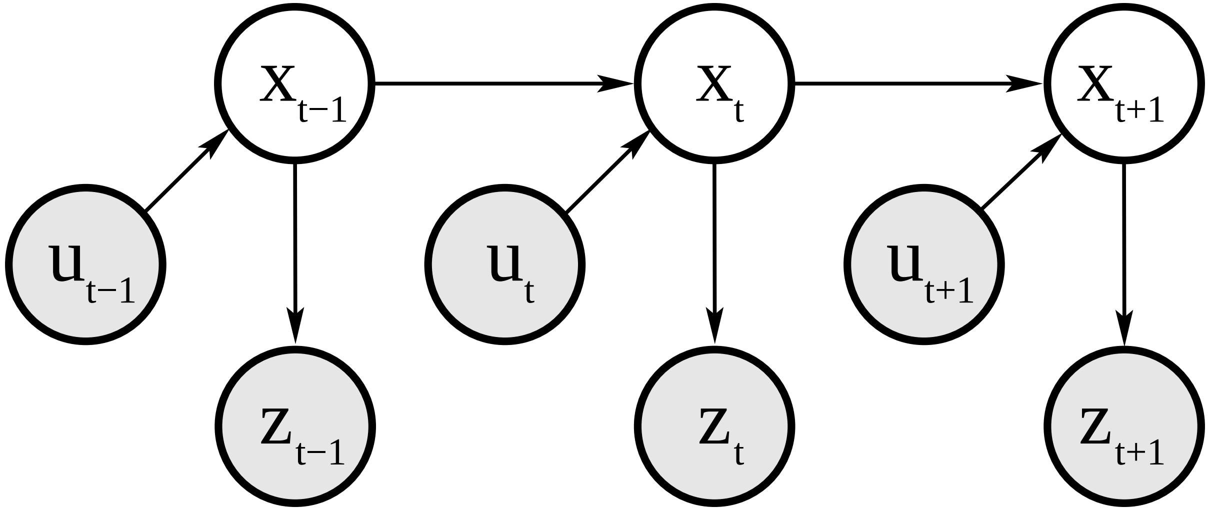
Robot Perception

Dynamic Bayes Network for Robot Pose



Dynamic Bayes Network for Robot Pose

Under the Markov assumption, we are required to model

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

which is called the **state transition probability** and shows how the robot's pose changes in time due to input actions.

And,

$$p(z_t|x_t)$$

which is called the **measurement probability** and shows the probability of sensor measurements given the robot's pose

Sensor modeling

We do not have to model the phenomena that generate the measurements

We aim to model the **typical noise** observed by the sensor.

`Note` **We may use any insights based on the specific phenomena that generate the sensor measurements, but we can treat is a black box**

Laser rangefinder models

Laser rangefinder

Sensor commonly used in robotics uses a narrow laser beam to determine the distance to objects using the time it takes the beam to be reflected from the object (`time of flight`)

2D rangefinders $(z,\ \theta)$

Uses a mirror to rotate a laser beam. For each beam, we have distance z and angle θ

Sensor	RP-1	Hokuyo UST-05	Hokuyo UTM-30	Hokuyo UXM-30
				
Range [m]	0.15~6	0.06~5	0.1~30	0.1~100
beams/scan	360	1080	1080	1080
FoV [deg]	360	270	270	270

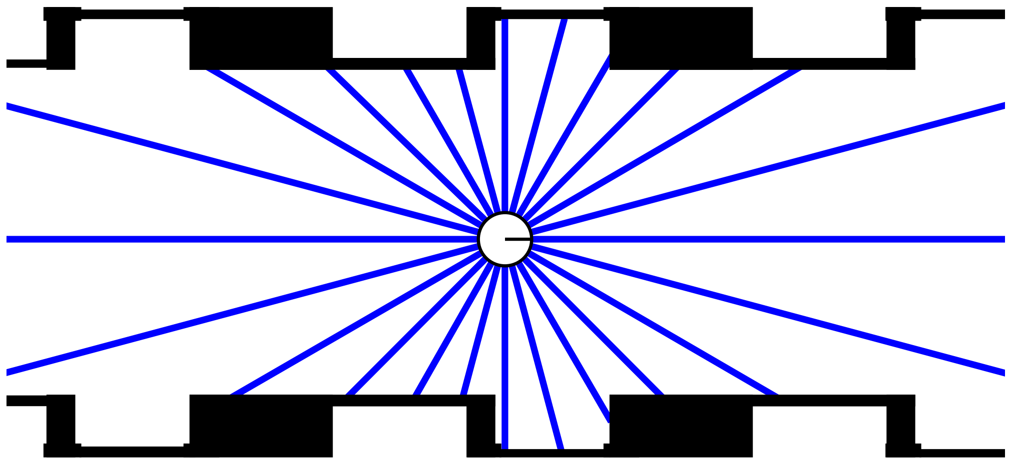
3D rangefinders (XYZ)

Sensor	Velodyne Puck	Velodyne HDL32	Robosense rs-ruby
			
Range [m]	1~100	1~100	0.4~200
beams/horizontal scan	900~3600	900~3600	900/1800
Channels	16	32	128

Beam Model

We match the distance obtained from each beam of the rangefinder to the distance that should be obtained given a geometric map of the environment.

Example of range data (ideal case)

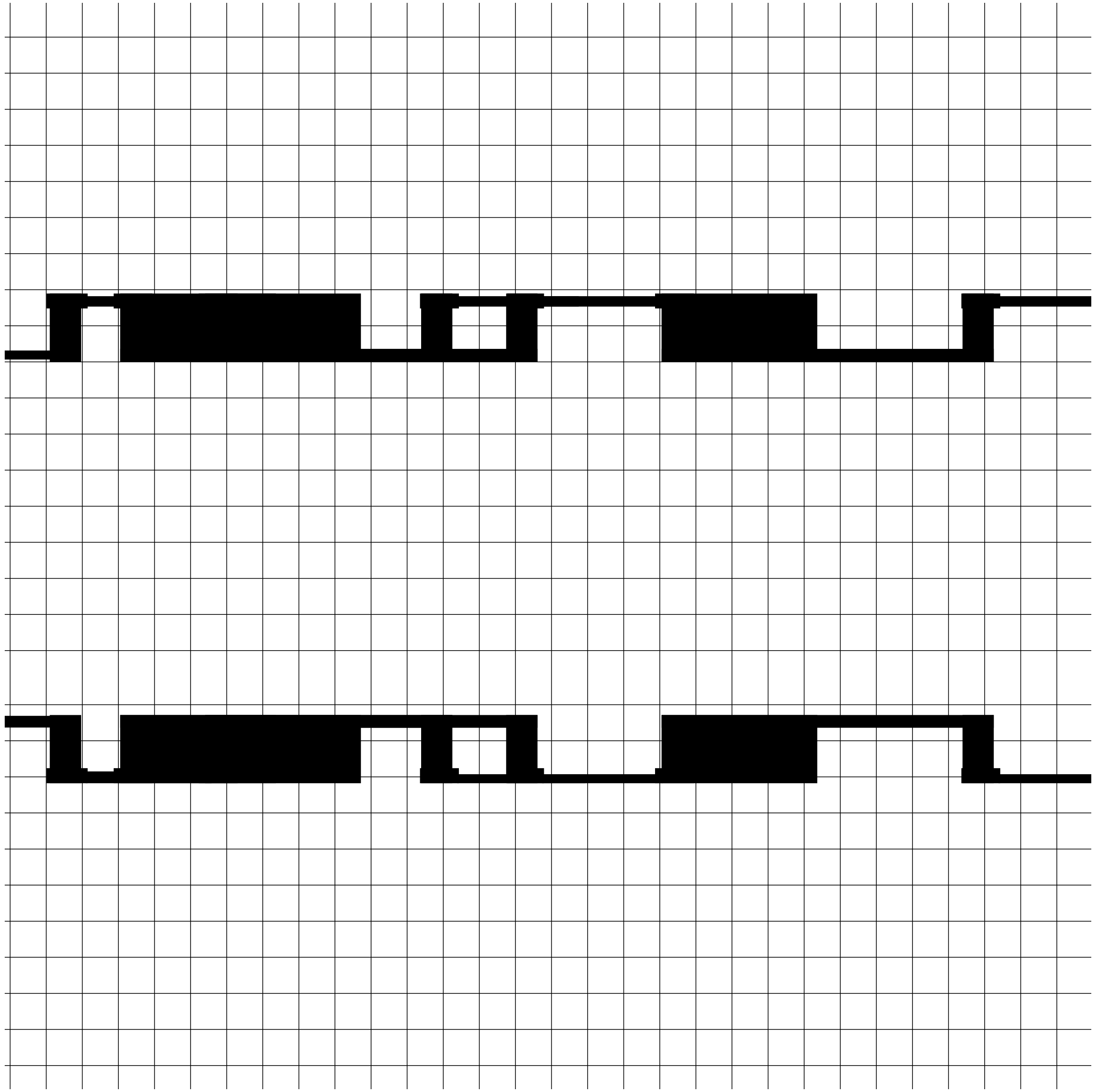


Ray Casting

To implement this model, for any possible location $\mathbf{x}$ we need to find the distance of beam i to its closest object in the map z_d.

Considering a grid map, where each cell is either

1. empty/free
2. occupied

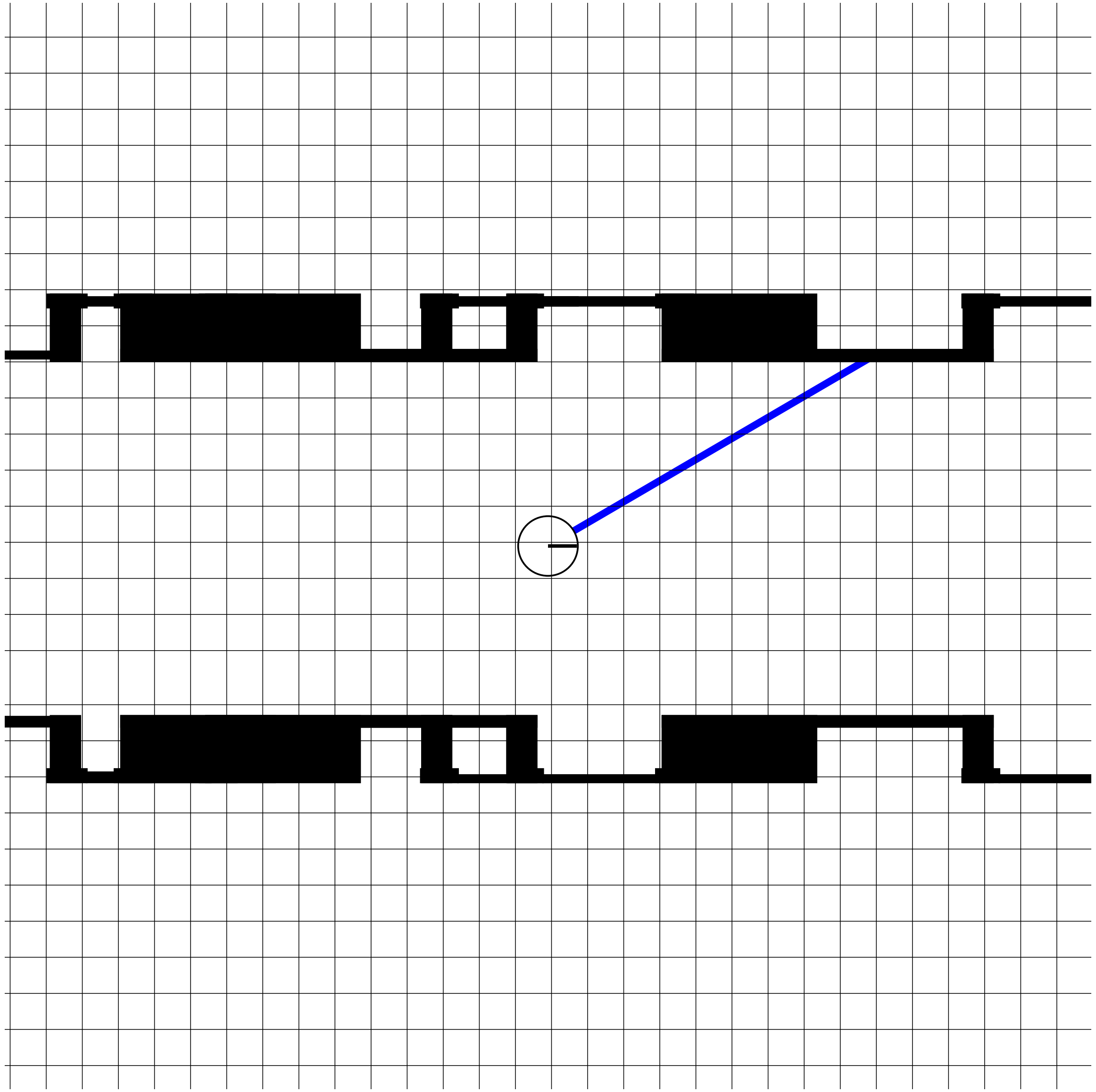


Ray Casting

We check for the closest obstacle by checking every cell on the path of the beam

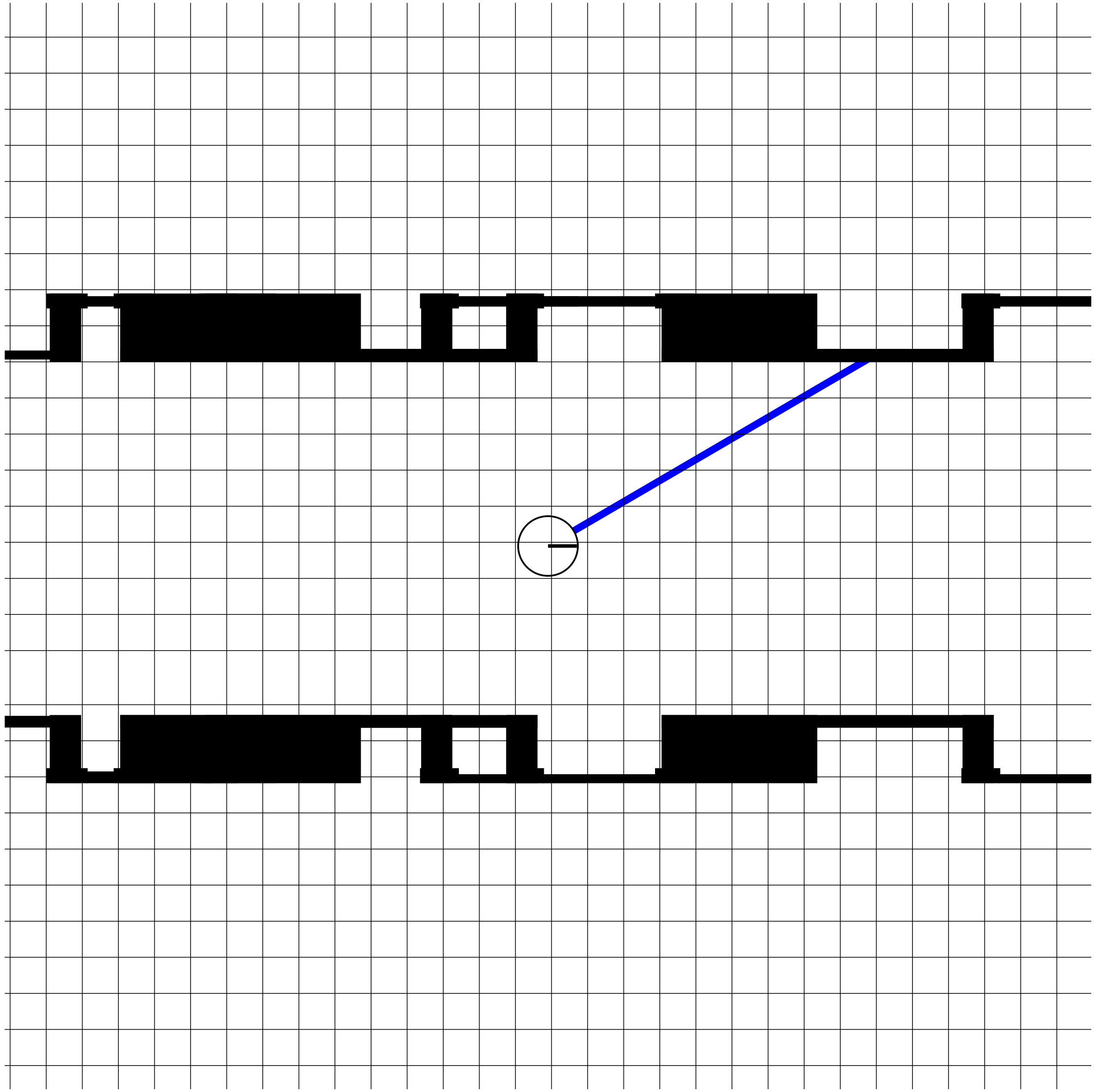
We stop once we get an *occupied cell*, and return z_d as the distance from the robot to the occupied cell.

If we do not find an occupied cell until the sensors maximum range, we return z_{max}

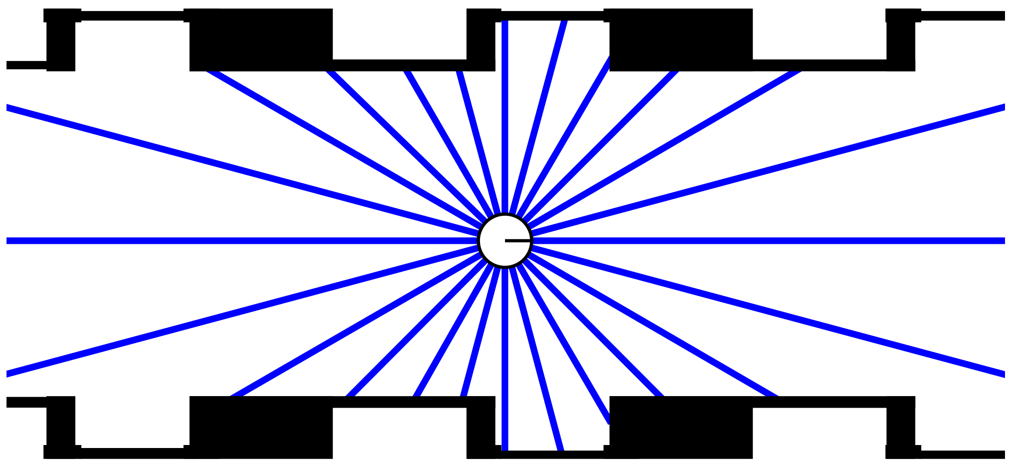


Ray Casting

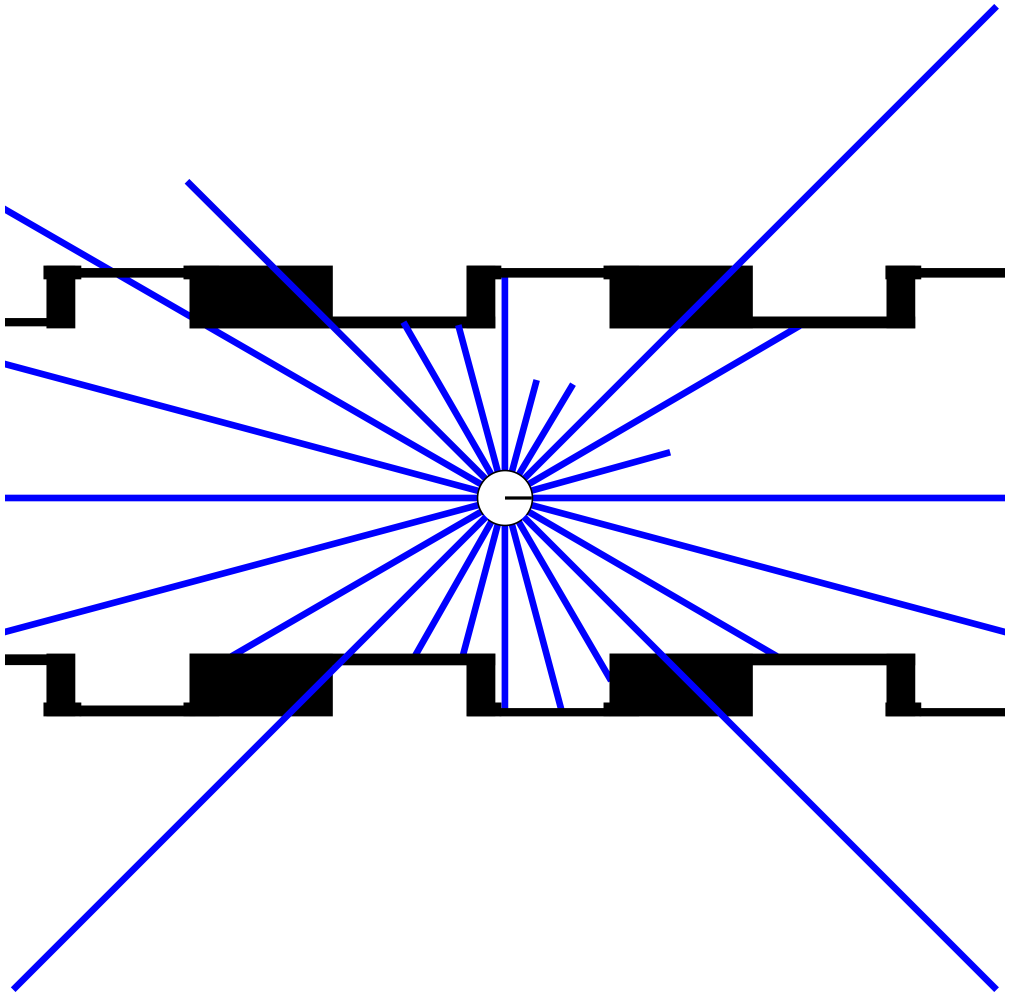
This approach is known as ray-casting and can be considerably expensive unless GPU acceleration is used.



Ideally, distances from the rangefinder z would be the same as the ones obtained from the map



However, typically we get something closer to

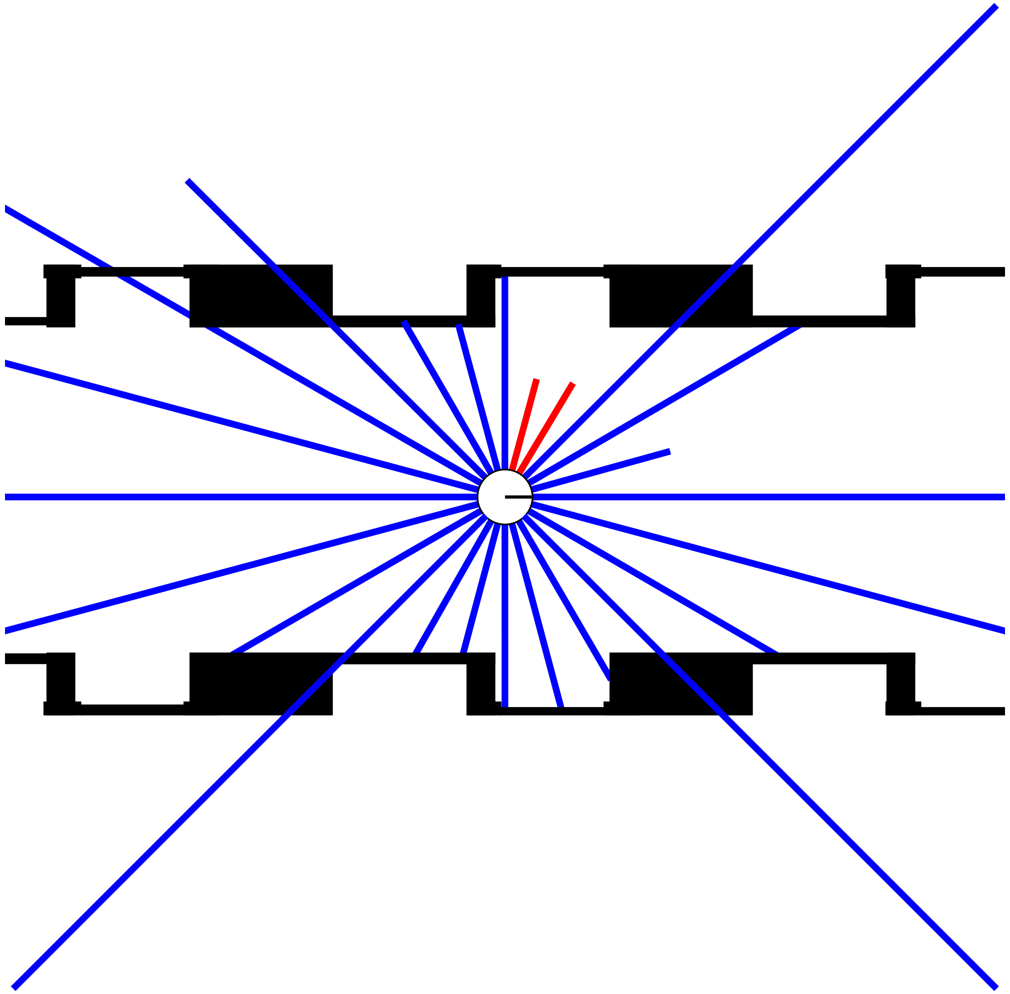


Sensor modeling

> We aim to model the **typical noise** observed by the sensor.

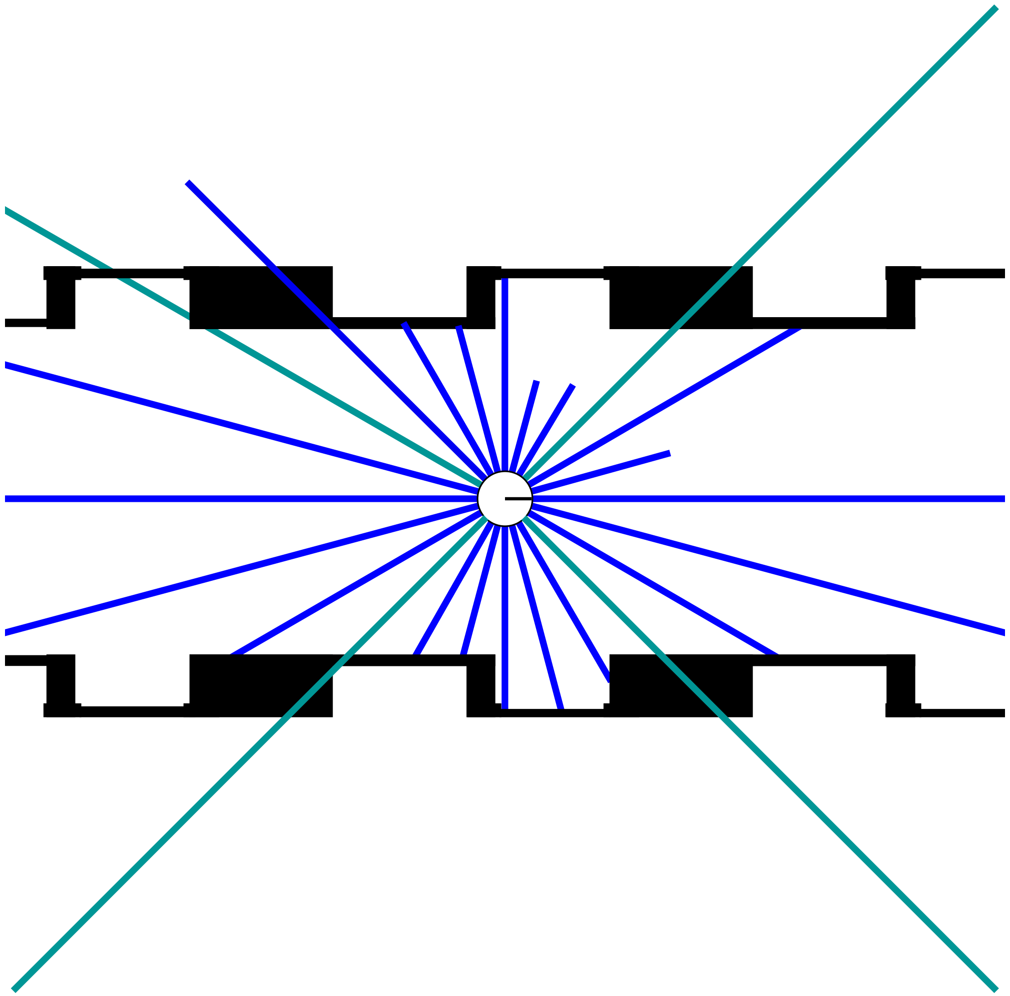
Occlusions

Due to people, unmapped objects, etc



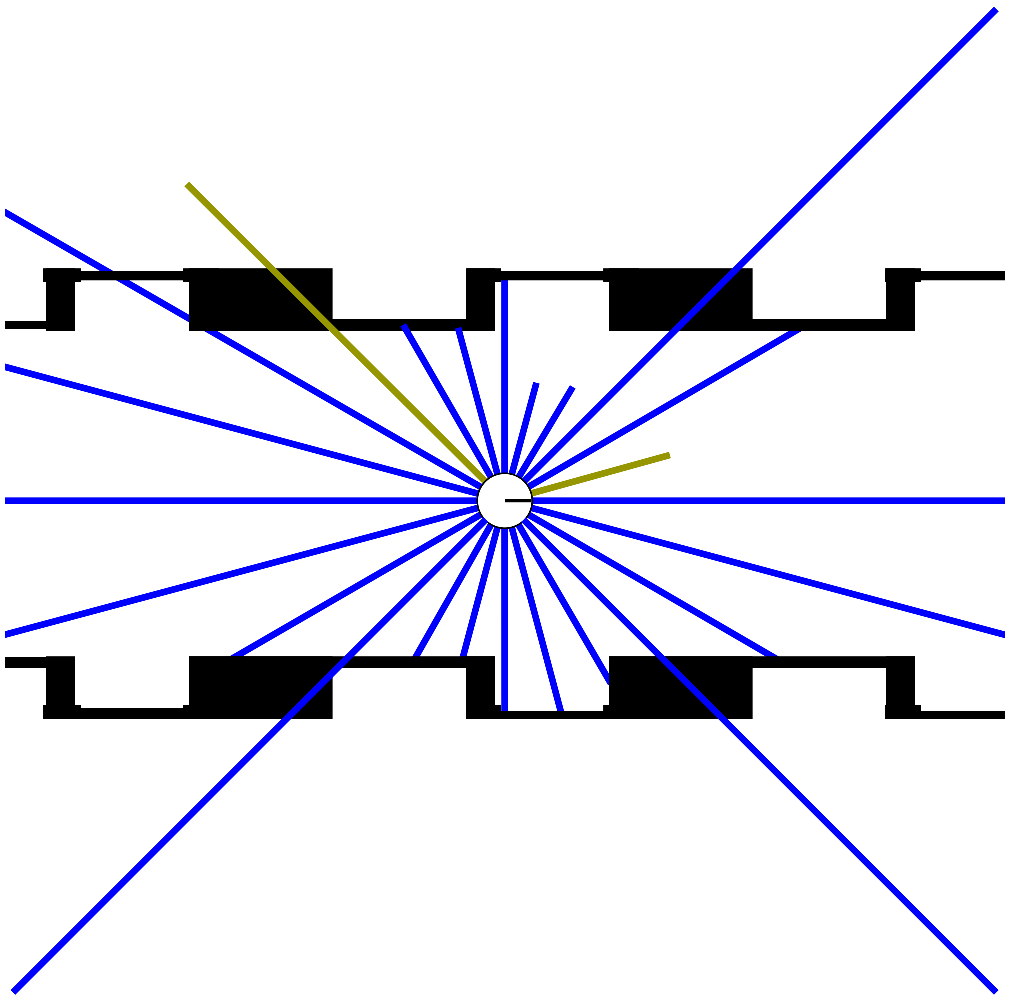
Misrecognitions

Objects outside sensor range (min/max), translucent objects, sensor errors, etc



Random errors

Electronic noise, reflective surfaces, etc



Pobabilistic Model

4 error types are used:

1. Measurement noise - p_{hit}
2. Occlusions - p_{short}
3. Missrecognitions - p_{max}
4. Random errors - p_{rand}

Measurement noise (p_{hit})

The noise around the correct distance z_d is used to model errors due to sensor noise/sensor accuracy.

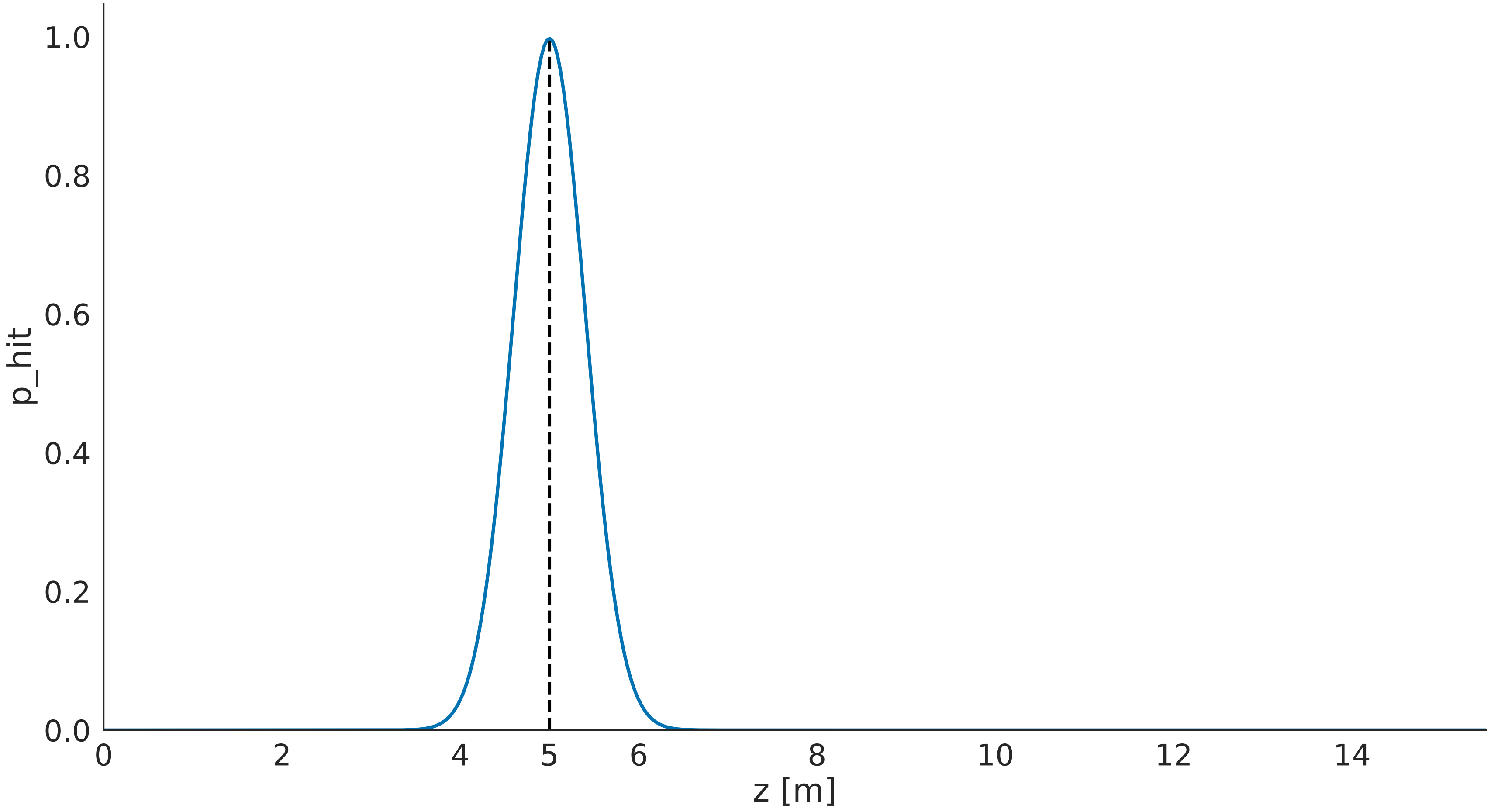
Modeled as a Gaussian around the true distance z_d with standard devition σ_{hit} and support $[0, z_{max}]$

$$p_{hit}(z) = \begin{cases} \eta \mathcal{N}(z | \mu = z_d, \sigma = \sigma_{hit}) & \forall z \in [0, z_{max}] \\ 0 & \text{otherwise} \end{cases}$$

with η being a normalization constant (due to the modified support) of z.

$$\eta = \frac{1}{CDF(z_{max}) - CDF(0)}$$

Measurement noise (p_{hit})



Occlusions (p_{short})

Probability error to account for dynamic objects (such as people), unmapped objects, etc.

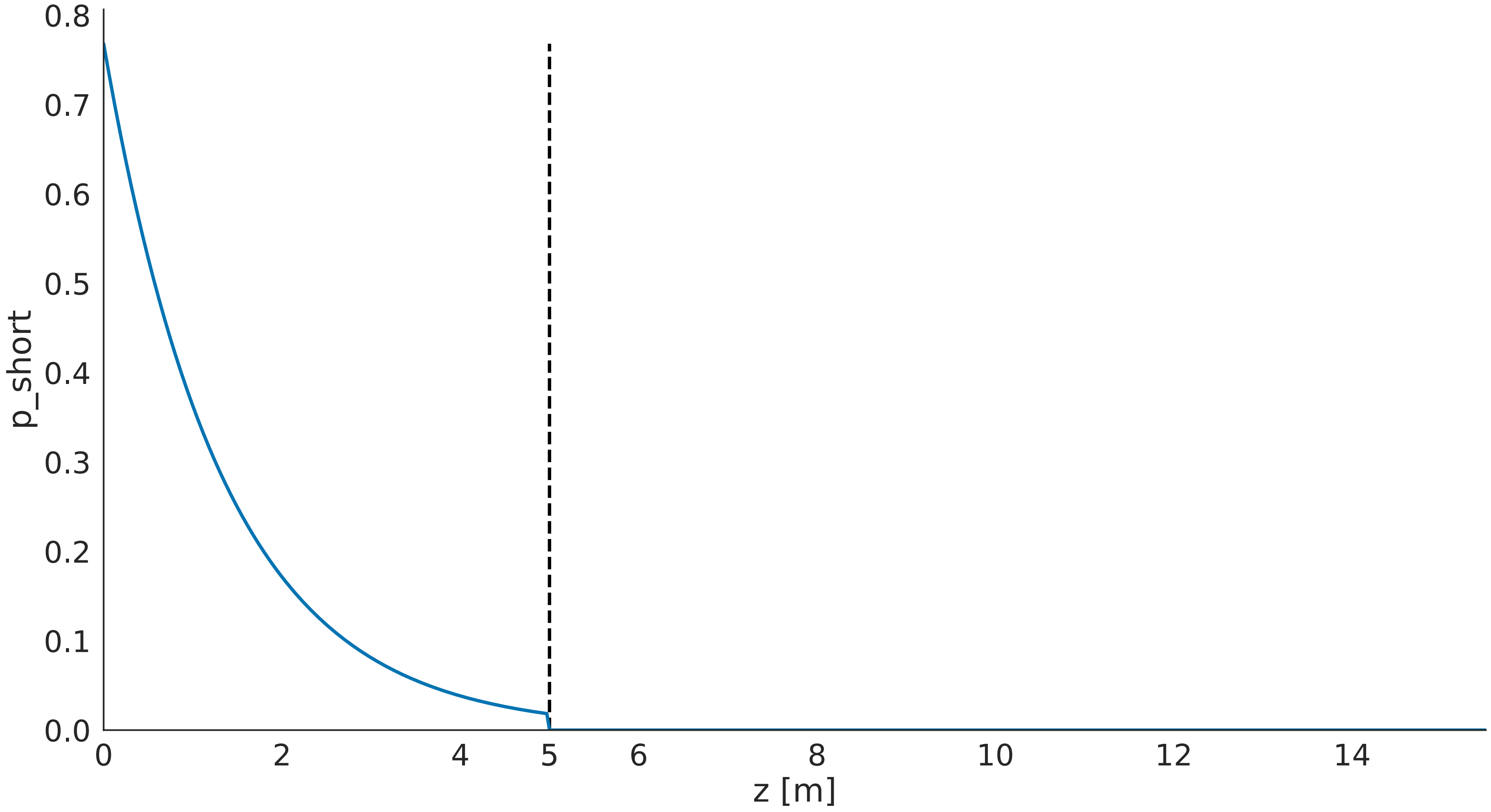
Modeled as an exponential distribution with exponential constant λ_{short} and support from zero until the true distance $[0, z_d]$

$$p_{short}(z) = \begin{cases} \eta \lambda_{short} \exp\left(-\lambda_{short} z\right) & \forall z \in [0, z_d] \\ 0 & \text{otherwise} \end{cases}$$

with η being a normalization constant

$$\eta = \frac{1}{1 - \exp\left(-\lambda_{short} z_d\right)}$$

Occlusions (p_{short})



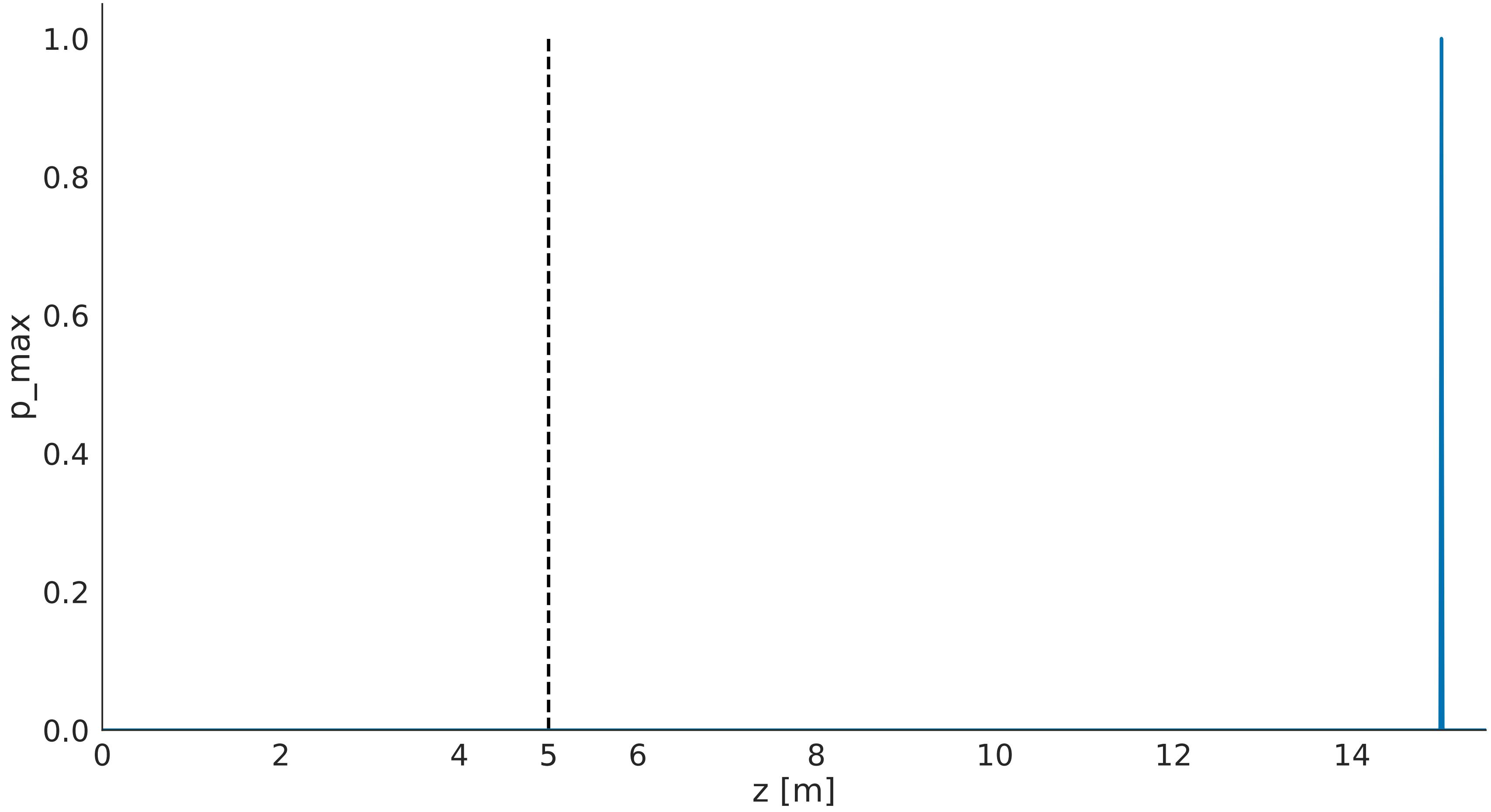
Missrecognitions (p_{max})

Probability error to account for objects outside sensor range (min/max), translucent objects, sensor errors, etc.

Modeled as a probability mass at z_{max}

$$p_{short}(z) = \begin{cases} 1 & \text{if} \quad z = z_{max} \\ 0 & \text{otherwise} \end{cases}$$

Missrecognitions (p_{max})



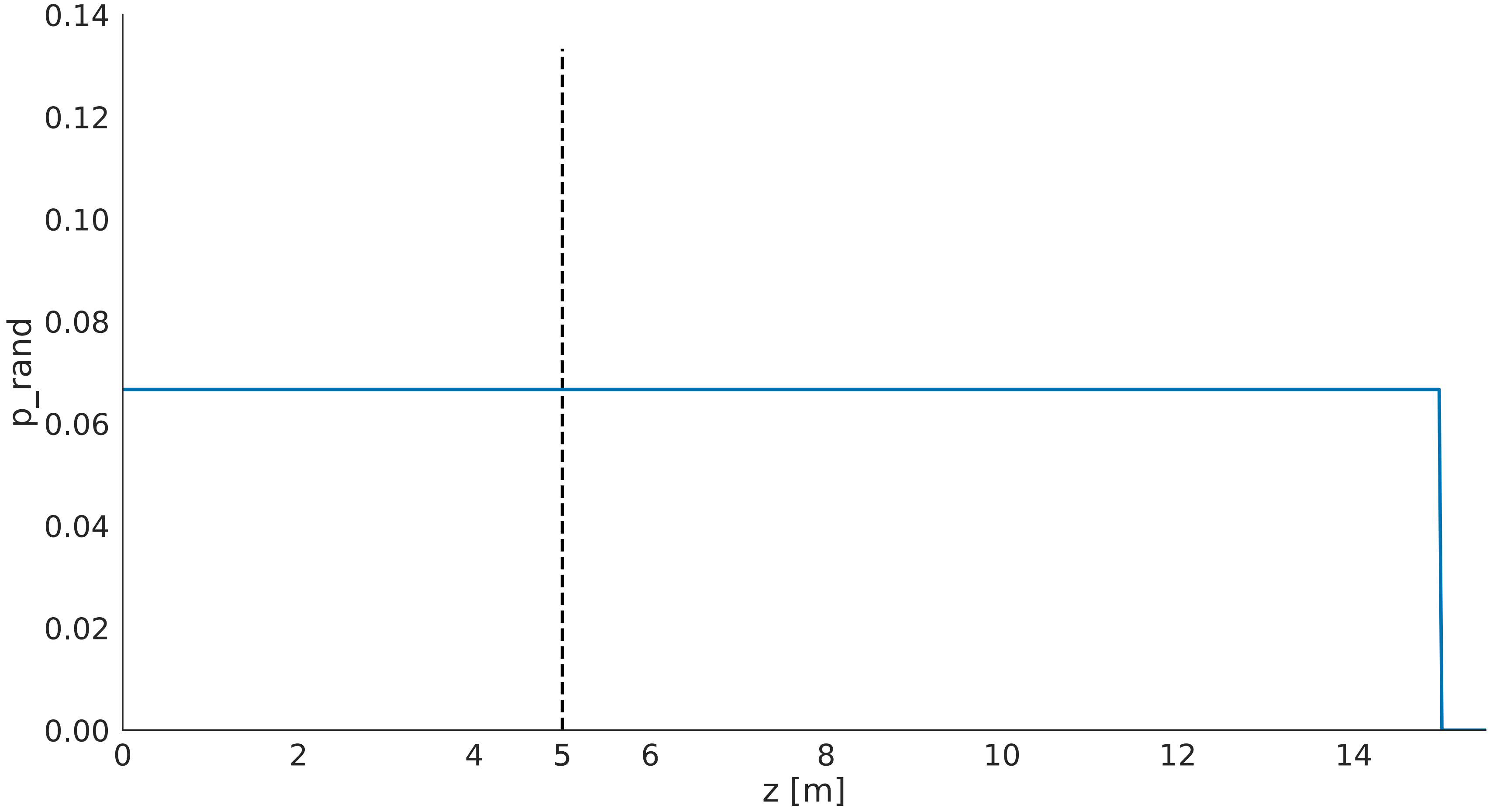
Random errors (p_{rand})

Probability error to account for electronic noise, reflective surfaces, etc.

Modeled as a uniform distribution with support $[0, z_{max}]$

$$p_{short}(z) = \begin{cases} \frac{1}{z_{max}} & \forall z \in [0, z_{max}] \\ 0 & \text{otherwise} \end{cases}$$

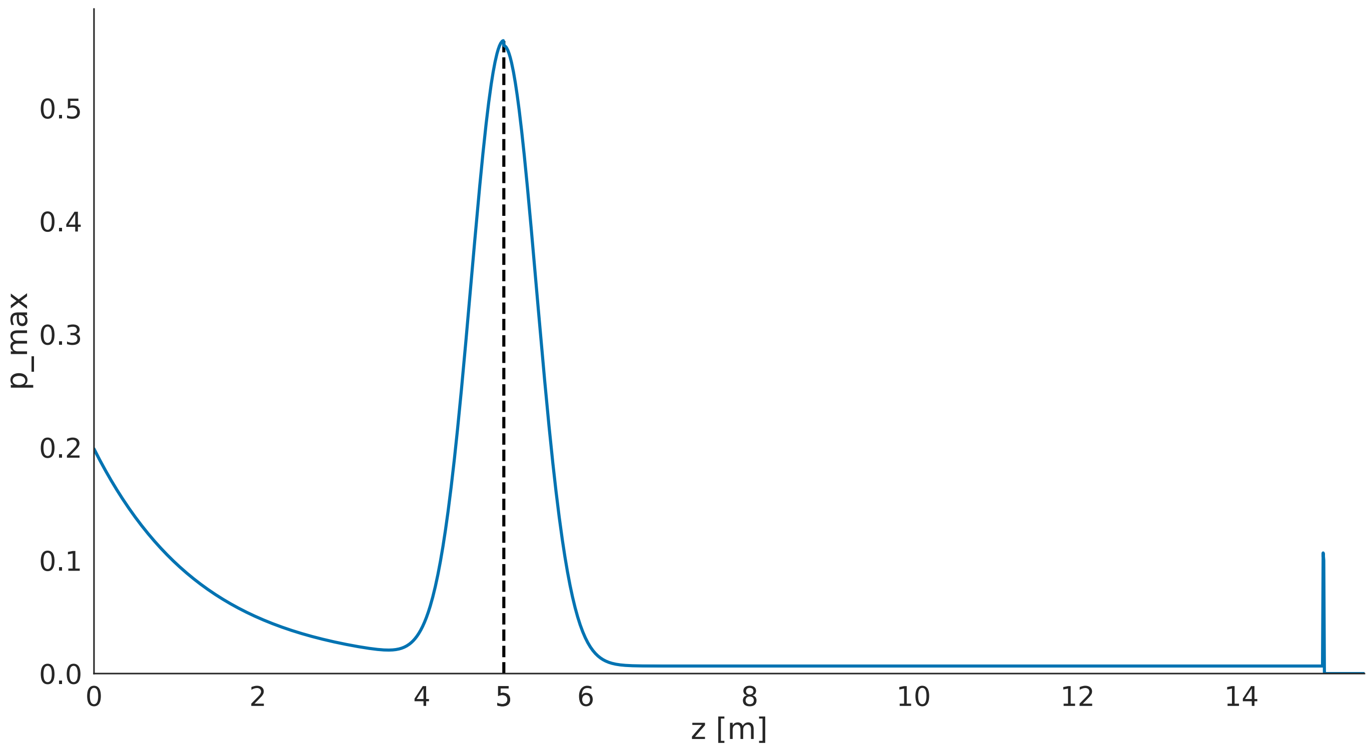
Random errors (p_{rand})



Beam model

The previous 4 errors are combined using a weighted constants $\alpha_{0:3}$, with $\sum \alpha_{0:3} = 1$

$$p_{tot} = \alpha_0 p_{hist} + \alpha_1 p_{short} + \alpha_2 p_{zmax} + \alpha_3 p_{rand}$$



Beam model

For each horizontal scan, we have m beams $\mathbf{z}_{0:m-1}$. If we consider iid,

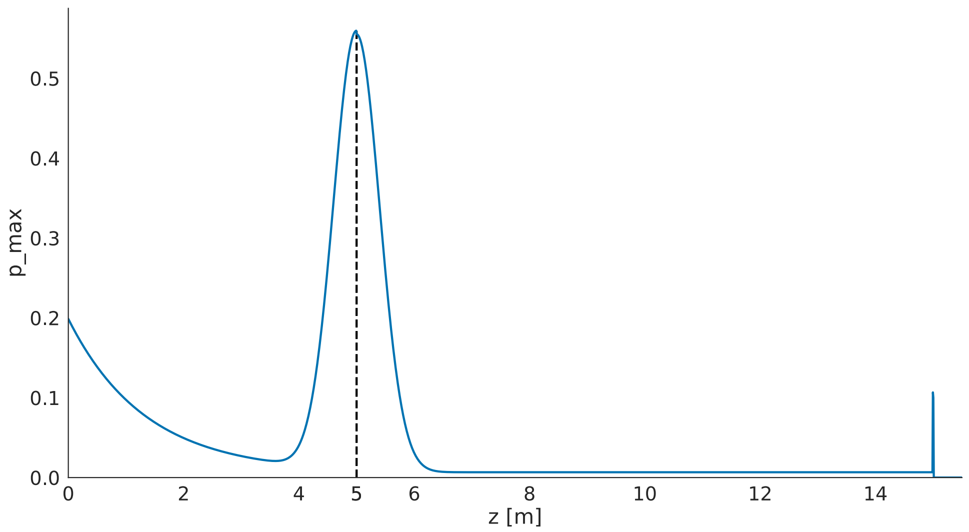
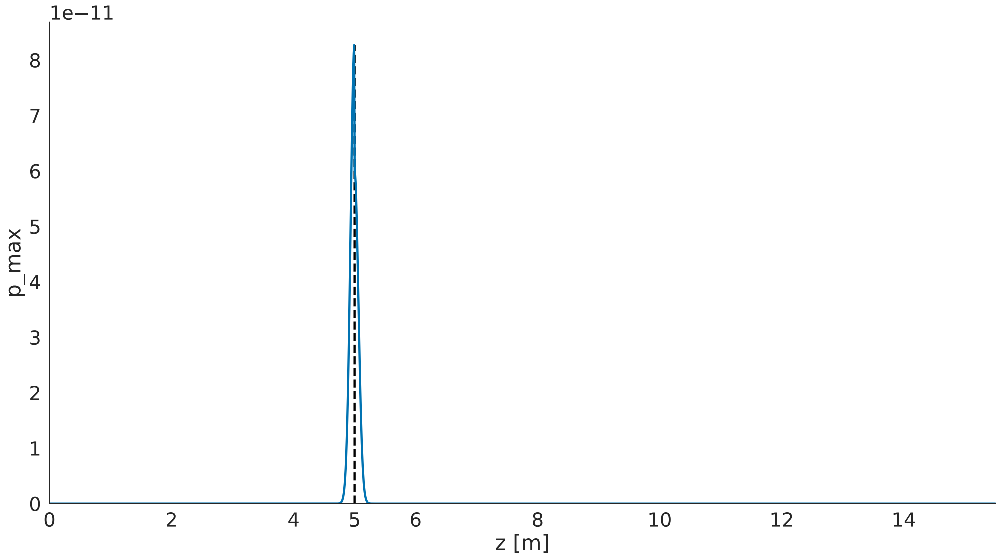
$$p(\mathbf{z}|\mathbf{x}, m) = \prod_{z_i \in \mathbf{z}} p_{tot}(z_i|\mathbf{x}, m)$$

Practical Considerations

Considering most rangefinders have hundreds of beams, computing this model for each beam is

1. Computationally expensive
2. Using the iid assumption tends to generate extremely peaked distributions.

IID assumption

single beam		40 beams
	→	

Practical considerations

In practice, it is common to average neighboring beams and only performs ray-tracing on this subset.

- Usually around 40 beams

Use a smoothing factor λ to avoid overconfident predictions

$$p(\mathbf{z}|\mathbf{x}, m) = \prod_{z_j \subset \mathbf{z}} p_{tot}(z_j|\mathbf{x}, m)^{\lambda}$$

- λ is often set to 1/(number of beams in the subset used)

Likelihood Field Model

An approximation that generates similar likelihoods but lacks a physical explanation.

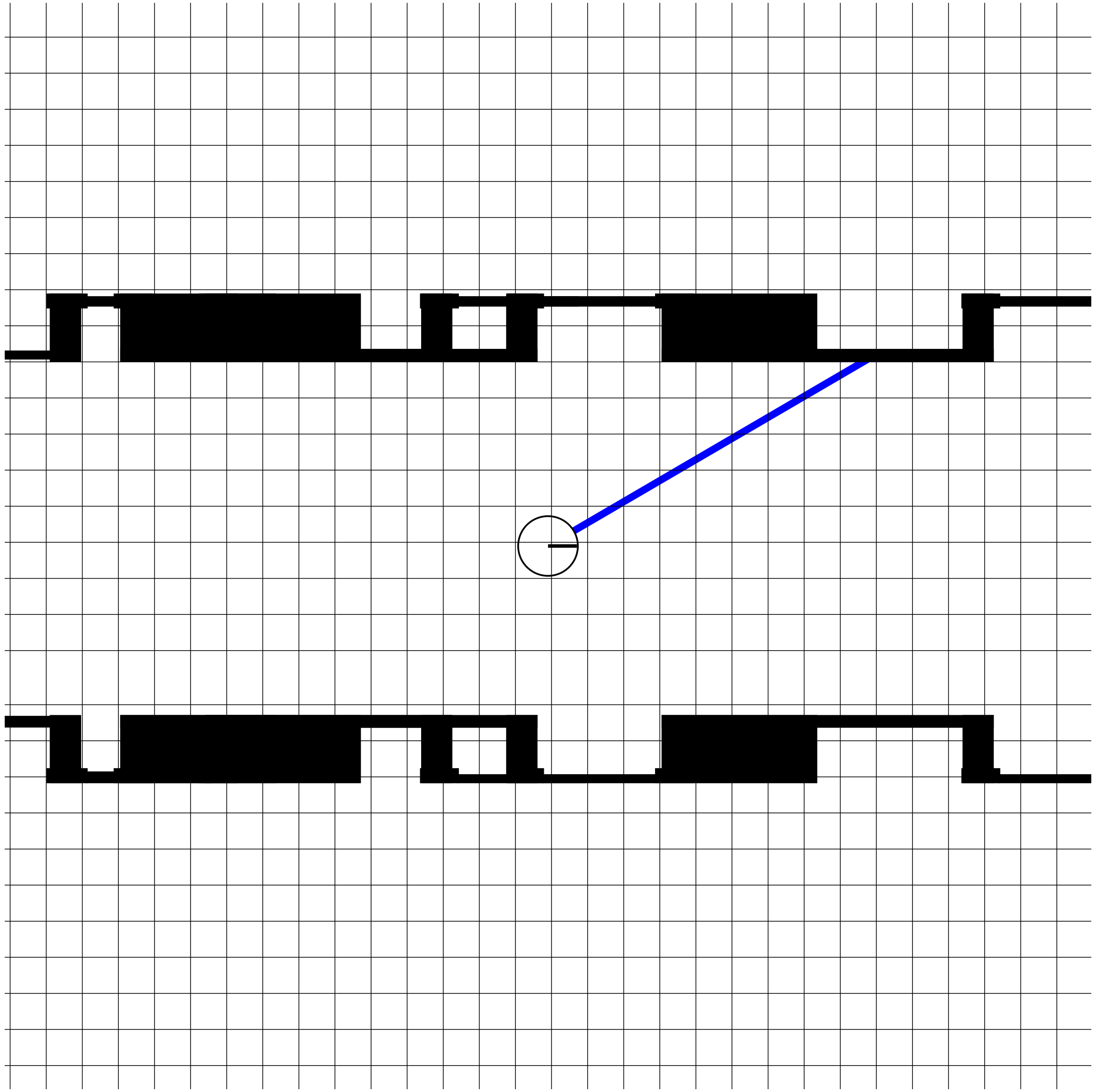
Computationally efficient and works well in practice

Likelihood Fields Model

Compute the probability of a measurement based on the euclidean distance transform at the beam's endpoint.

End-point

Measurements z are in the robot's reference frame, so we transform it into the global reference frame.



Distance Transform

a.k.a distance map, distance field

For each cell in a grid map, it assigns its value as the distance to the closest obstacle.

Euclidean Distance Transform:

Occupancy map (0:empty, 1:occupied) $\rightarrow$ Distance Transform

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1.4 & 1 & 1 & 1.4 & 2.2 & 2.8 & 2.2 & 2 \\ 1 & 0 & 0 & 1 & 2 & 2.2 & 1.4 & 1 \\ 1 & 0 & 0 & 1 & 2 & 2 & 1 & 0 \\ 1.4 & 1 & 1 & 1.4 & 2.2 & 2 & 1 & 0 \\ 2.2 & 2 & 2 & 2.2 & 2.8 & 2 & 1 & 0 \end{bmatrix}$$

Measurement Noise (p_{hit})

Errors due to sensor noise/sensor accuracy are modeled using the value of the distance transform d at the endpoint's location

Modeled as a zero-mean Gaussian with standard deviation σ_{hit}

$$p_{hit}(z) = \mathcal{N}(d|\mu = 0, \sigma = \sigma_{hit})$$

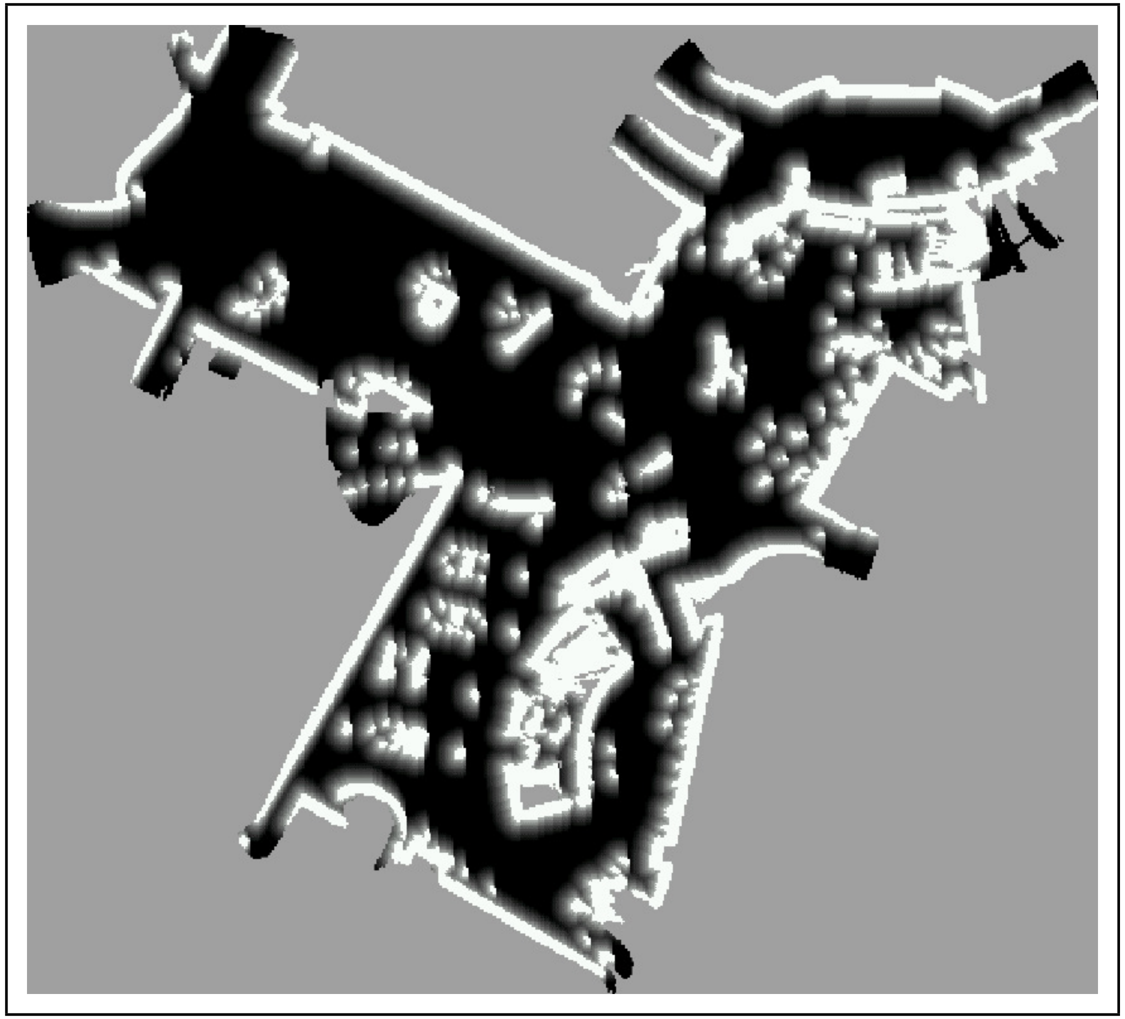
Missrecognitions (p_{max})

Same as before, using a point mass probability p_{max} if the range is z_{max}

Random errors (p_{rand})

Same as before, using an uniform distribution p_{rand} with scope $[0,\ z_{max}]$

Example

Disadvantages

1. Does not consider dynamic obstacles
2. Rangefinder *sees through walls*

Advantages

Extremely fast

Gaussian Distribution over Distance Transform can be pre-computed, making online computation simply finding end-points for beams and a look-up table to retrieve p_{hit}