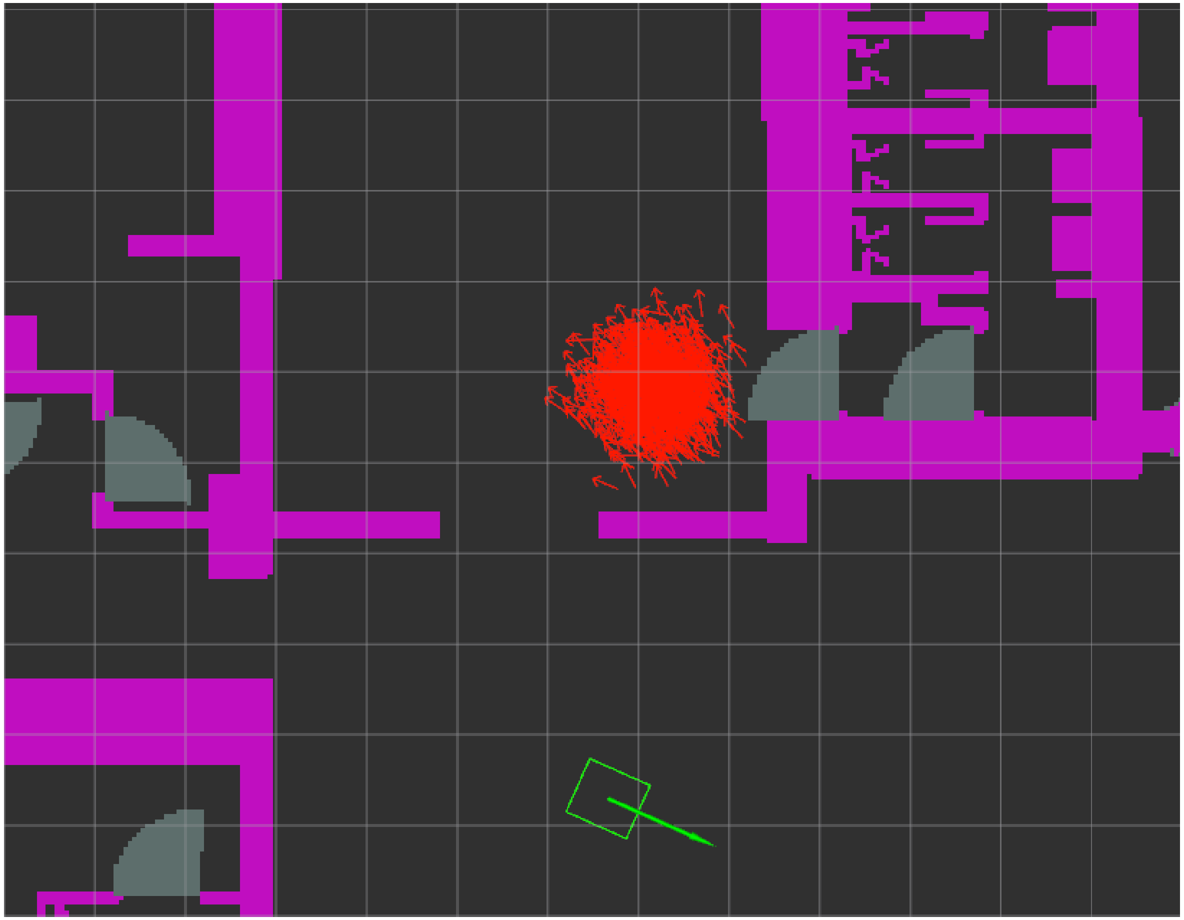
- Keep sampling until the new measurements comes in.
- Work with particles produced thus far
- Faster CPU achieves better precision.

Sampling Against the Deadline

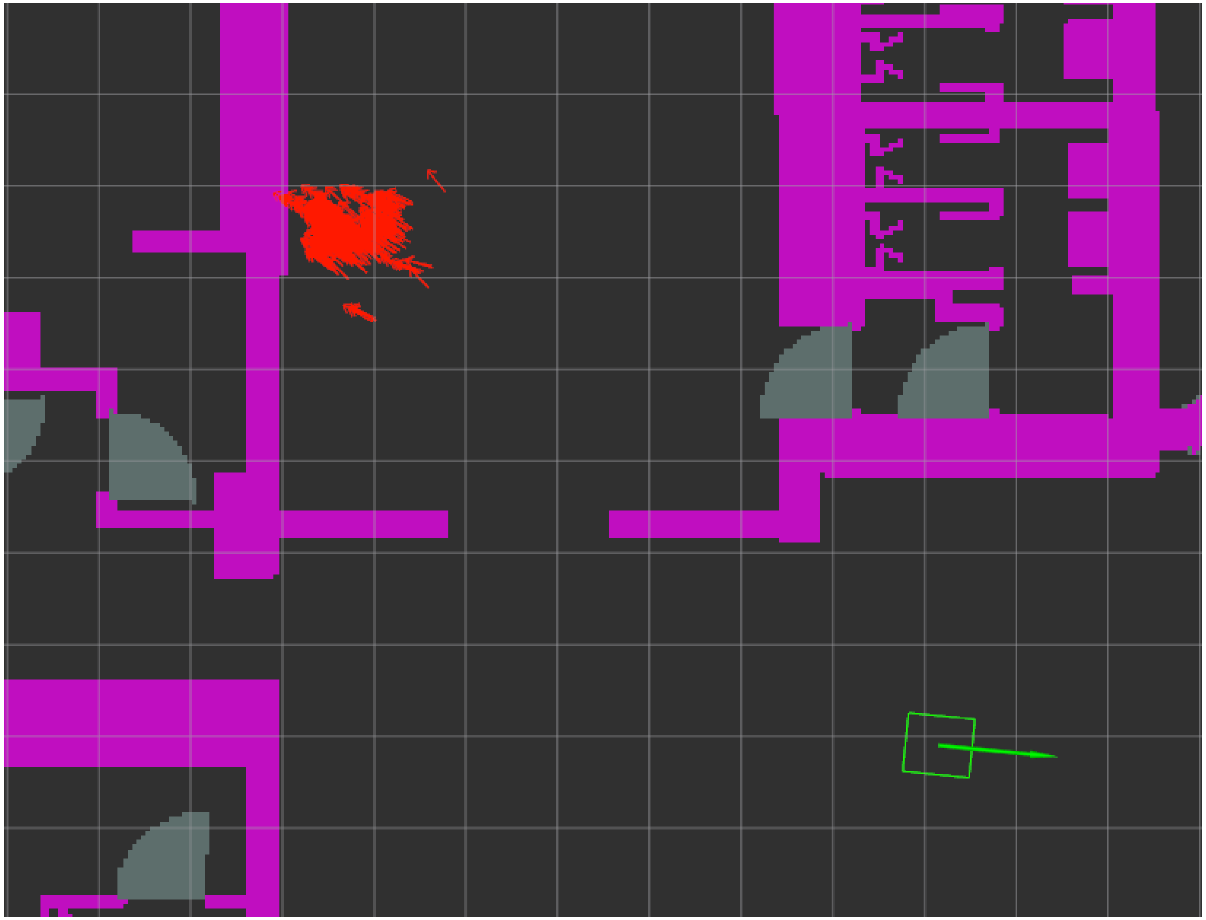


- If we have time budget left, add more samples.
- The number of samples converges to fill the available CPU time.
- Algorithm adapts to available compute resources.

Adding Random Particles



Adding Random Particles



Adding Random Particles

- Problem:
  - Estimate remains stuck to the initial cloud
  - No way to get out
- Solution:
  - Add random particles
  - How many?
  - Where?

Adding Random Particles

$$w^{[j]}[n] = p(z[n] | \bar{x}^{[m]}[n])$$

$$w_{\text{avg}}[n] = \frac{1}{N}\sum_{j=1}^{N} w^{[j]}[n]$$

$$w_{\text{slow}} = w_{\text{slow}} + \alpha_{\text{slow}}(w_{\text{avg}} - w_{\text{slow}})$$

$$w_{\text{fa}st} = w_{\text{fast}} + \alpha_{\text{fast}}(w_{\text{avg}} - w_{\text{fast}})$$

$$0 < \alpha_{\text{slow}} \ll \alpha_{\text{fast}}$$

Adding Random Particles

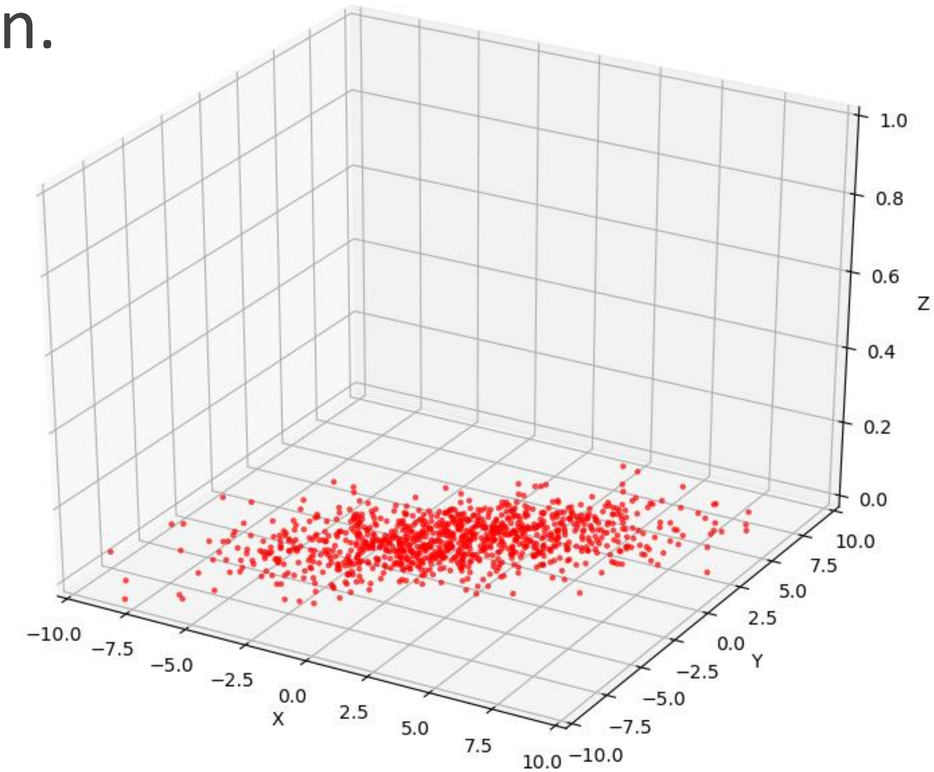
- Probability of replacing a traced particle with the random particle

$$\max\left\{0, 1 - \frac{w_{\text{fast}}}{w_{\text{slow}}}\right\}$$

- Picking the area to cover with random particles is usually based on heuristics.
- Landmark identification can help:
  - Example: Spotting the Statue of Liberty narrows down the area to a few streets on Manhattan or in New Jersey

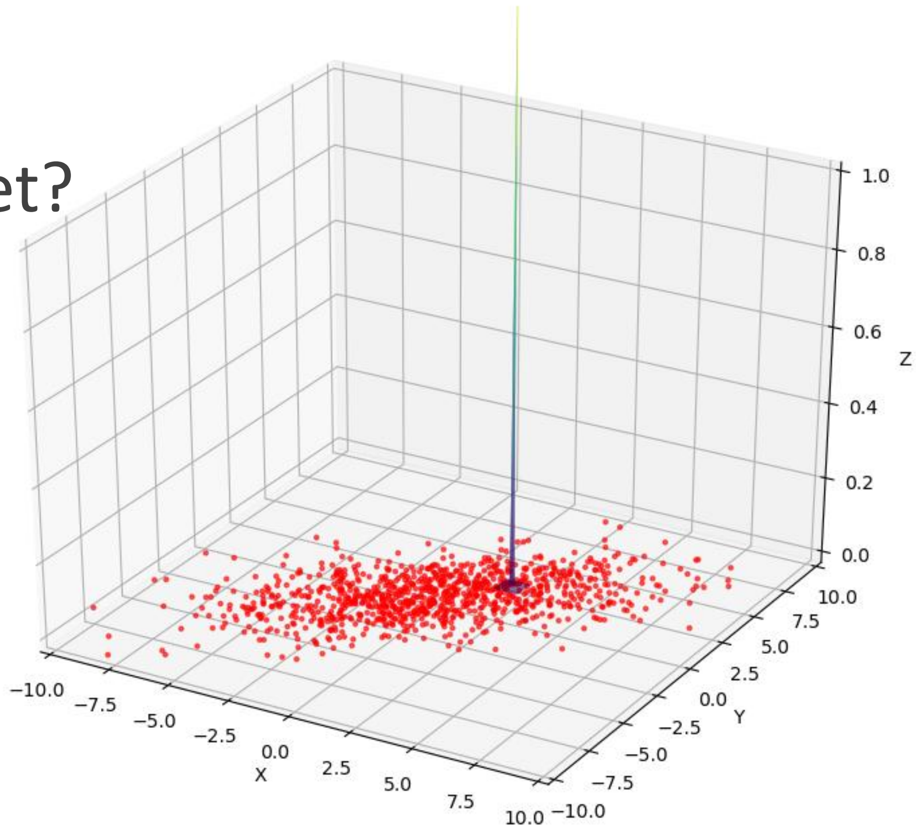
Perfect Sensor Paradox

- Consider a particle cloud.
- Measurement comes in.
- Score them!



Perfect Sensor Paradox

- Perfect measurement!
- What is the variance?
- What weights do we get?
- Why?



Remedies

- Add artificial noise to the sensor.
- Explicitly recognize the case and defend from it:
  - Take the measurement at the face value
  - Initialize the particle cloud
- Mid-ground solution:
  - For a subset of particles, use measurement distribution
  - Do the regular processing of the rest
  - Problem: This can be computationally expensive
- Exercise:
  - Take the code from previous class
  - Keep lowering the measurement variance until it crashes
  - Explain the crash and fix it!

KLD Sampling

- Kullback-Leibler Divergence:
  - Measure of how different two distributions are.
- In theory:
  - Measures sampled and true distributions
  - If the difference is high, add more particles.
- In practice:
  - Watch where particles are landing.
  - If they are landing at empty spots, keep adding.
  - If they are landing at similar spots, stop.

KLD Sampling

- Parameters: δ, ϵ
- Set some minimum number of samples $M_{\min}$
- Partition the space into bins
- Count the number of bins with at least one sample
- Calculate this:

$$M_x = \frac{k-1}{2\epsilon}\left(1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}\, z_{1-\delta}\right)^3$$

- We need $\max(M_x, M_{\min})$ samples.

ROS: AMCL Node

Parameter	Min	Max	Value
min_particles	0	1000	500
max_particles	0	10000(	5000
kld_err	0.0	1.0	0.05
kld_z	0.0	1.0	0.99
update_min_d	0.0	5.0	0.2
update_min_a	0.0	6.283185307	0.2
resample_interval	0	20	1
transform_tolerance	0.0	2.0	0.2
recovery_alpha_slow	0.0	0.5	0.0
recovery_alpha_fast	0.0	1.0	0.0
do_beamskip			☐
beam_skip_distance	0.0	2.0	0.5
beam_skip_threshold	0.0	1.0	0.3
tf_broadcast			☐
force_update_after_initialpose			☐
force_update_after_set_map			☐
gui_publish_rate	-1.0	100.0	10.0
save_pose_rate	-1.0	10.0	-1.0
use_map_topic			☐
first_map_only			☐

Parameter	Min	Max	Value
laser_min_range	-1.0	1000.(	-1.0
laser_max_range	-1.0	1000.(	50.0
laser_max_beams	0	250	180
laser_z_hit	0.0	1.0	0.5
laser_z_short	0.0	1.0	0.1
laser_z_max	0.0	1.0	0.05
laser_z_rand	0.0	1.0	0.5
laser_sigma_hit	0.0	10.0	0.2
laser_lambda_short	0.0	10.0	0.1
laser_likelihood_max_dist	0.0	20.0	2.0
laser_model_type			likelihood_field_const (likelihood_field)
odom_model_type			diff_corrected_const (diff-corrected)
odom_alpha1	0.0	10.0	0.02
odom_alpha2	0.0	10.0	0.01
odom_alpha3	0.0	10.0	0.01
odom_alpha4	0.0	10.0	0.02
odom_alpha5	0.0	10.0	0.02
odom_frame_id			odom
base_frame_id			base_link
global_frame_id			map
restore_defaults			☐