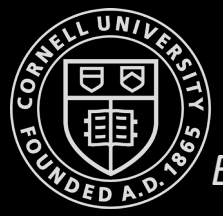
# ECE 4960

Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

# Fast Robots

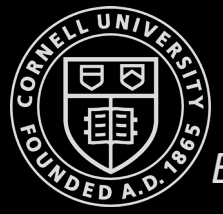
## Sensor Models



# Lab 8 – Stunts

- **It's time to vote up the best stunts: https://tinyurl.com/vp5wrten**
  - The voting is active until Sunday night (April 24th)
  - Remember, you get 10 points to distribute among your teammates for best open-loop stunt and 1 point for best blooper video*
  - https://cei-lab.github.io/ECE4960-2022/StudentPages.html

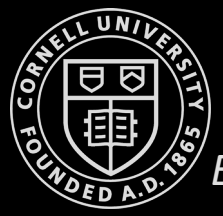
* We will discount any votes given to your own submission, and we will normalize back to 10 points if you accidentally give out more than that.



Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

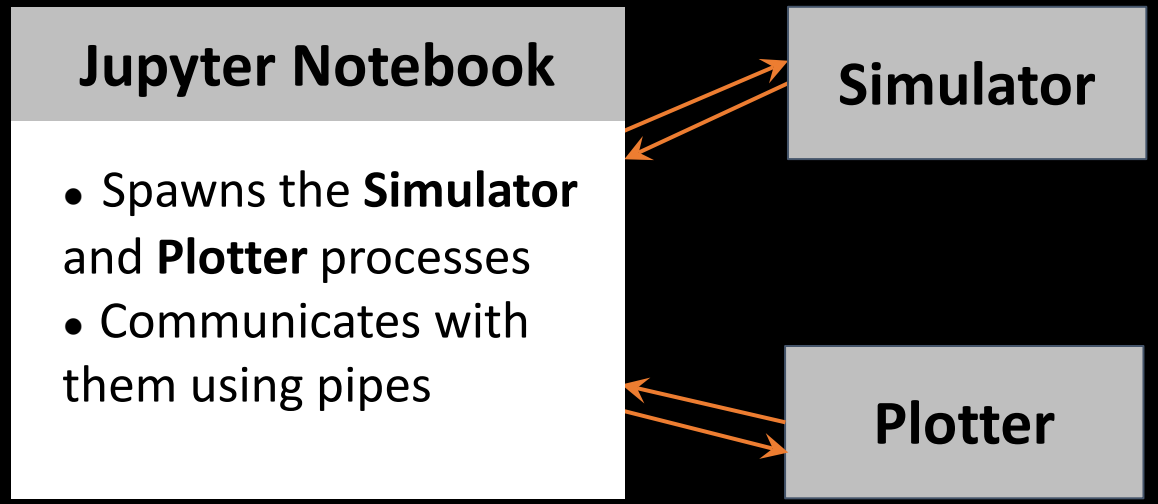
# Fast Robots

## Lab 10-13 overview



# Lab 10 – Simulation Software

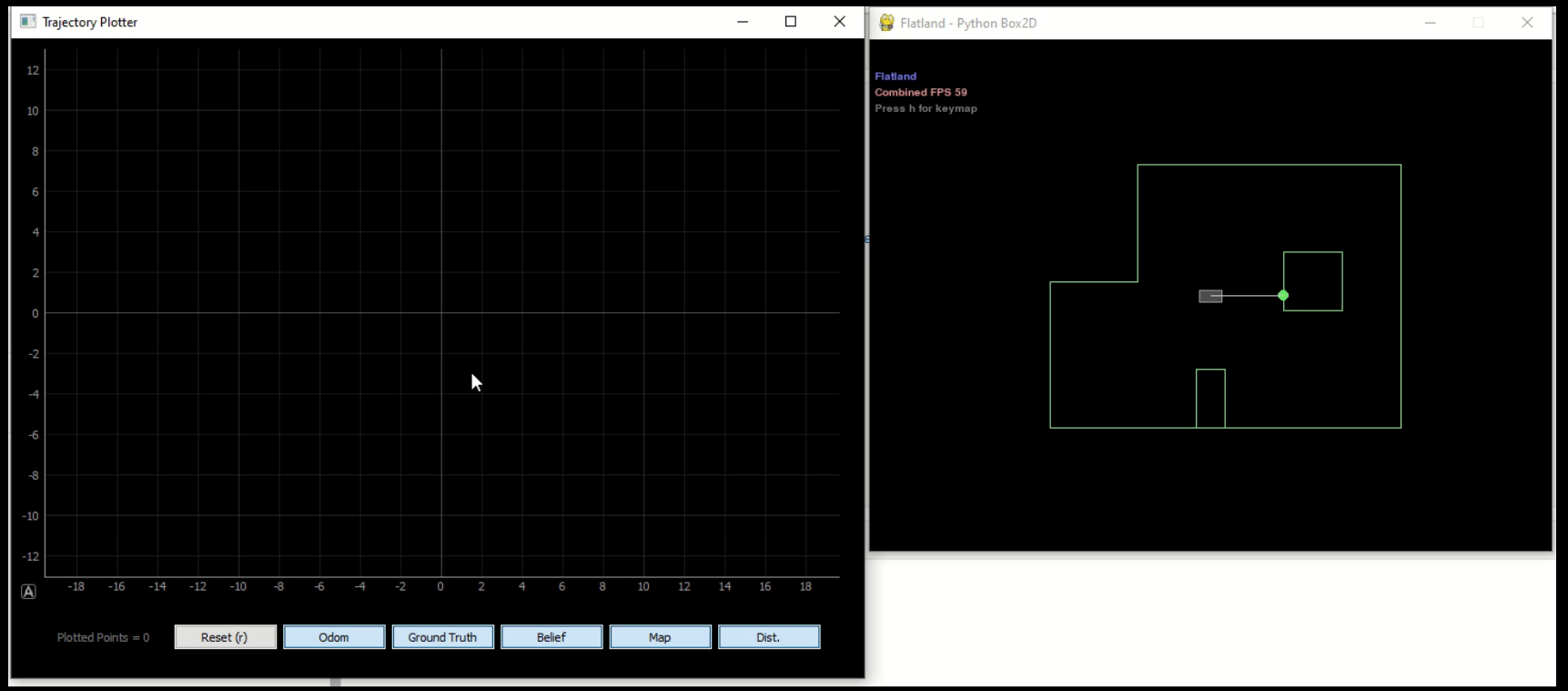
- Multiple processes
  - Simulator
    - Robot
      - Motion
      - Ground truth
    - YAML (map and other parameters)
  - Plotter
  - Controller
    - Get odometry pose, get and plot sensor data, move the robot, etc.



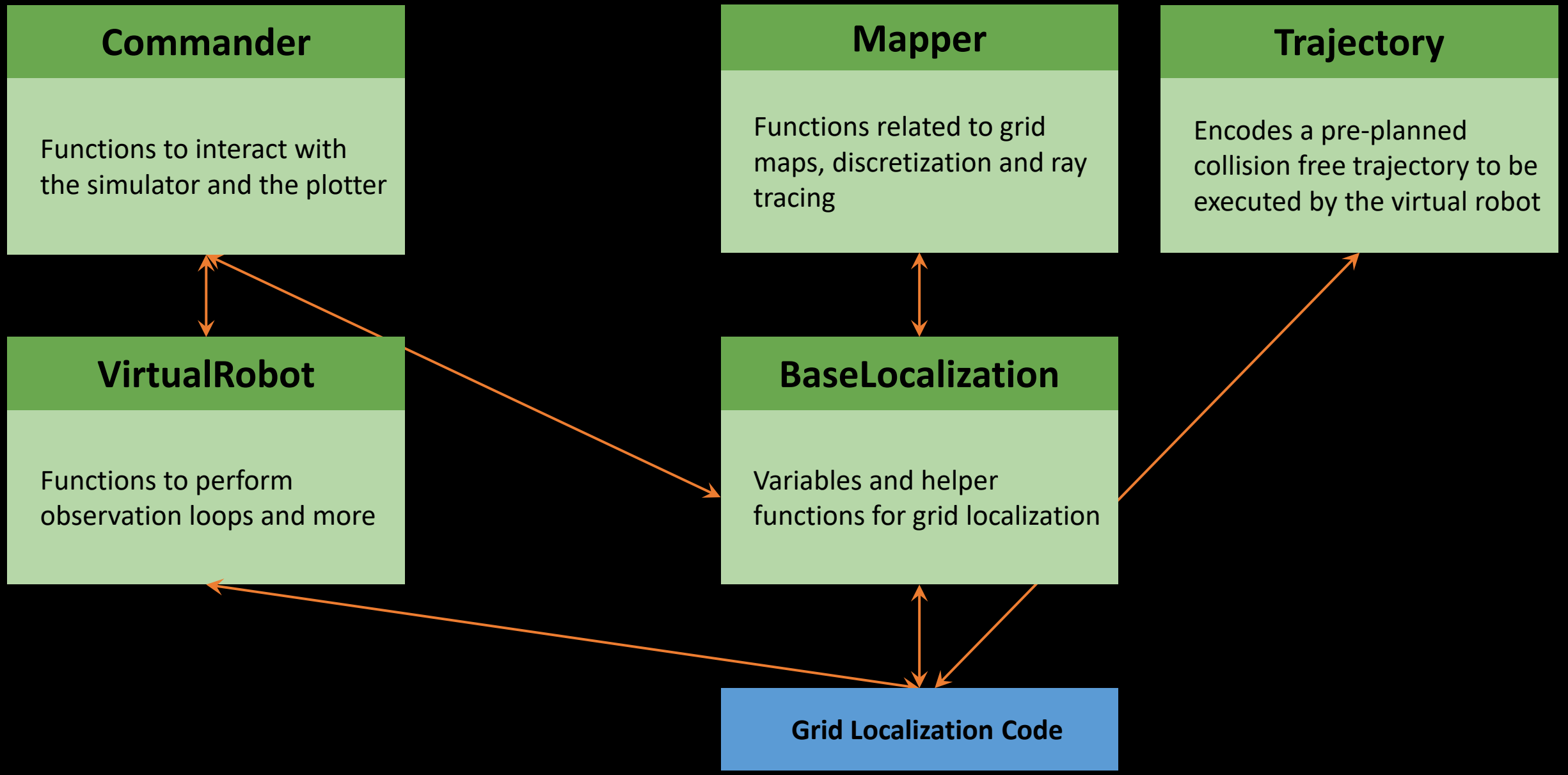
# Lab 10 - Simulation

## Commander

Functions to interact with the simulator and the plotter



# Lab 11 - Localization on the virtual robot



# Lab 11 - Localization

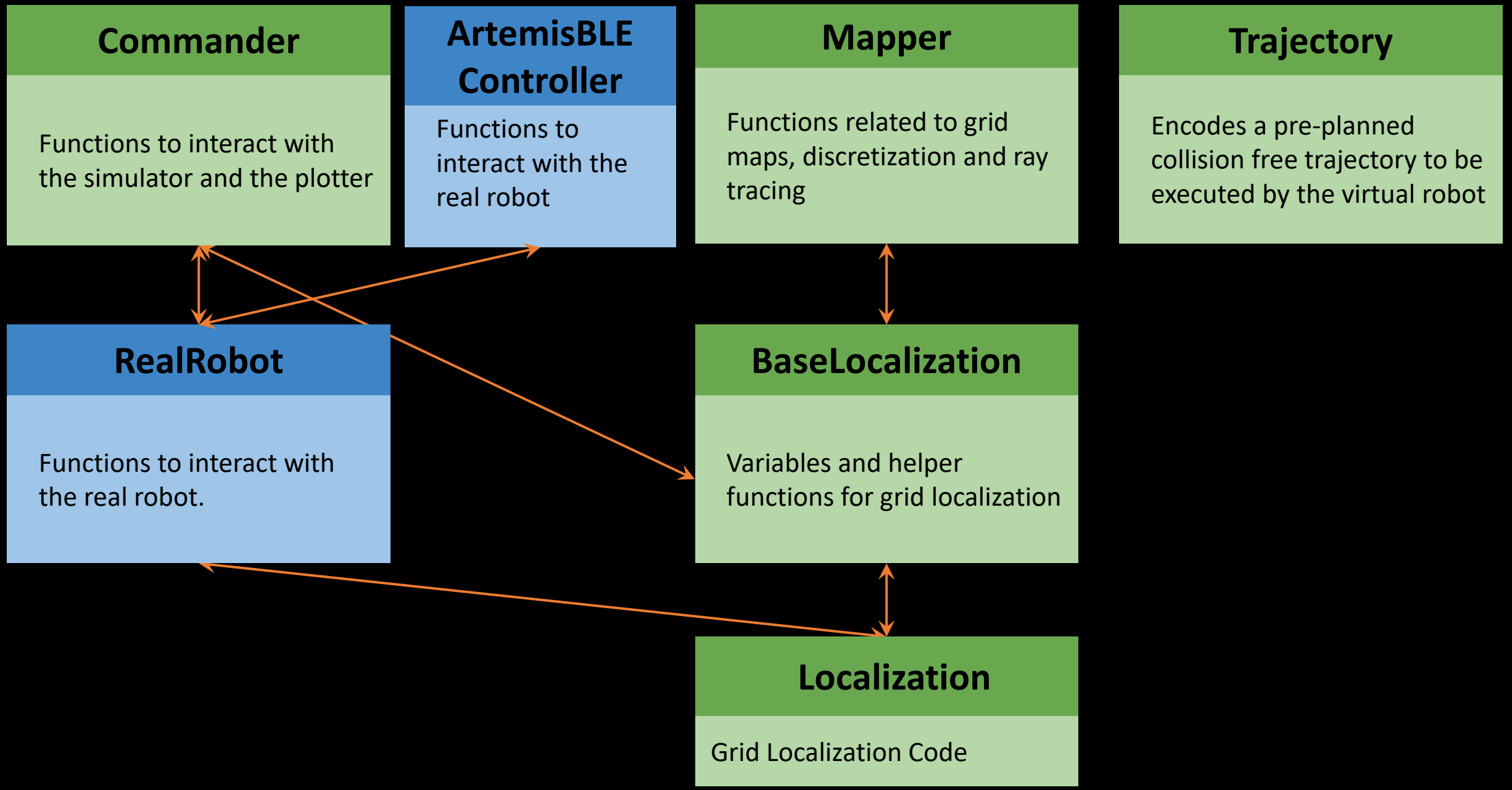
- Technically just 5-11 lines of code
- But you need to get it just right!
- And the underlying framework is pretty massive
- Read the background information carefully
- Feel free to discuss your approach with your TAs before you start
- Computation time is critical…
  - Prediction step
  - Update step

- What is the dimension of your state space?
- How many nested for loops do you need?

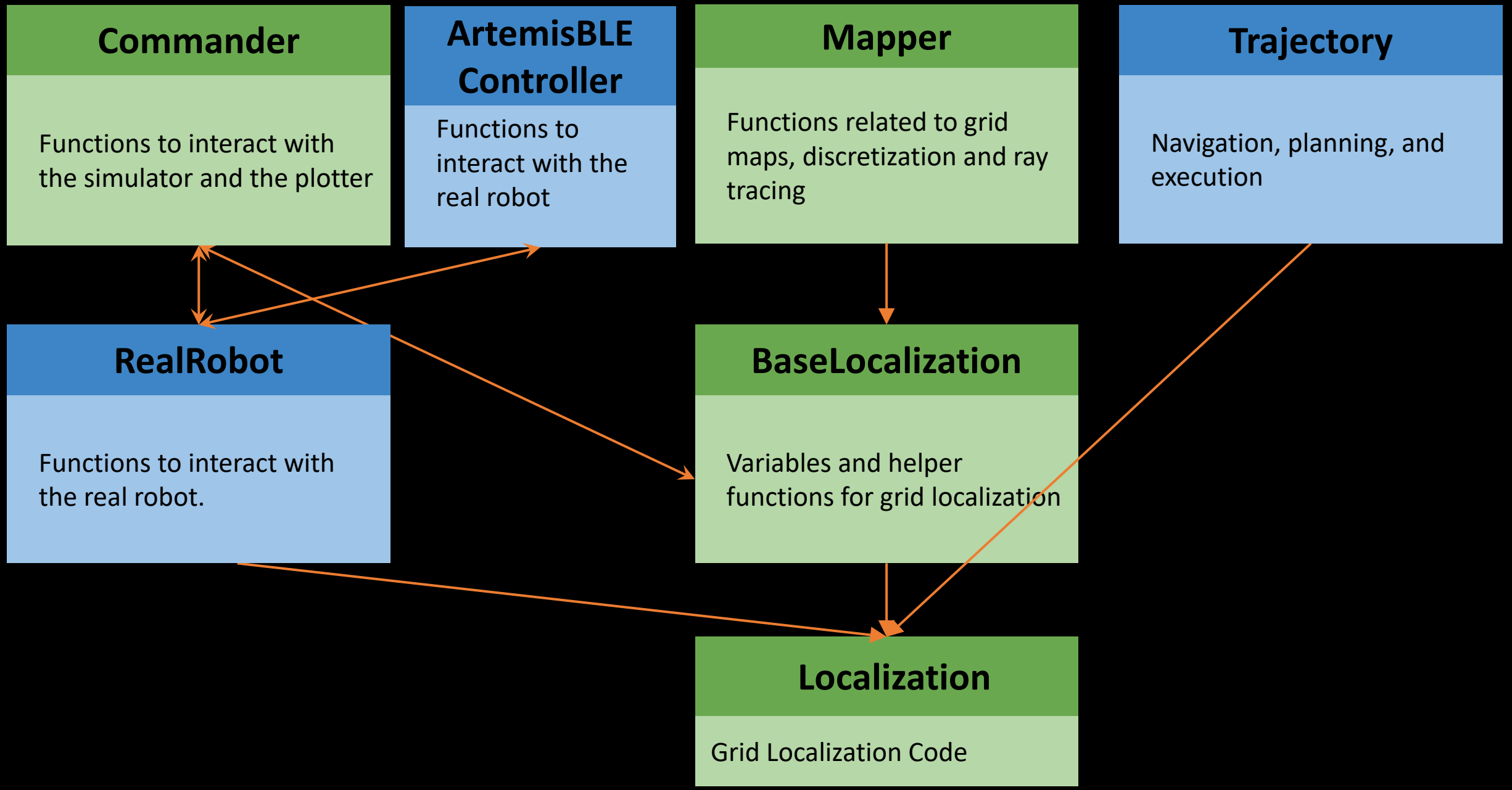
**Algorithm Bayes_Filter** $(bel(x_{t-1}), u_t, z_t)$:

1. for all $x_t$ do
2. $\quad \overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t|u_t, x_{t-1})\ bel(x_{t-1})$
3. $\quad bel(x_t) = \eta\, p(z_t|x_t)\ \overline{bel}(x_t)$
4. endfor
5. return $bel(x_t)$

# Lab 12 - Localization on the real robot



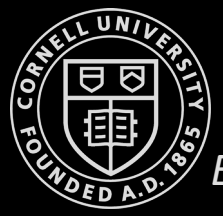
# Lab 13 - Localization and planning on the real/virtual robot



Prof. Kirstin Hagelskjær Petersen
kirstin@cornell.edu

# Fast Robots

## Sensor models

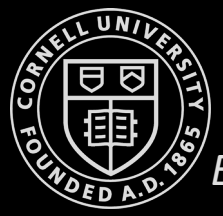


# Bayes Filter

**Lecture 18: Motion model**
- Odometry model
- Velocity model

1. **Algorithm Bayes_Filter** $(bel(x_{t-1}), u_t, z_t)$:
2. for all $x_t$ do
3. $\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t|u_t, x_{t-1})\ bel(x_{t-1})$ [ Prediction Step ]
4. $bel(x_t) = \eta\, p(z_t|x_t)\ \overline{bel}(x_t)$ [ Update/Measurement Step ]
5. endfor
6. return $bel(x_t)$

Measurement Probability / Sensor Model



# Probabilistic Sensor Model

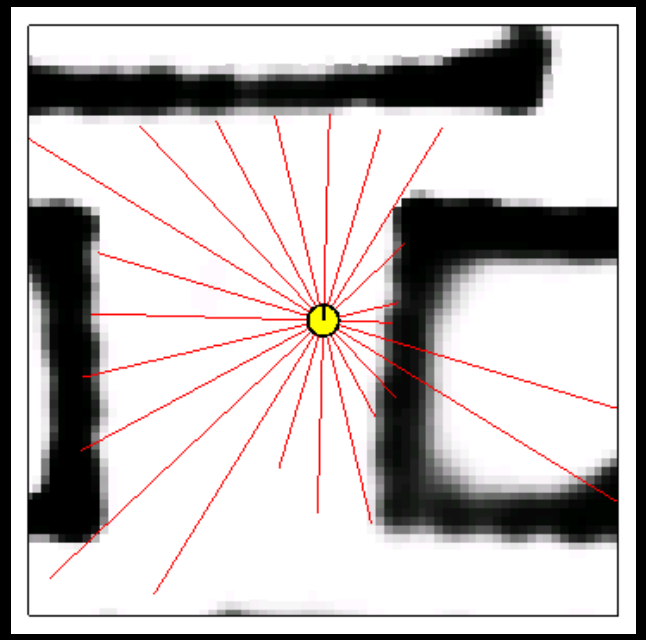
$$p(z|x)$$

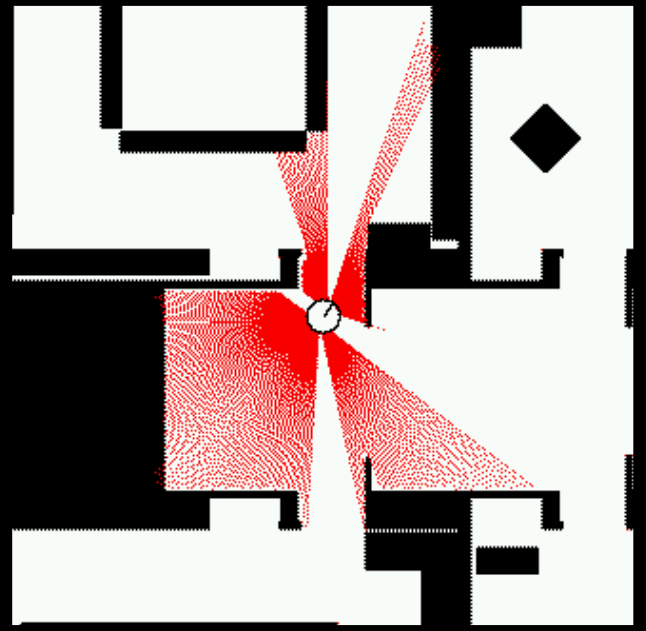
# Sensors for Mobile Robots

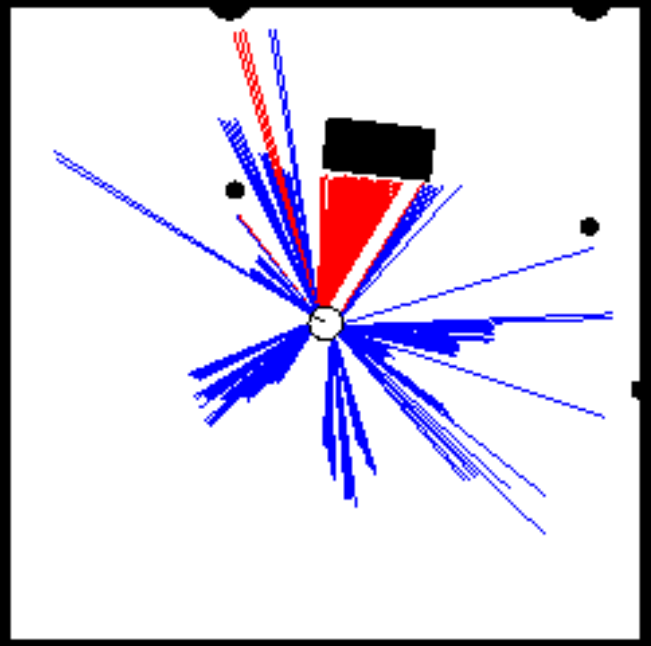
- **Contact Sensors:** Bumpers
- **Internal/Proprioceptive Sensors**
  - Accelerometers (spring-mounted masses)
  - Gyroscopes (spinning mass, laser light)
  - Compasses, inclinometers (earth magnetic field, gravity)
- **Range Sensors**
  - Sonar (time of flight)
  - Radar (phase and frequency)
  - Laser range finders (triangulation, tof, phase)
  - Infrared (intensity)
- **Visual Sensors:** Cameras
- **Satellite-based sensors:** GPS

# Sensor Model

- Probabilistic robotics explicitly models the noise in exteroceptive sensor measurements
  - *What about proprioceptive/odometry sensors?*
- Where does that noise come from?



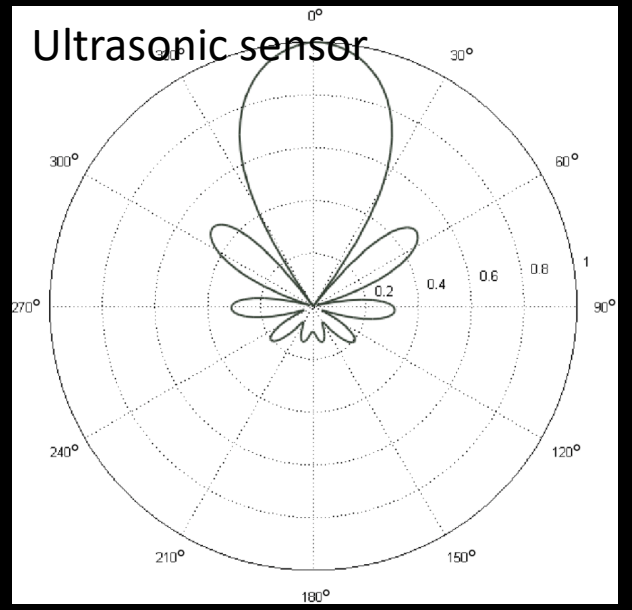


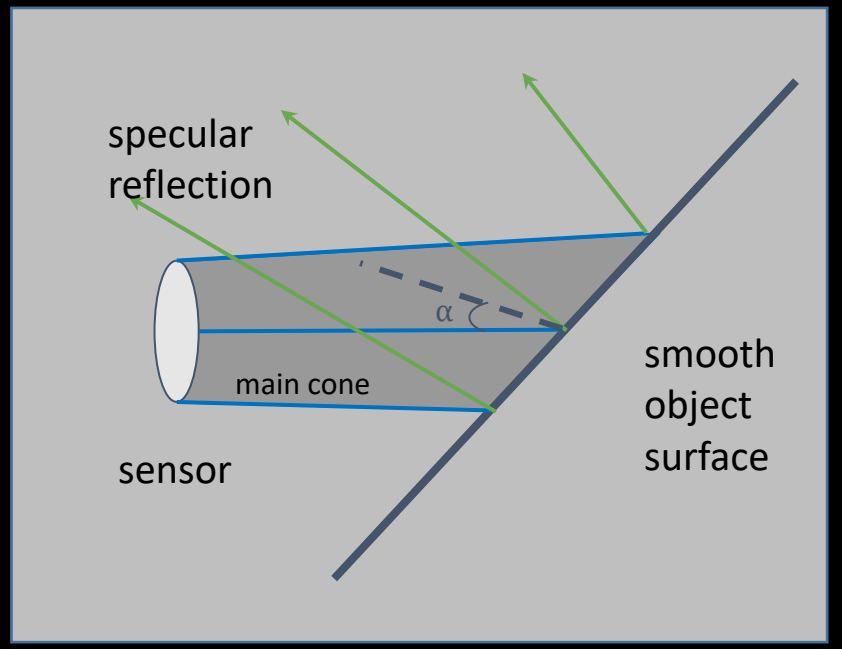


# Range Sensor Inaccuracies (“noise”)

- Larger readings
  - Surface material
  - Angle between surface normal and direction of sensor cone
  - Width of the sensor cone of measurement
  - Sensitivity of the sensor cone

- Shorter readings
  - crosstalk between different sensors
  - unmodeled objects in the proximity of the robot, such as people



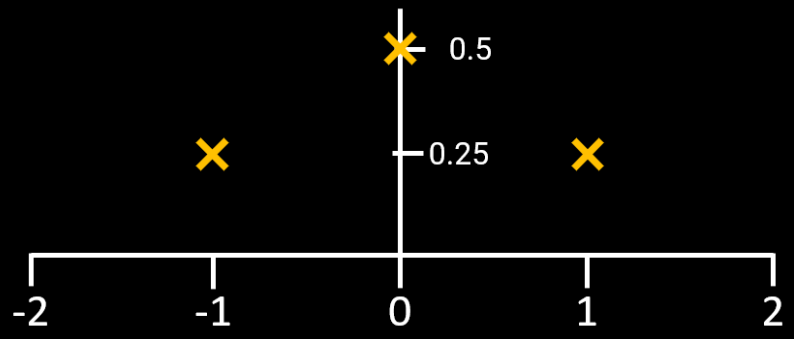
# Probabilistic Sensor Model

- Perfect sensor models…
  - $z = f(x)$
  - …practically impossible
  - …computationally intractable
- Practical sensor models…
  - $p(z|x)$

- Three common sensor models
  - Beam model
  - Likelihood model
  - Feature-based model

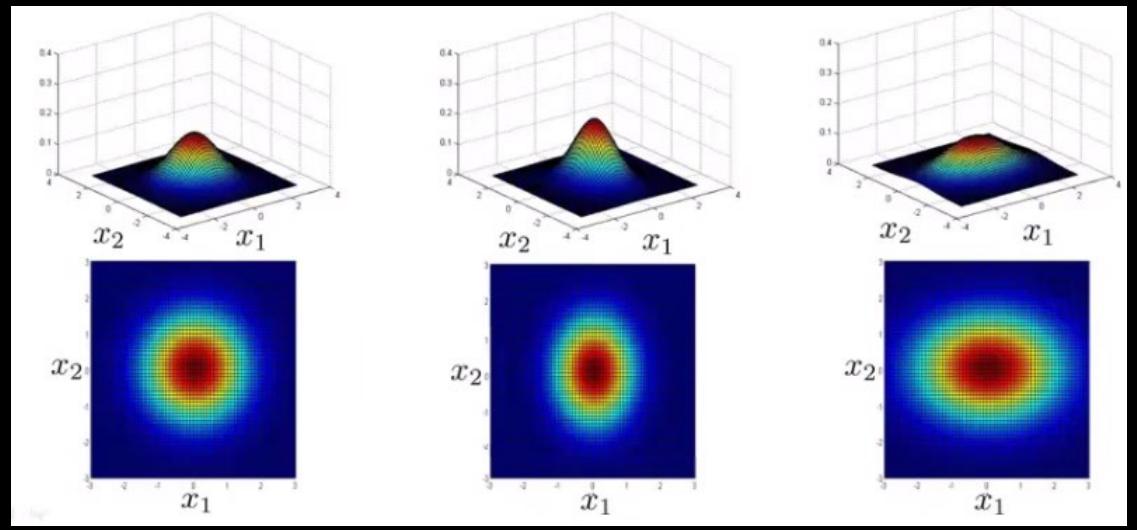
Up till now our sensor models have been simple

- p(z=correct)
- p(z|X) for a small state space
-

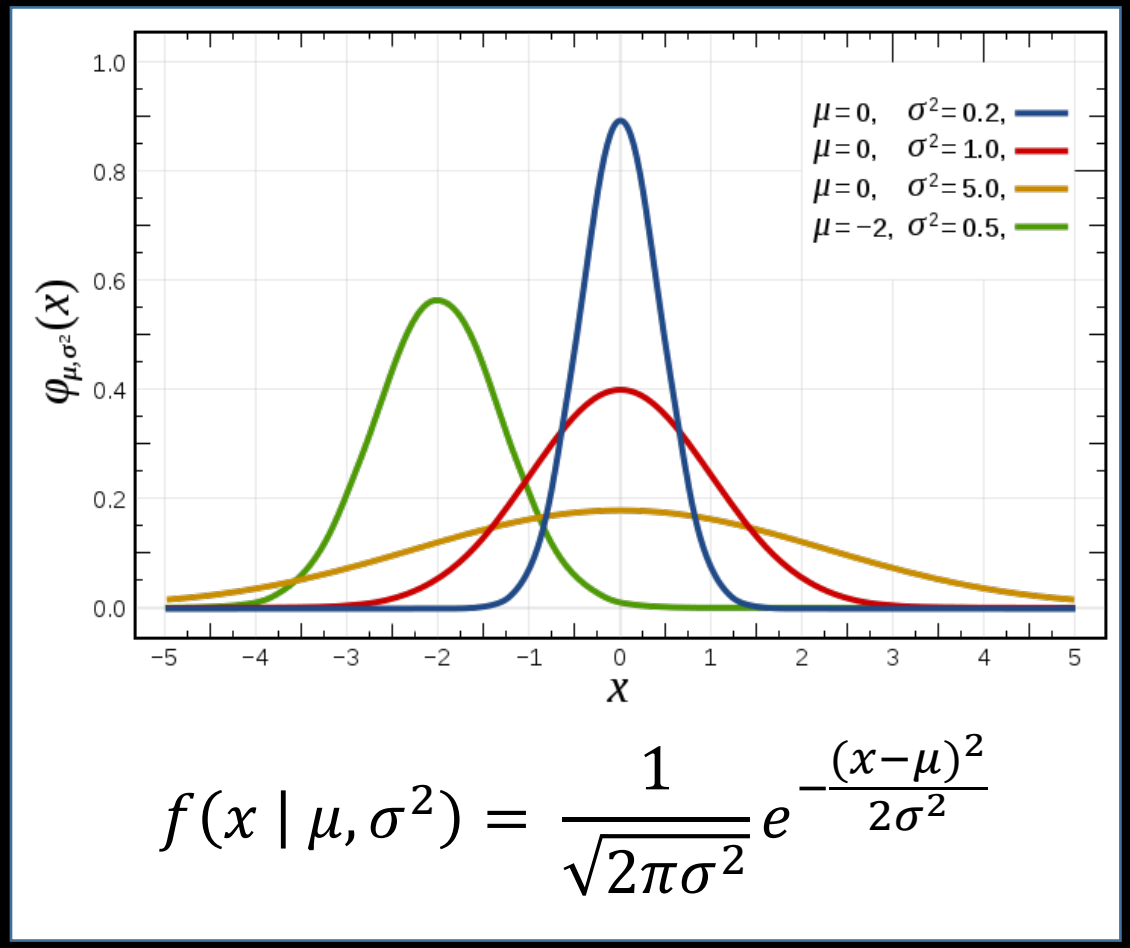


# Quick Detour: Gaussian Distribution

- Or "normal distribution" or "bell curve"
- Defined by two parameters
  - mean $\mu$
  - standard deviation $\sigma$
- Can be defined for multidimensional data




1D Gaussian Probability Density Function




$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

# Beam Model

# Beam Model of Range Finders

- Let there be $K$ individual measurement values within a measurement $z_t$

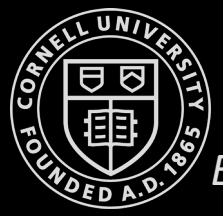
$$z_t = \{z_t^1, z_t^2 \ldots, z_t^K\}$$

- Individual measurements are independent given the robot state

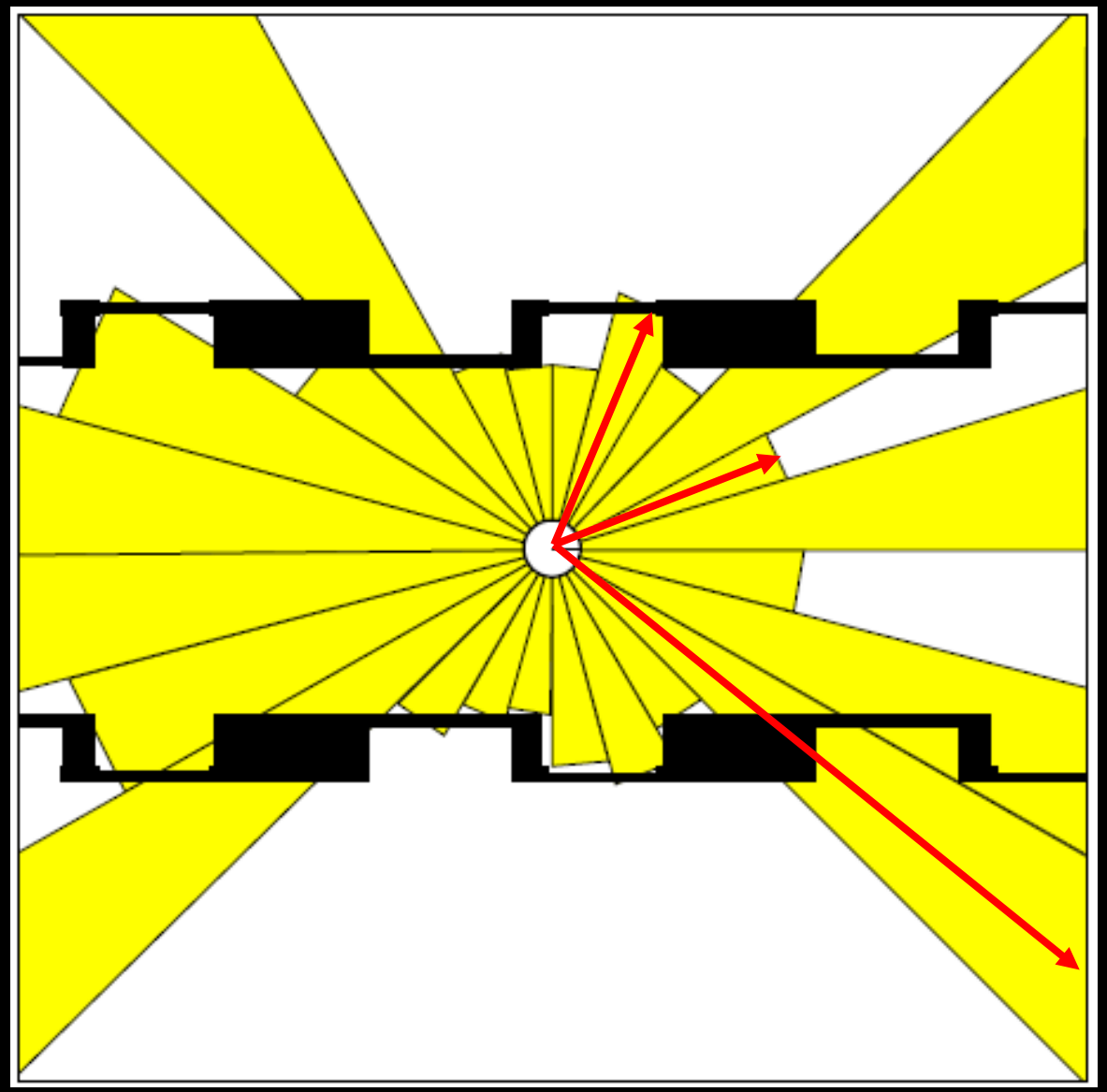
$$p(z_t|x_t, m) = \prod_{k=1}^{K} p\left(z_t^k \middle| x_t, m\right)$$

(Sensor measurements are caused by real world objects)

- *Realistically, dependencies will exist*
  - People, errors in the map model *m*, approximations in the posterior, etc.



# Typical Measurement Errors of an Range Measurements



# Typical Measurement Errors of an Range Measurements

1. Correct Range Measurements

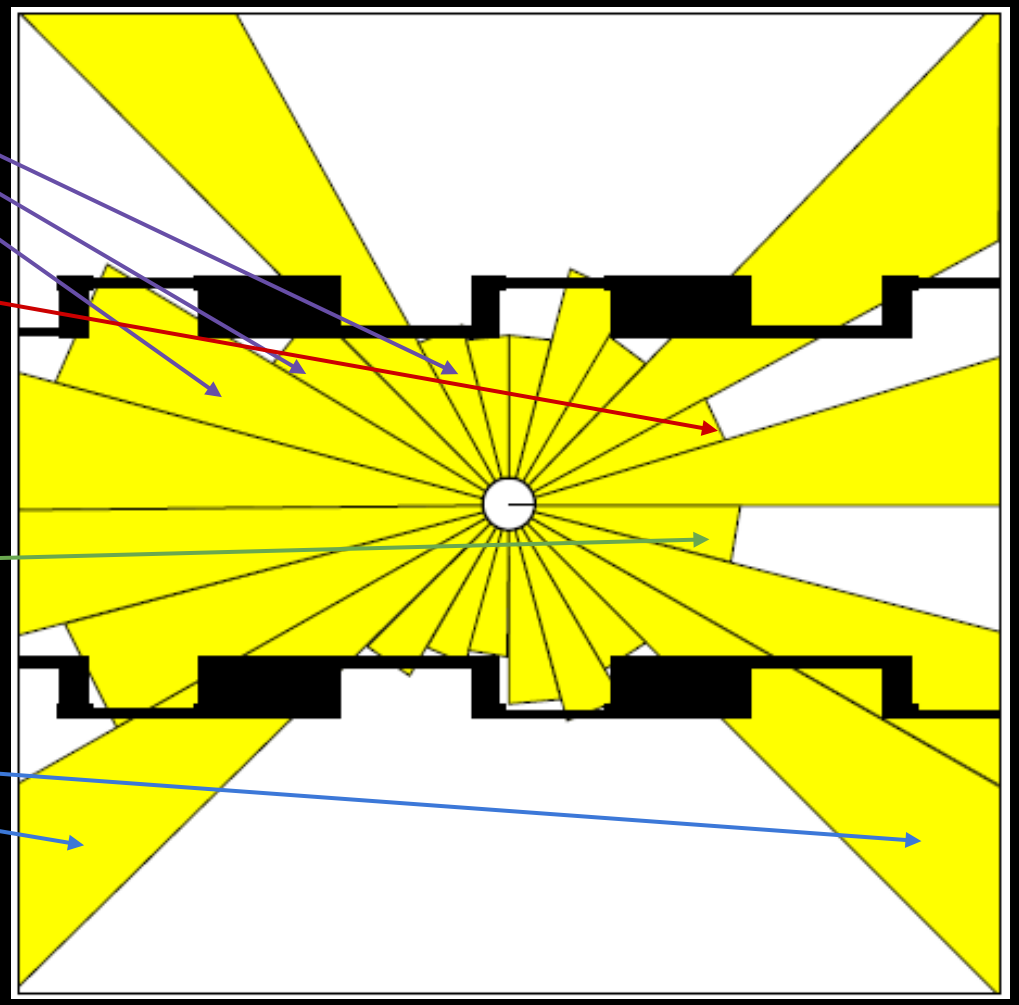
   Beams reflected by obstacles

2. Unexpected Objects

   Beams reflected by persons / caused by crosstalk

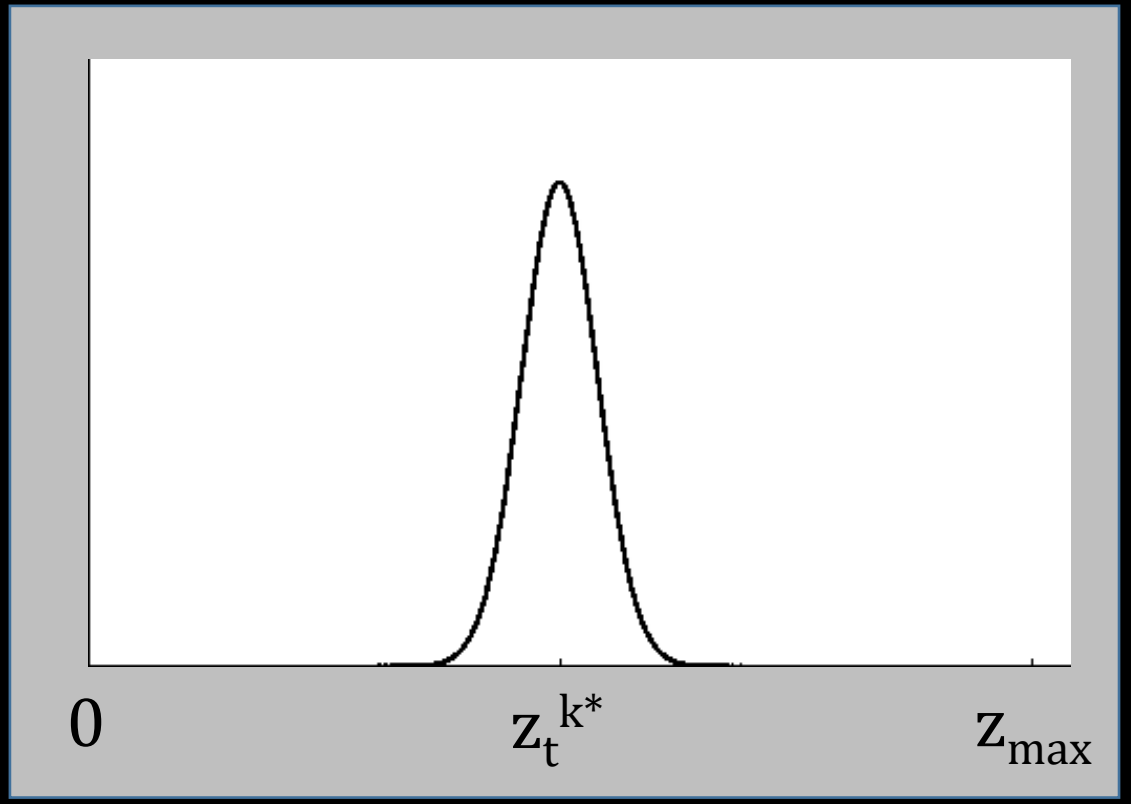
3. Failures

4. Random measurements



# 1. Correct Range Measurements

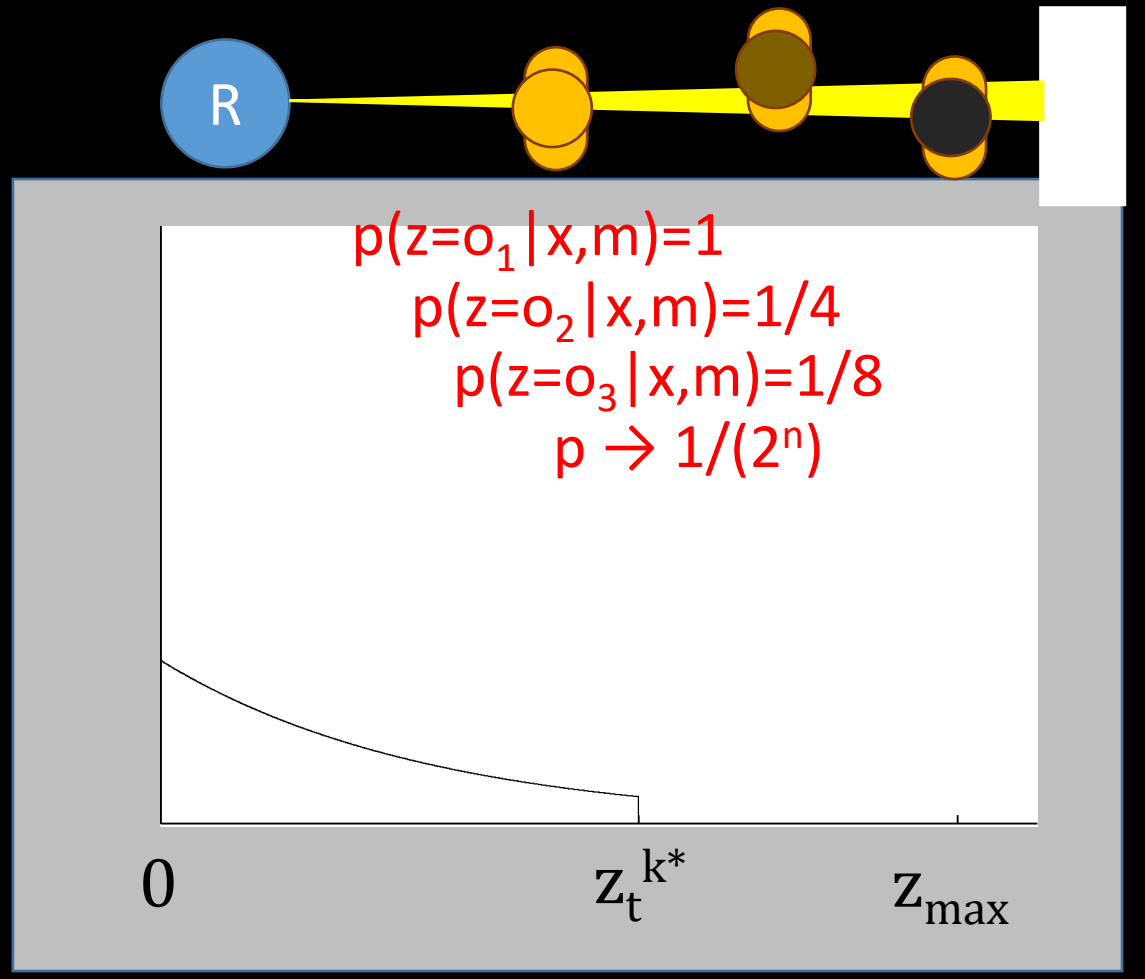
- Reading
  - $z_t^k$
- True value
  - $z_t^{k*}$
  - In a location-based map, $z_t^{k*}$ is usually estimated by *ray casting*
- Measurement noise
  - Narrow Gaussian $p_{hit}$ with mean $z_t^{k*}$ and standard deviation $\sigma_{hit}$




$$p_{hit}(z_t^k | x_t, m) = \begin{cases} \eta f(z_t^k; z_t^{k*}, \sigma_{hit}) & if\ 0 \le z_t^k \le z_{max} \\ 0 & otherwise \end{cases}$$

# 2. Unexpected Objects

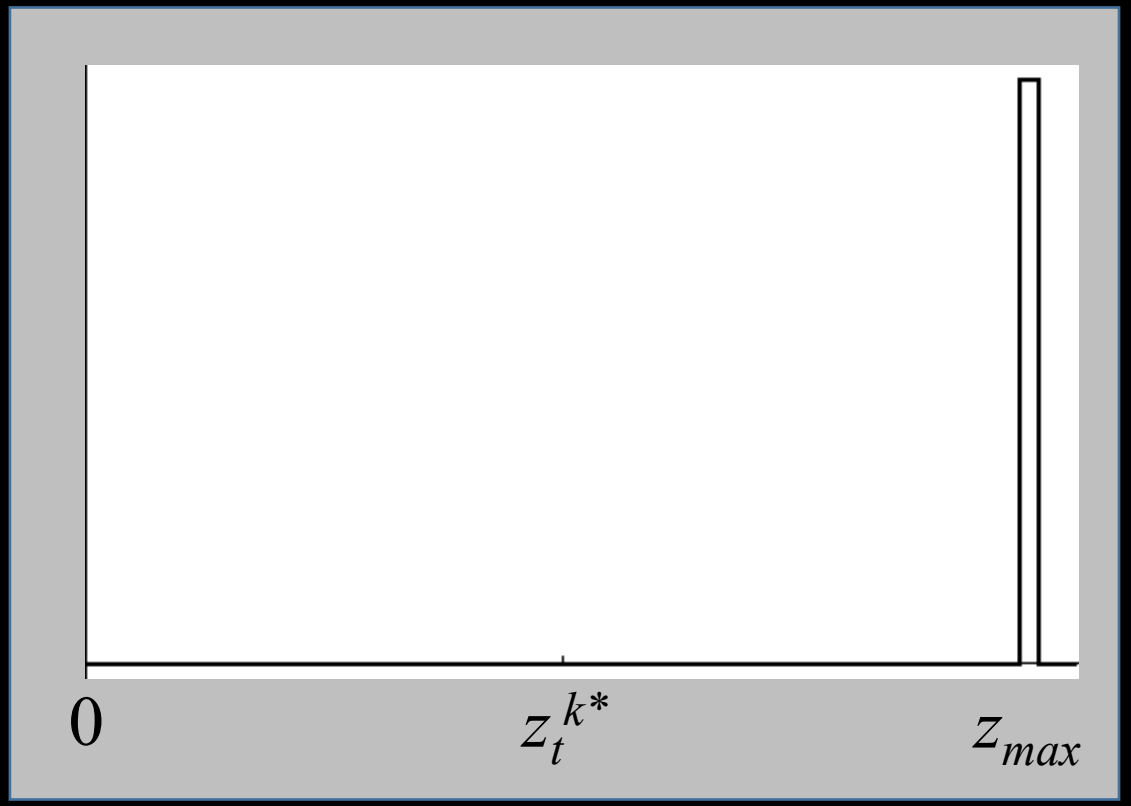
- Real world is dynamic
- Objects not contained in the map can cause shorter readings
  - treat them as part of the state vector and estimate their location
  - Or treat them as sensor noise
- The likelihood of sensing unexpected objects decreases with range
- Model as an exponential distribution $p_{short}$




$$p_{short}(z_t^k|x_t, m) = \begin{cases} \eta\, \lambda_{short}\, e^{-\lambda_{short} z_t^k} & if\ 0 \le z_t^k \le z_t^{k*} \\ 0 & otherwise \end{cases}$$

# 3. Failures

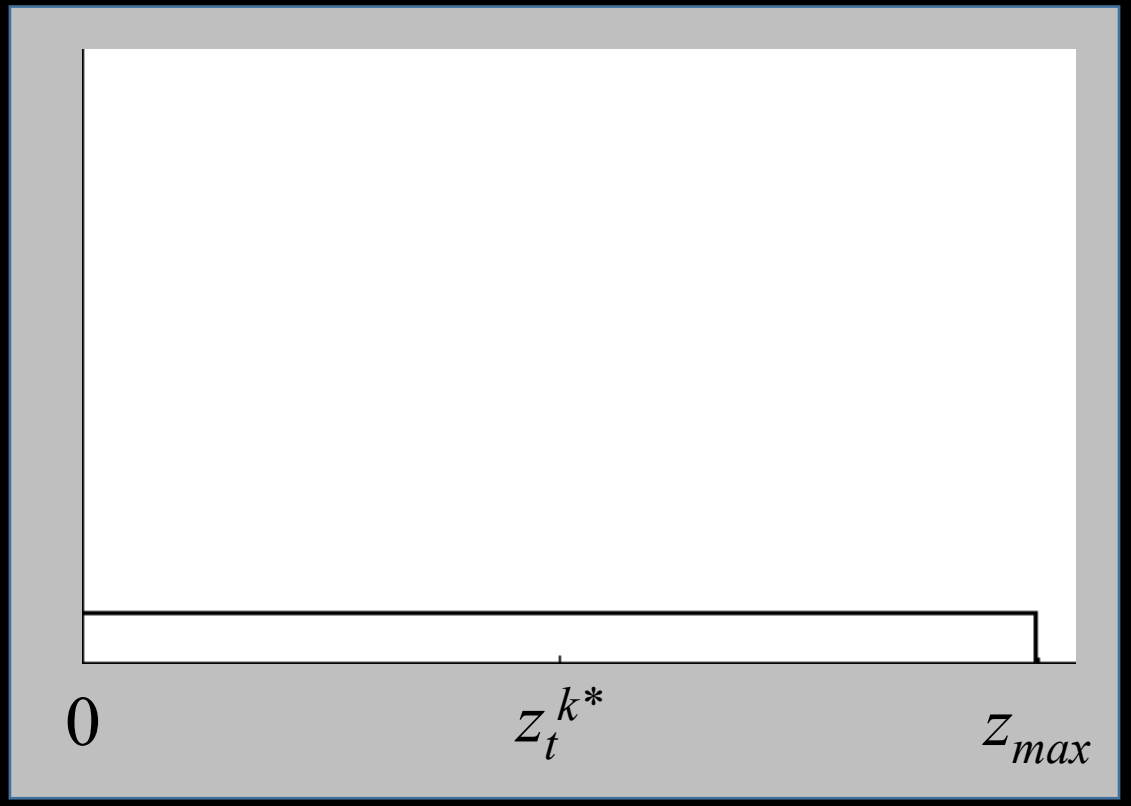
- Obstacles might be missed altogether
- The result is a max-range measurement $z_{max}$
- Model as a point-mass distribution $p_{max}$




$$p_{max}\left(z_t^k \middle| x_t, m\right) = I(z = z_{max}) = \begin{cases} 1 & if\ z = z_{max} \\ 0 & otherwise \end{cases}$$

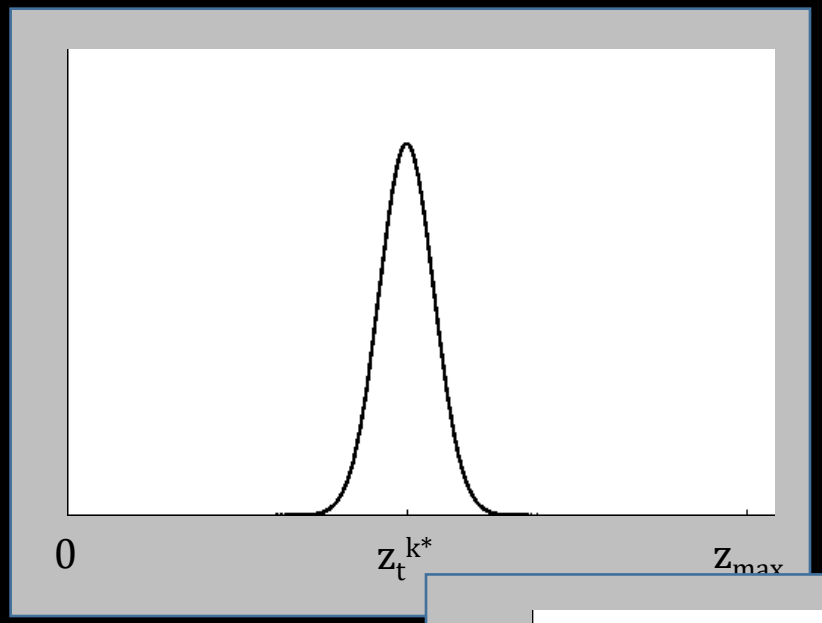
# 4. Random Measurements

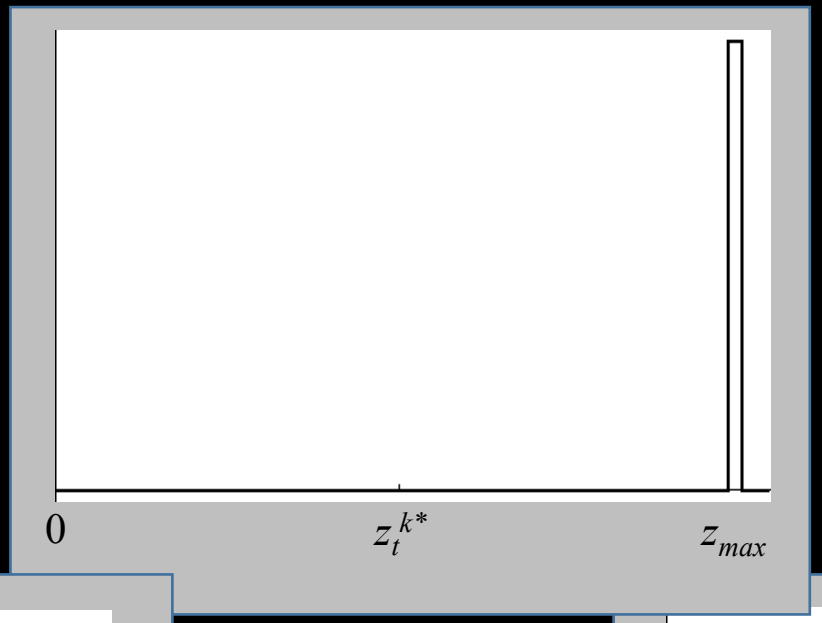
- Range finders can occasionally produce entirely unexplainable measurements
- Modelled as a uniform distribution $p_{rand}$ over the measurement range

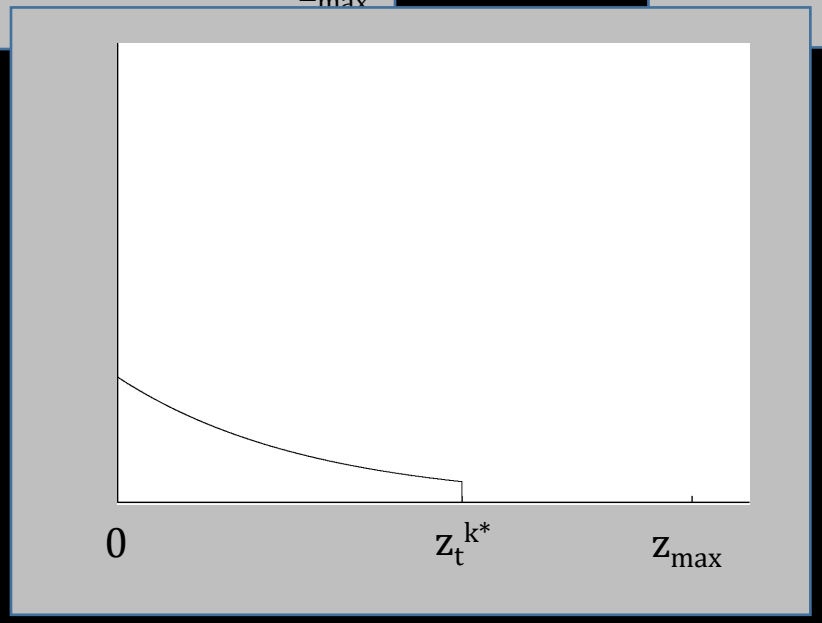


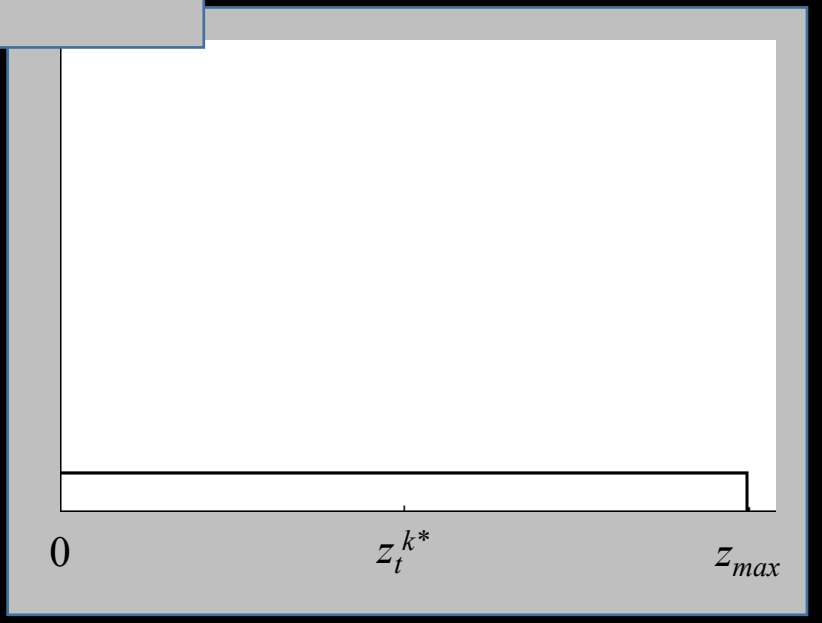

$$p_{rand}\left(z_t^k \middle| x_t, m\right) = \begin{cases} \frac{1}{z_{max}} & if\ 0 \leq z_t^k \leq z_{max} \\ 0 & otherwise \end{cases}$$

# Beam Model



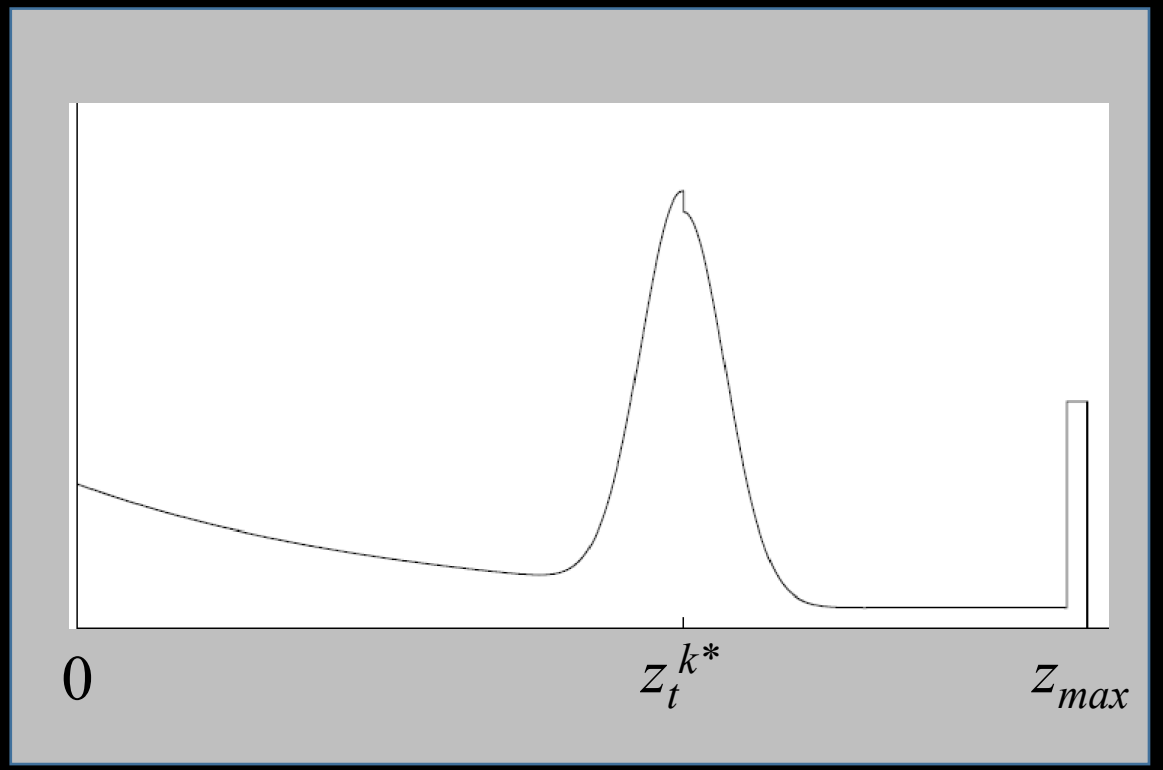



# Beam Range Model as a Mixture Density

- The four different distributions are mixed by a weighted average

$$p(z_t^k|x_t,m) = \begin{pmatrix} \alpha_{hit} \\ \alpha_{short} \\ \alpha_{max} \\ \alpha_{rand} \end{pmatrix} \cdot \begin{pmatrix} p_{hit}(z_t^k|x_t,m) \\ p_{short}(z_t^k|x_t,m) \\ p_{max}(z_t^k|x_t,m) \\ p_{rant}(z_t^k|x_t,m) \end{pmatrix}$$

- $\alpha_{hit} + \alpha_{short} + \alpha_{max} + \alpha_{rand} = 1$



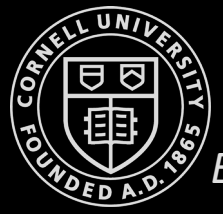
# Algorithm for Beam Model

1. **Algorithm beam_range_finder_model**$(z_t, x_t, m)$:
2. $q = 1$
3. for $k = 1$ *to* $K$ *do*
4. compute $z_t^{k*}$ for $z_t^k$ using ray casting
5. $p = \alpha_{hit}.\, p_{hit}\left(z_t^k \middle| x_t, m\right) + \alpha_{short}.\, p_{short}\left(z_t^k \middle| x_t, m\right)$
6. $+ \alpha_{max}.\, p_{max}\left(z_t^k \middle| x_t, m\right) + \alpha_{rand}.\, p_{rand}\left(z_t^k \middle| x_t, m\right)$
7. $q = q.p$
8. return $q$

# Parameters of Beam Range Model

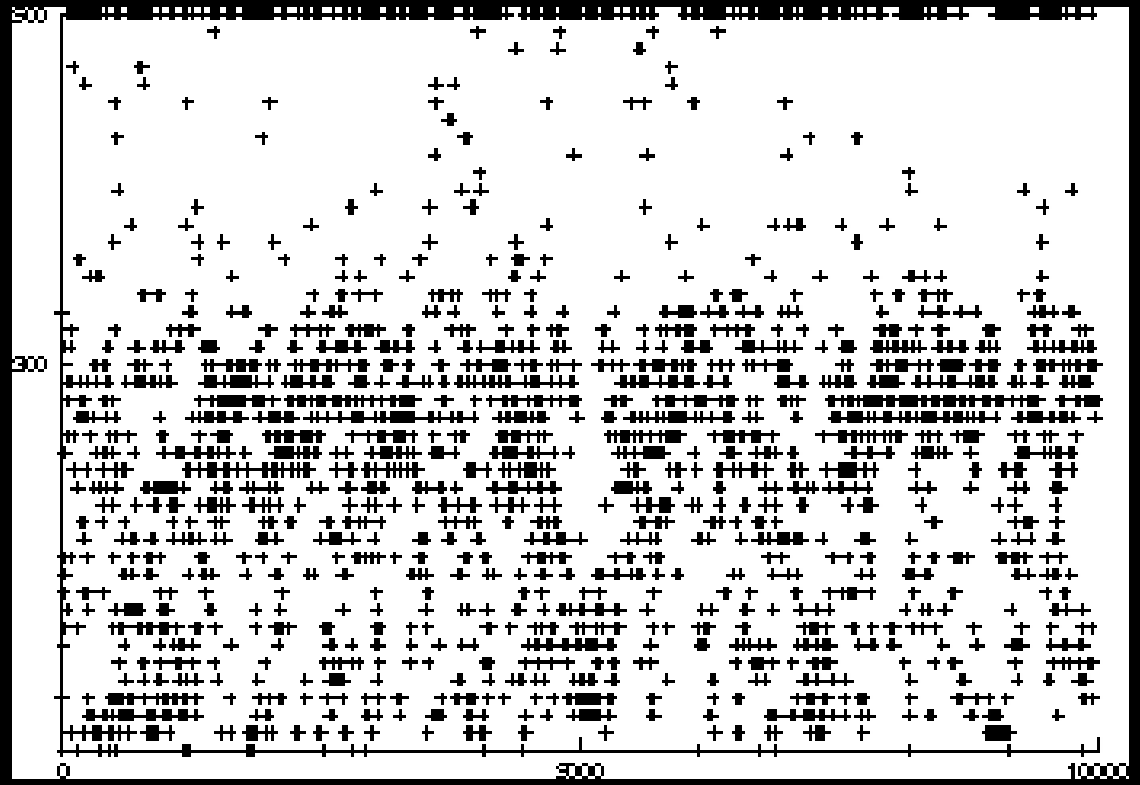
- Intrinsic Parameters $\Theta$ of the beam range model
  - $\alpha_{hit}$ , $\alpha_{short}$ , $\alpha_{max}$ , $\alpha_{rand}$ , $\lambda_{short}$
  - Affect the likelihood of any sensor measurement

- Estimation Methods
  - Guesstimate the resulting density
  - Learn parameters using a Maximum Likelihood Estimator
  - Hill Climbing, Gradient descent, Genetic algorithms, etc.

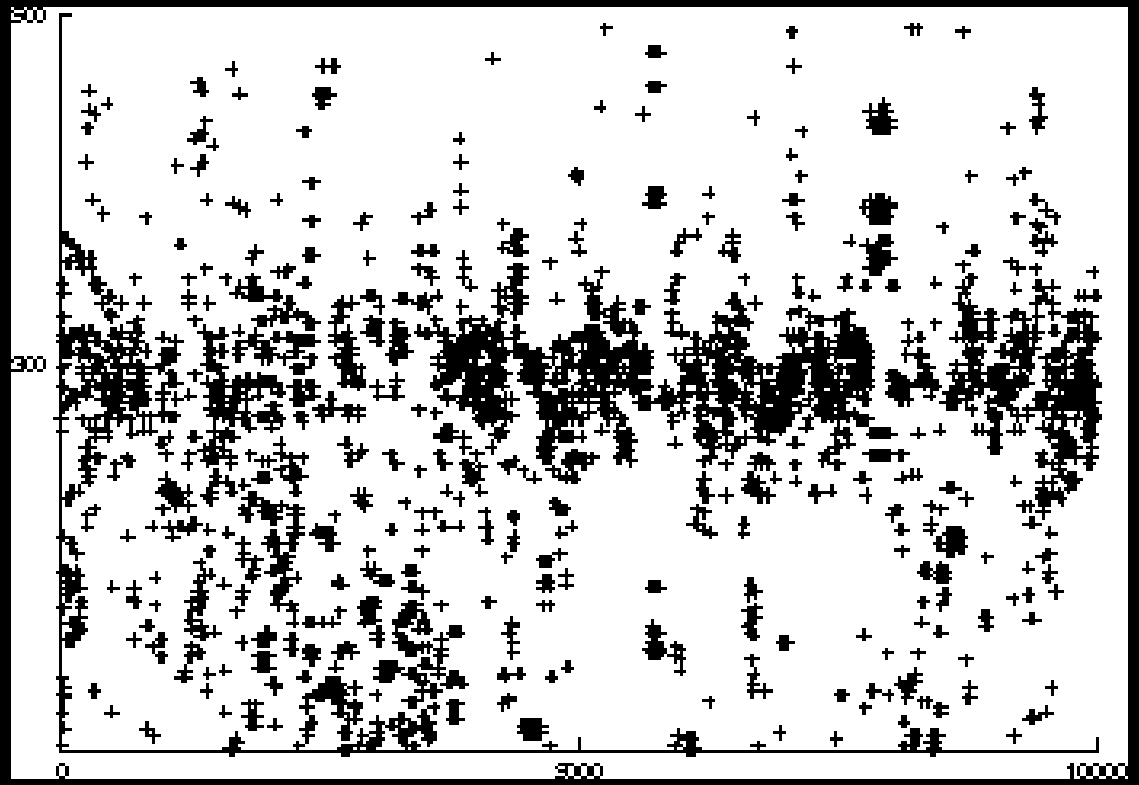


# Raw Sensor Data

Sonar senor data



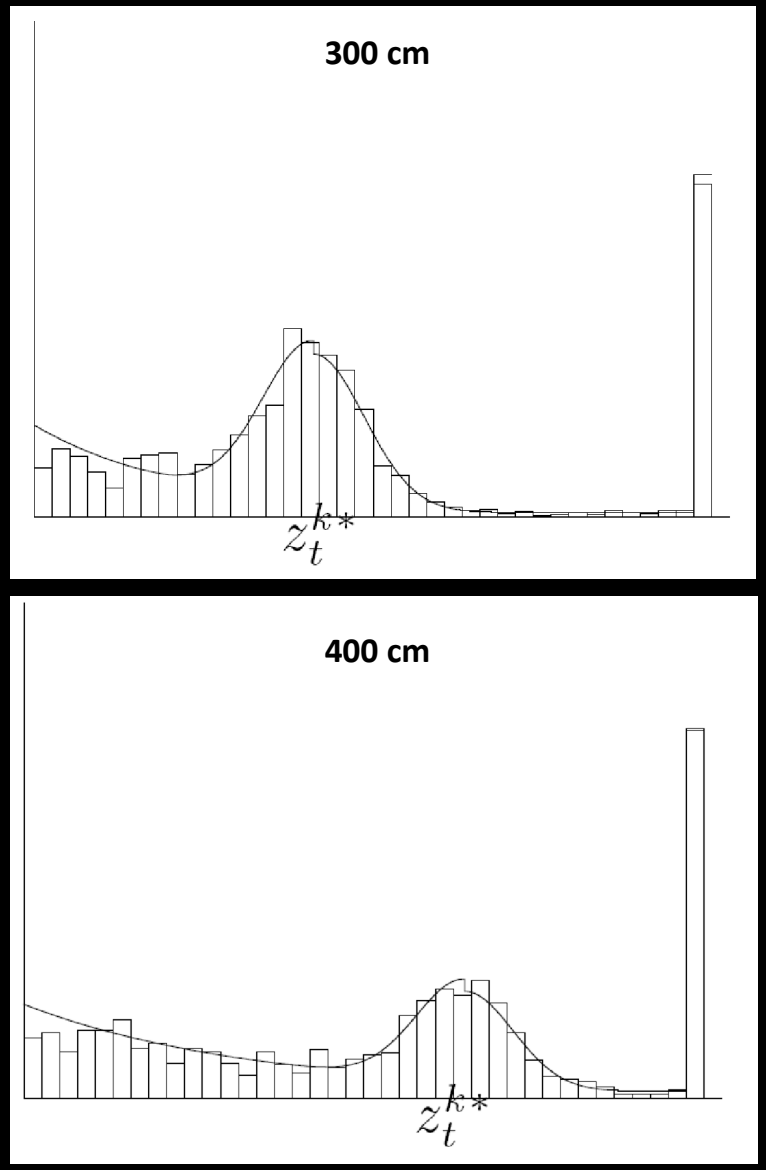

Laser range sensor



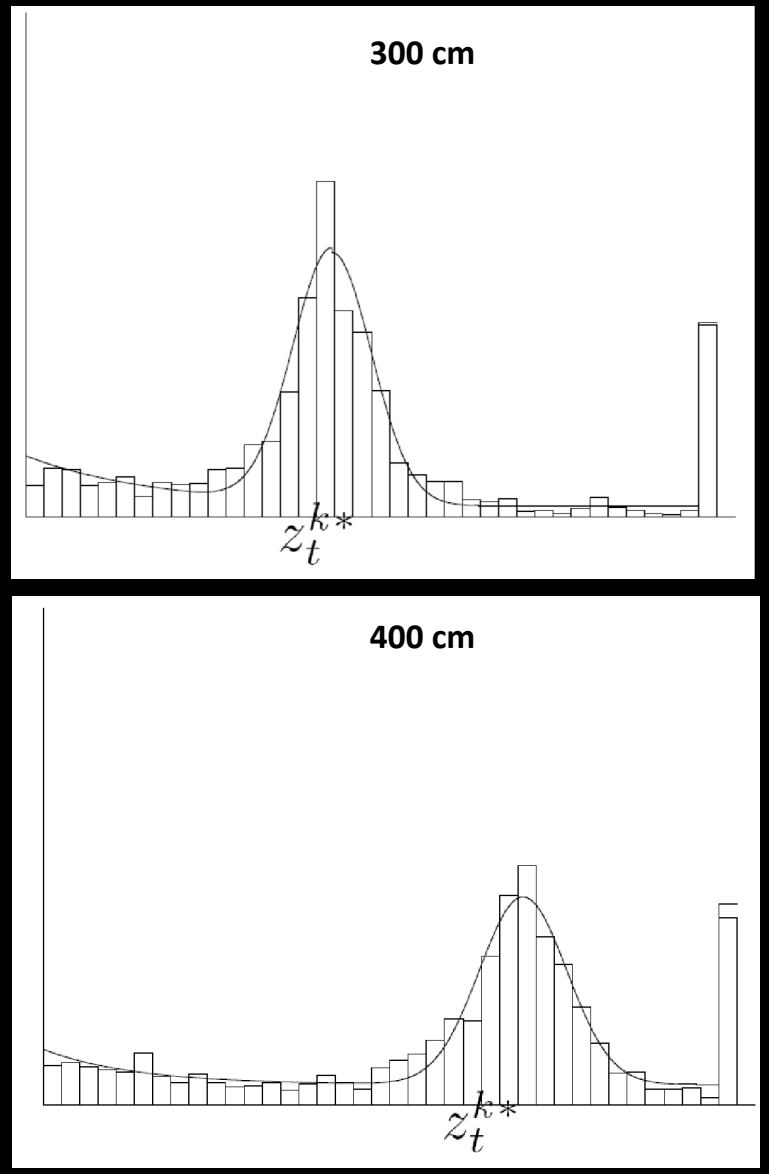

(True range is 300 cm and maximum range is 500 cm)

# Approximation Results of Beam Model (with MLE)

Sonar Data

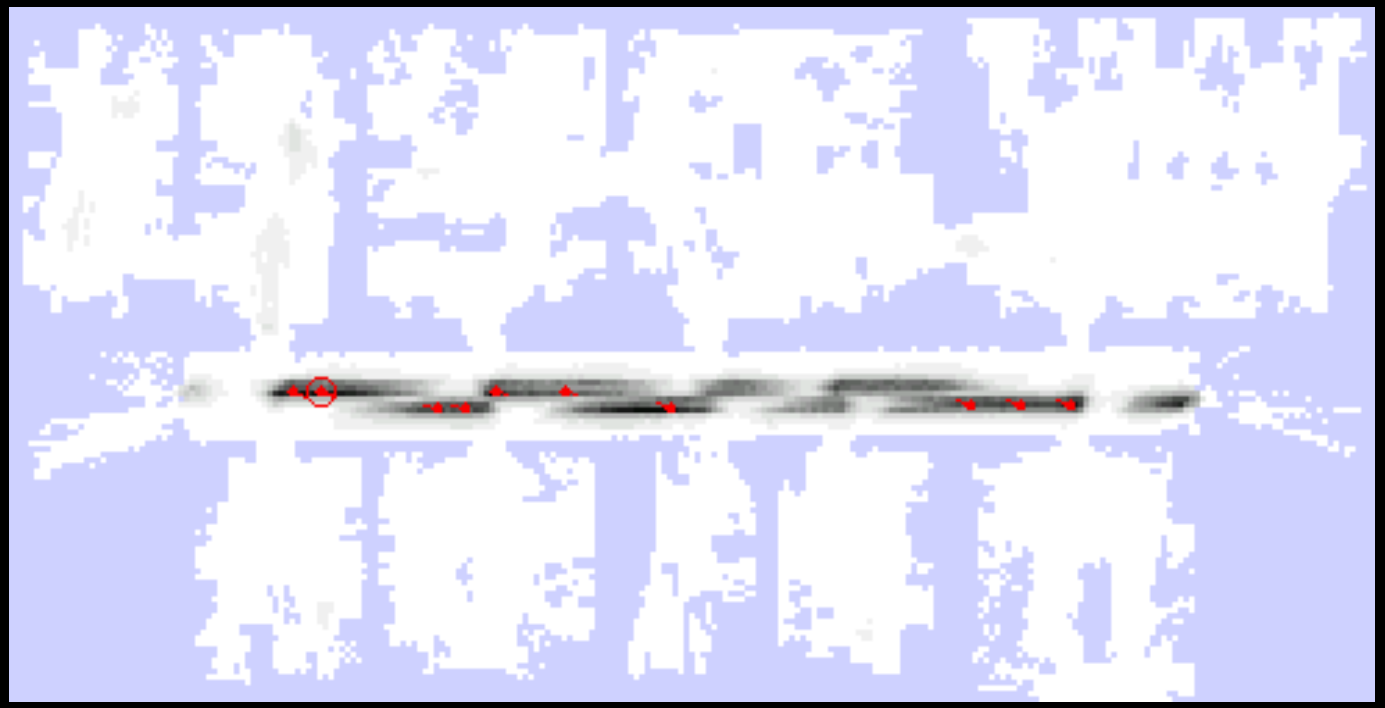



Laser Data



# Beam Model in Action

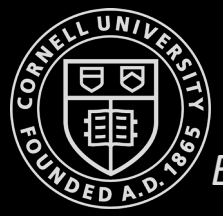
Laser scan projected into a partial map $m$.



Likelihood $p(z_t|x_t, m)$ for all positions $x_t$ projected into the map. The darker a position, the larger $p(z_t|x_t, m)$

# Summary of Beam Model

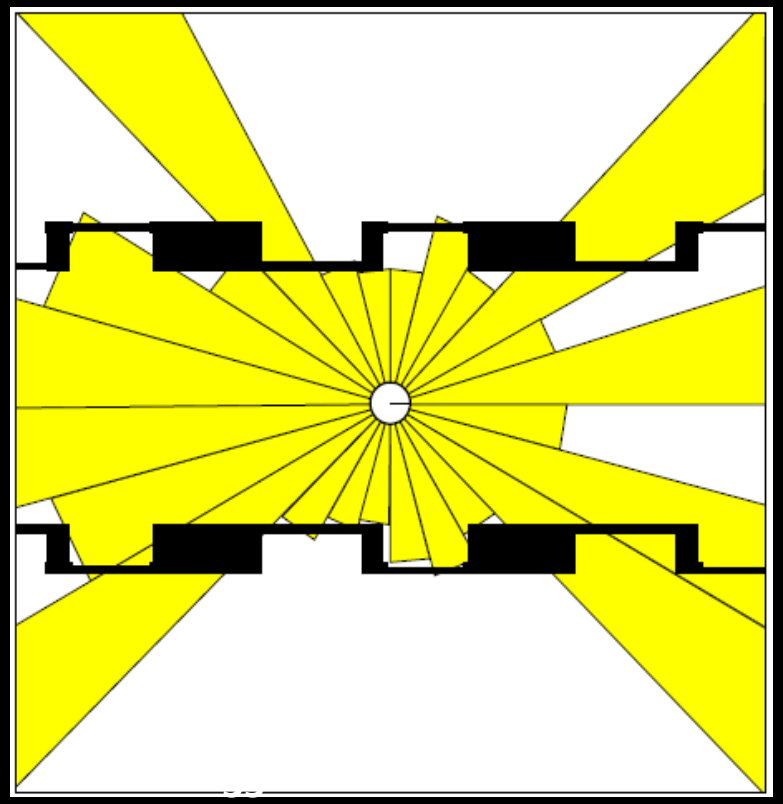
- Overconfident
  - Assumes independence between individual measurements
- Models physical causes for measurements
- Implementation involves learning parameters based on real data
- Limitations
  - Often, different models are needed for different hit angles
  - Raytracing is computationally expensive
    - But can be pre-processed
  - Not smooth for small obstacles, at edges, or in cluttered environments



# Likelihood Fields

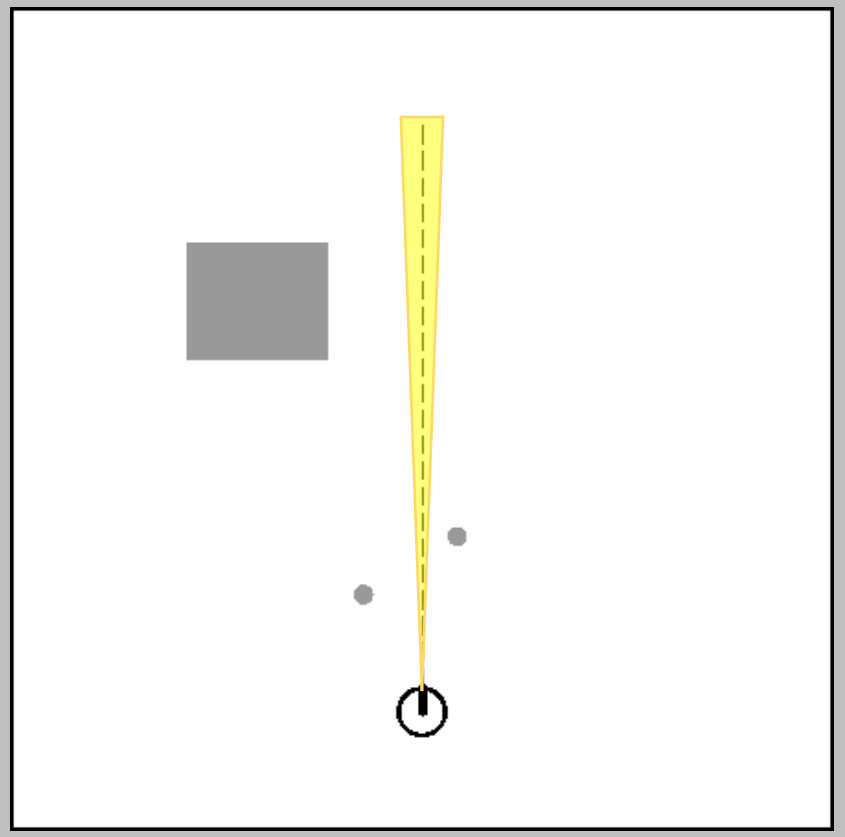
# Likelihood Fields of Range Finders

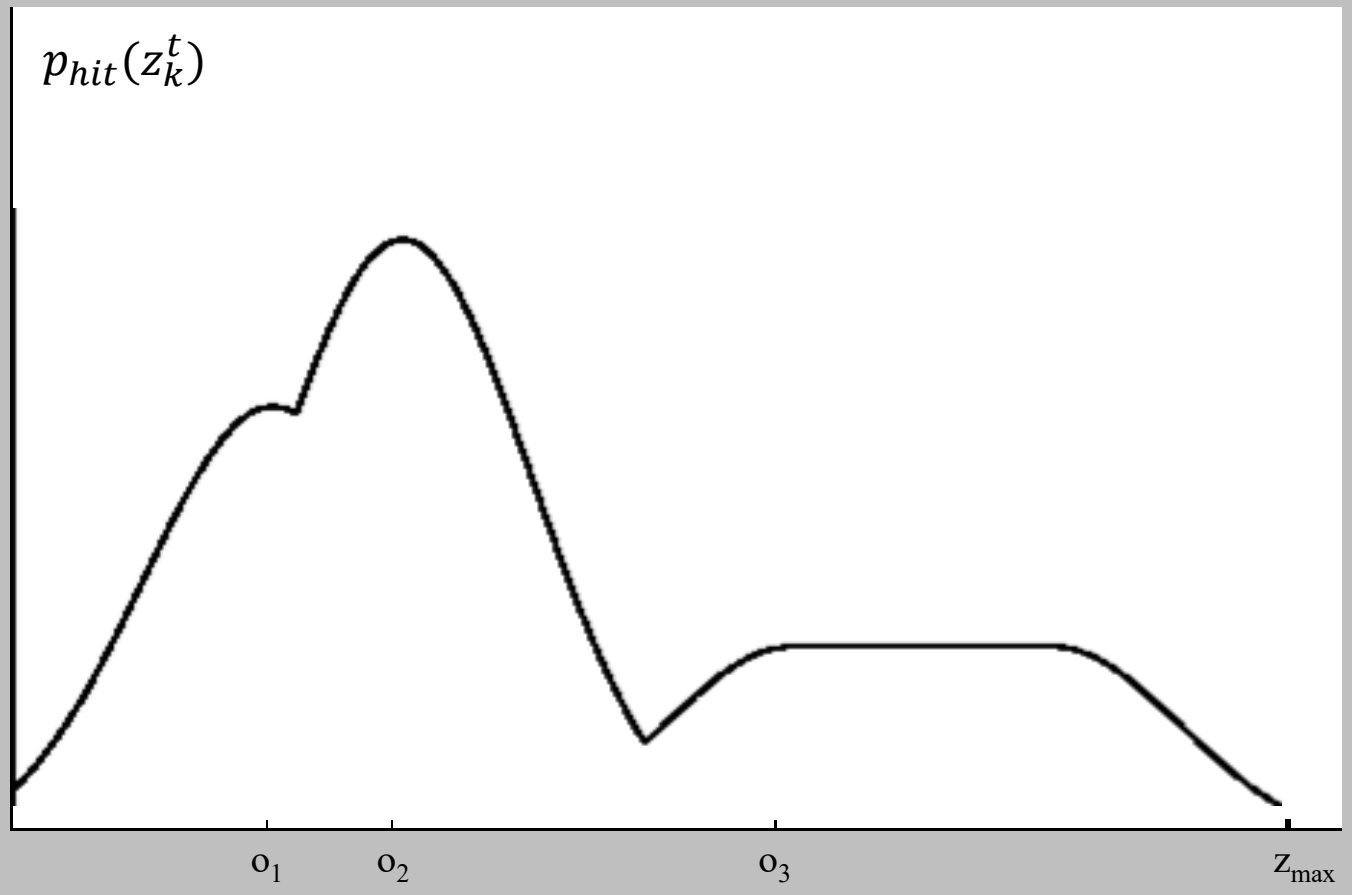
- Instead of following along the beam, just check the end point
- Project sensor scan $z_t$ into the map and compute end point
- Probability function is a mixture of
  - a Gaussian distribution with mean at the distance to closest obstacle
  - a uniform distribution for random measurements
  - a point mass distribution for max range measurements



# Measurement Noise

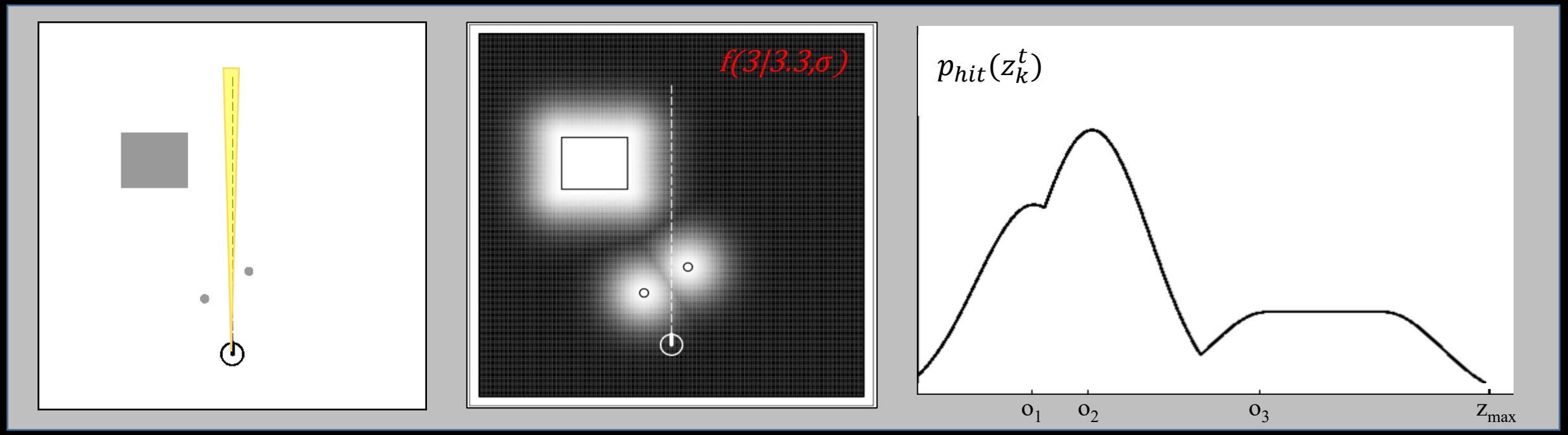
- Modelled using Gaussians





# Measurement Noise

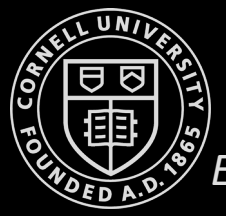
- Modelled using Gaussians
- In xy space, this involves finding the nearest obstacle in the map
- The probability of a sensor measurement is given by a Gaussian that depends on the euclidean distance between measurement coordinates and nearest object in the map $m$



# Likelihood Fields for Range Finders

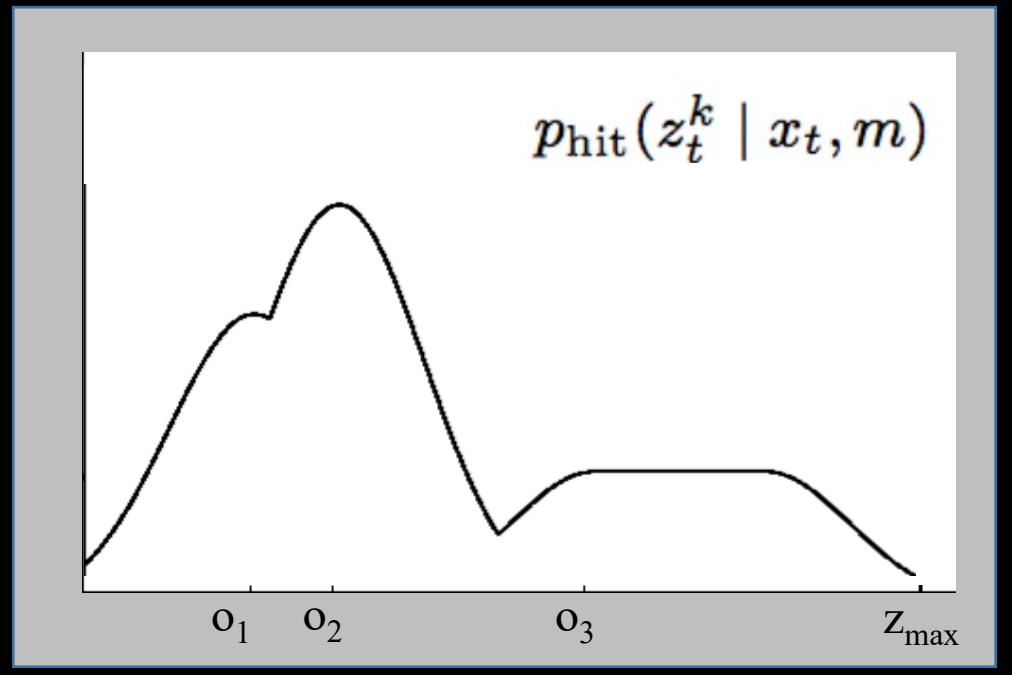
- Robot pose in world frame: $x_t = (x, y, \theta)^T$
- Sensor pose in the robot frame: $(x_{k,sens}, y_{k,sens}, \theta_{k,sens})$
- $z^k_t$ hit/"end" points in the global coordinate frame

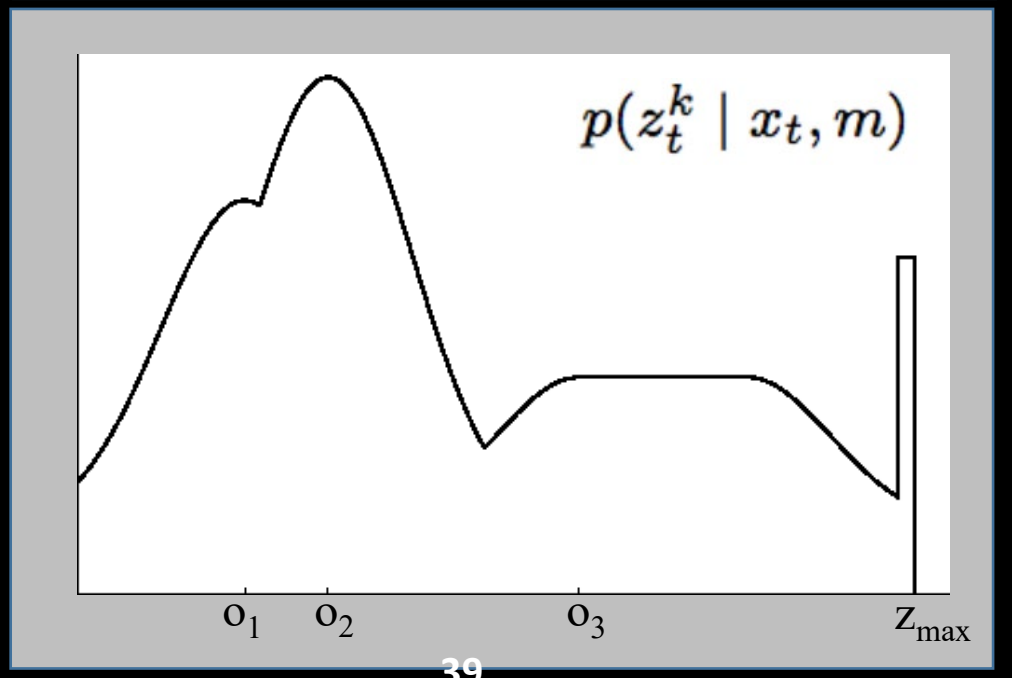
$$\begin{pmatrix} x_{z_t^k} \\ y_{z_t^k} \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x_{z^{k,sens}} \\ y_{z^{k,sens}} \end{pmatrix} + z_t^k \begin{pmatrix} \cos(\theta + \theta^{k,sens}) \\ \sin(\theta + \theta^{k,sens}) \end{pmatrix}$$



# Likelihood Fields for Range Finders

- Assume independence between individual measurements
- Three types of sources of noise and uncertainty
  - Measurement noise
  - Failures
    - Max range readings are modelled by a point-mass distribution
  - Unexplained random measurements
    - Uniform distribution



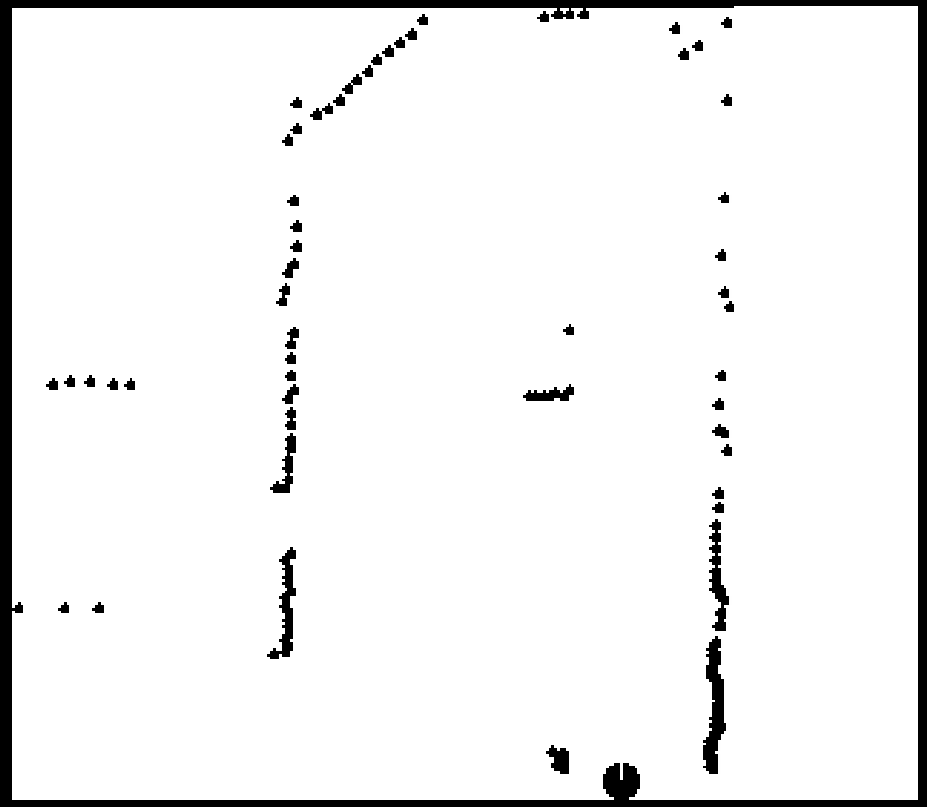



# Algorithm for Likelihood Fields

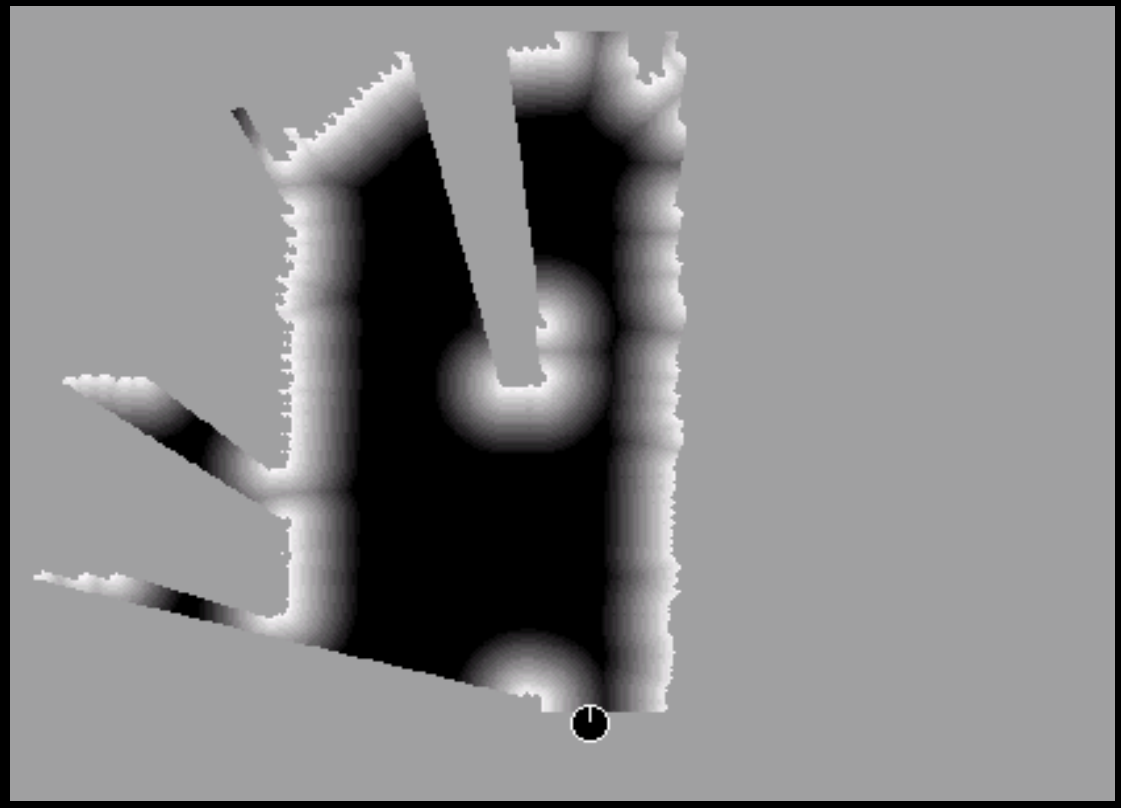
1. **Algorithm likelihood_field_range_finder_model**$(z_t, x_t, m)$:
2. q = 1
3. for $k = 1\ to\ K\ do$
4. $x_{z_k^t} = x + x_{z_k,sens}\cos(\theta) - y_{z_k,sens}\sin(\theta) + z_k^t\cos(\theta + \theta_{k,sens})$
5. $y_{z_k^t} = y + y_{z_k,sens}\cos(\theta) + x_{z_k,sens}\sin(\theta) + z_k^t\sin(\theta + \theta_{k,sens})$
6. $dist = \min_{x',y'}\left\{\sqrt{(x_{z_k^t} - x')^2 + (y_{z_k^t} - y')^2} \mid \langle x', y'\rangle \text{ occupied in } m\right\}$
7. $q = q.(z_{hit}.f(dist; 0, \sigma_{hit}) + \frac{z_{rand}}{z_{max}})$
8. return $q$

# Likelihood Field from Sensor Data

Sensor data projected into map

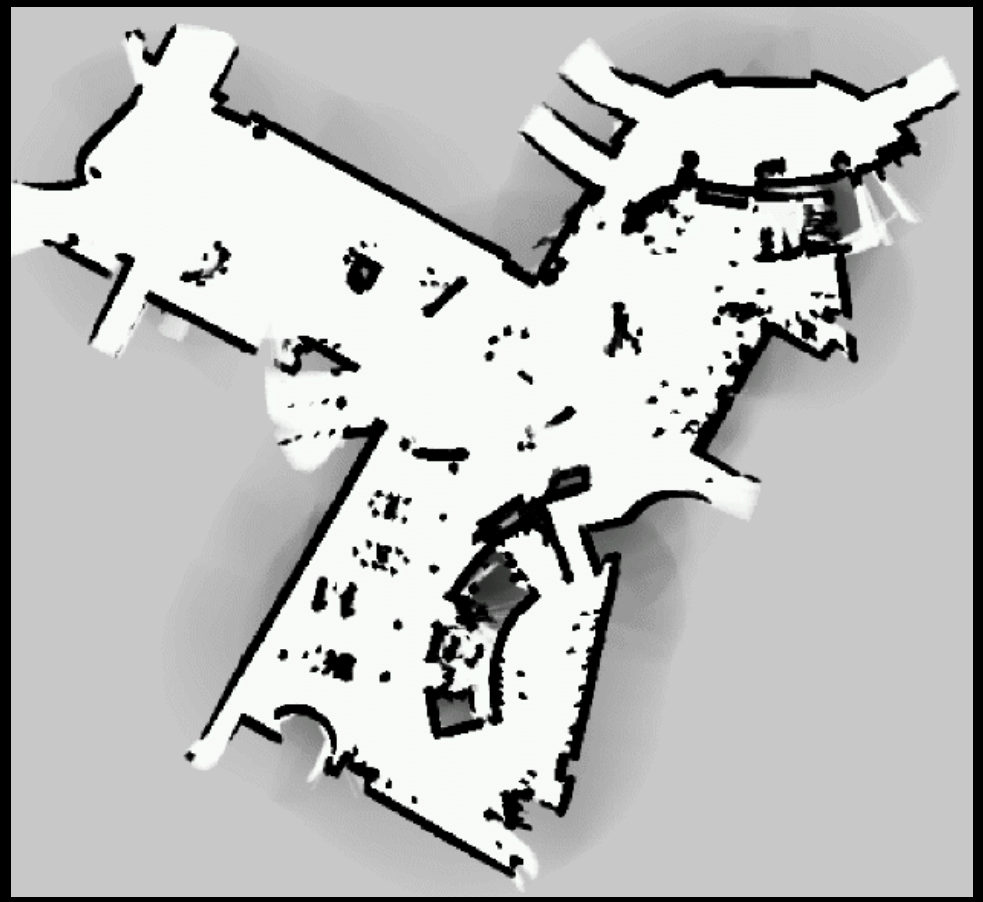


Corresponding likelihood function

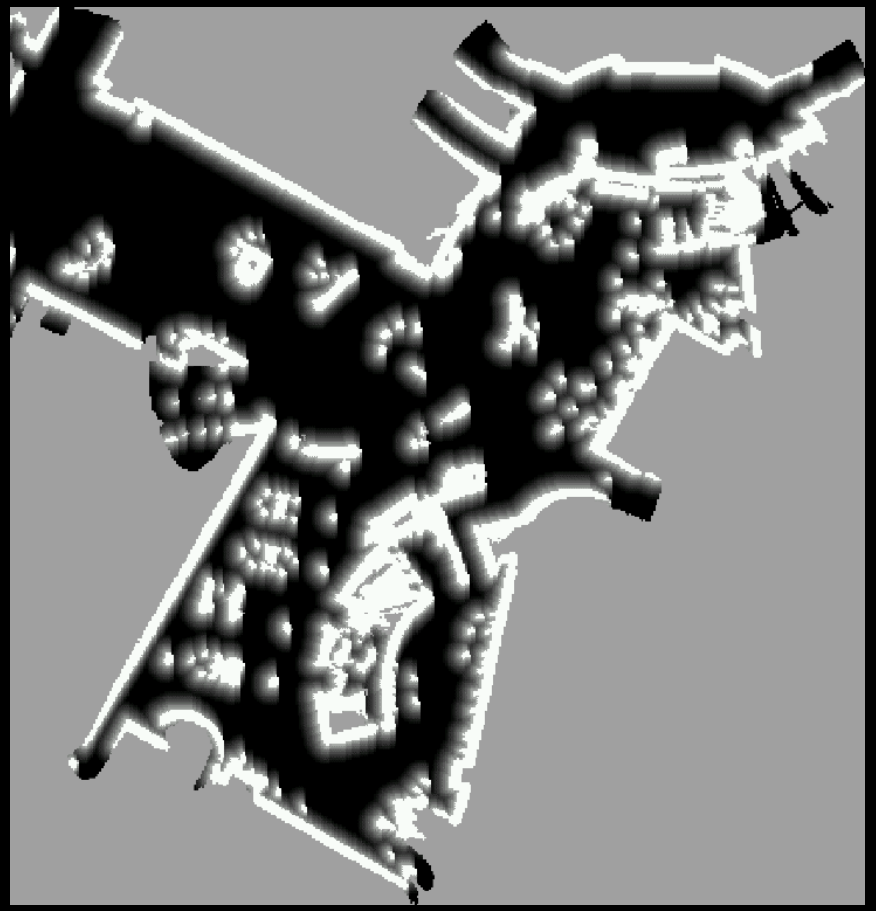


# San Jose Tech Museum

Occupancy grid map

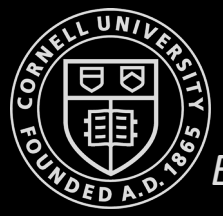


Likelihood field



# Summary of Likelihood Fields

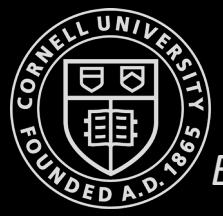
- Advantages
  - Highly efficient (computation in 2D instead of 3D)
  - Smooth w.r.t. to small changes in robot position
- Limitations
  - Does not model people and other dynamics that might cause short readings
  - Ignores physical properties of beams



# Feature Based Models

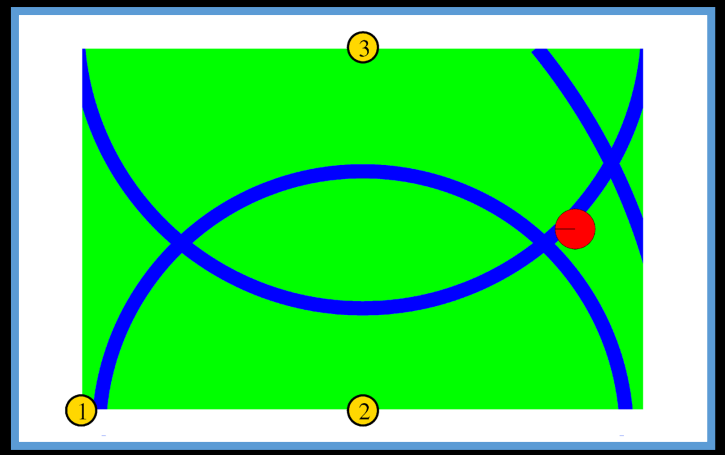
# Feature Based Models

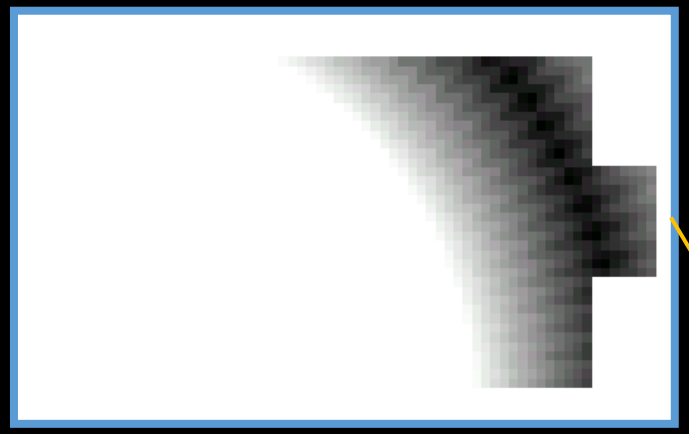
- Extract features from dense raw measurements
  - For range sensors: lines and corners
  - Often from cameras (edges, corners, distinct patterns, etc.)
- Feature extraction methods
- Features correspond to distinct physical objects in the real world and are often referred to as *landmarks*
  - Sensors output the range and/or bearing of the landmark w.r.t to the robot frame
  - Trilateration
  - Inference in the feature space can be more efficient

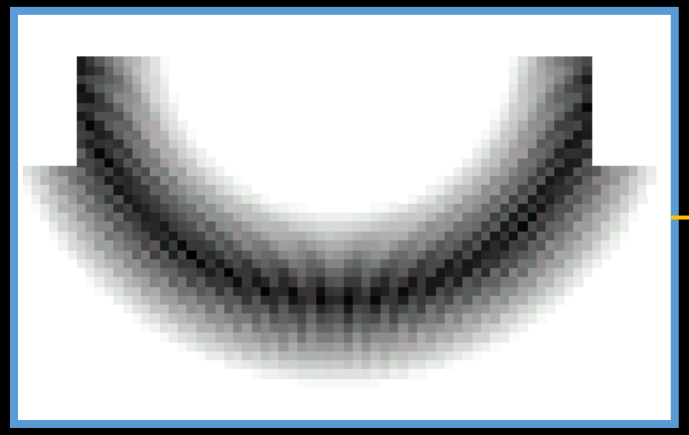


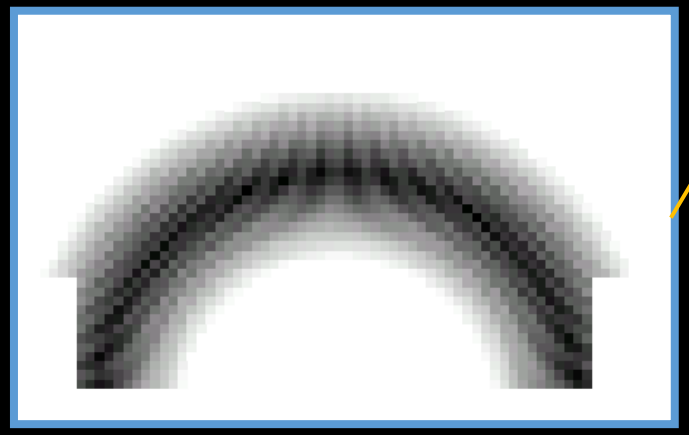
# Trilateration using Range Measurements

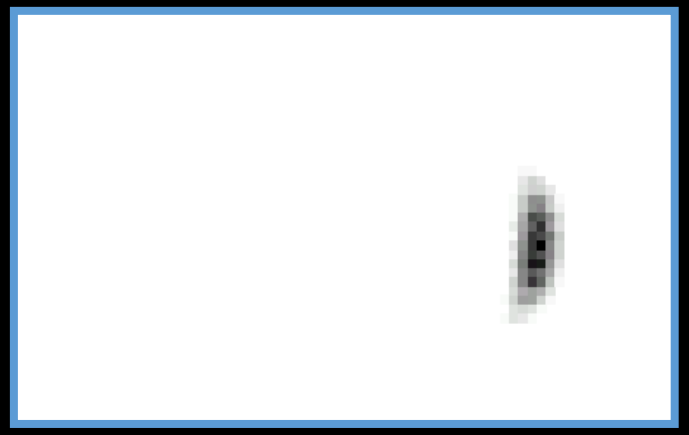

# Trilateration using Range Measurements



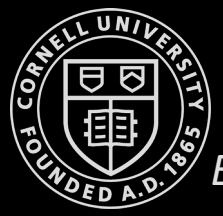






# Summary of Sensor Models

- Robustness comes from explicitly modeling sensor uncertainty
- Measurement likelihood is given by “probabilistically comparing” the actual with the expected measurement
- Often, good models can be found by
    1. Determining a parametric model of noise free measurements
    2. Analyzing the sources of noise
    3. Adding adequate noise to parameters (mixed density functions)
    4. Learning (and verify) parameters by fitting model to data
- It is extremely important to be aware of the underlying assumptions!



# Reference


1. Thrun, Sebastian, Wolfram Burgard, and Dieter Fox. Probabilistic robotics. MIT press, 2005.
2. http://ais.informatik.uni-freiburg.de/teaching/ss10/robotics/slides/07-sensor-models.pdf
3. http://www.cs.cmu.edu/~16831-f14/notes/F12/16831_lecture03_mtaylormshomin.pdf
4. Gaussian Distribution: https://www.asc.ohio-state.edu/gan.1/teaching/spring04/Chapter3.pdf
5. Gaussian Distribution: http://www2.stat.duke.edu/~rcs46/modern_bayes17/lecturesModernBayes17/lecture-3/03-normal-distribution.pdf