Introduction to Autonomous Electric Vehicles

Lecture 4

- Recap
- Localization
- Mapping

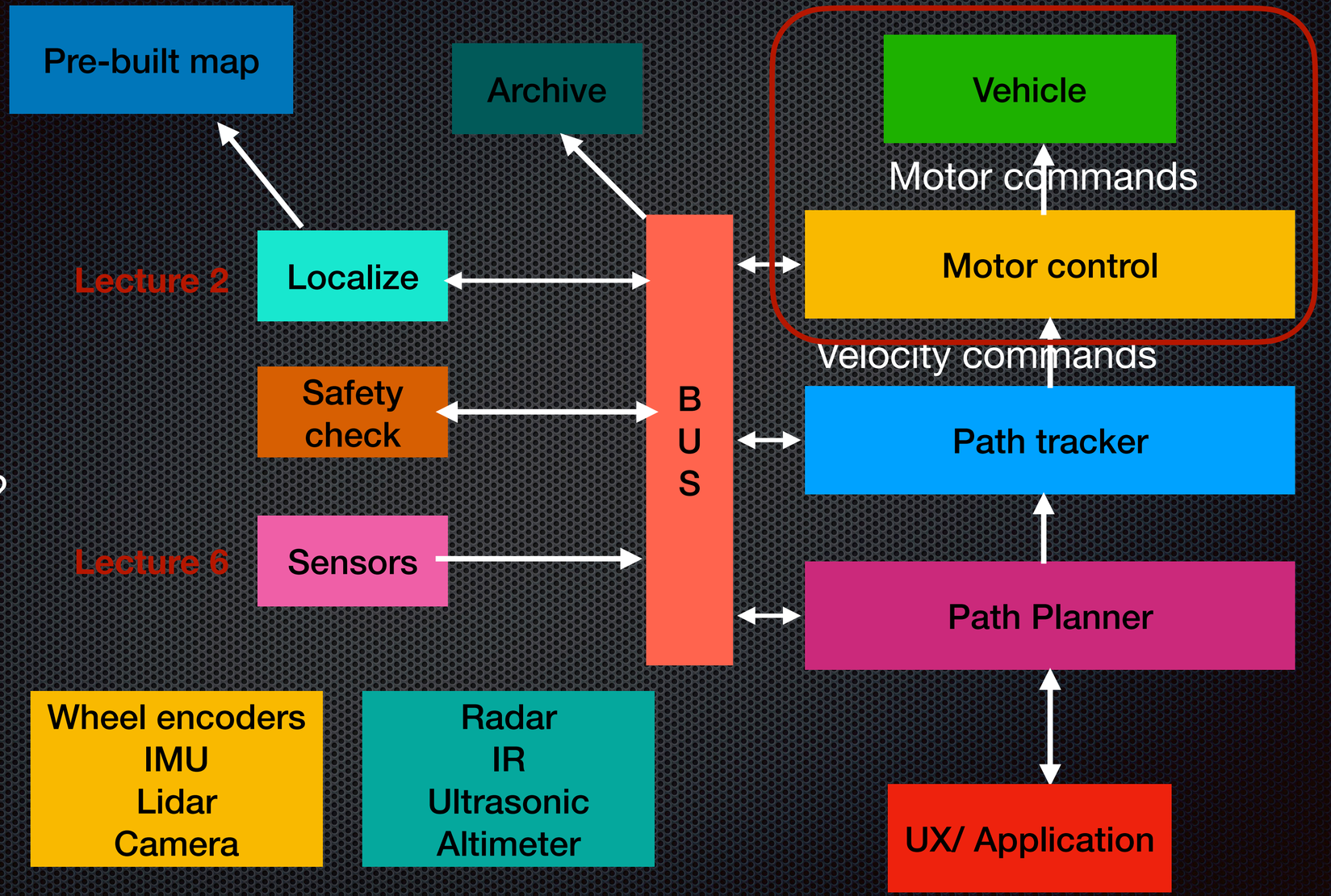
EEE department, BITS-Goa

31-March-2021

Lecture plan

- Recap
  - Autonomy and control architecture
  - Need for Closed-loop feedback
  - Sensors
  - Wheel encoders: drift
  - Lidars: Noise
- Localization
    - What is localization
    - How to localize using Lidars
    - Kalman filters: using multiple sensors to localize
- Mapping
    - How to map the environment
    - Example for mapping

Autonomy and Control: Pub-Sub architecture



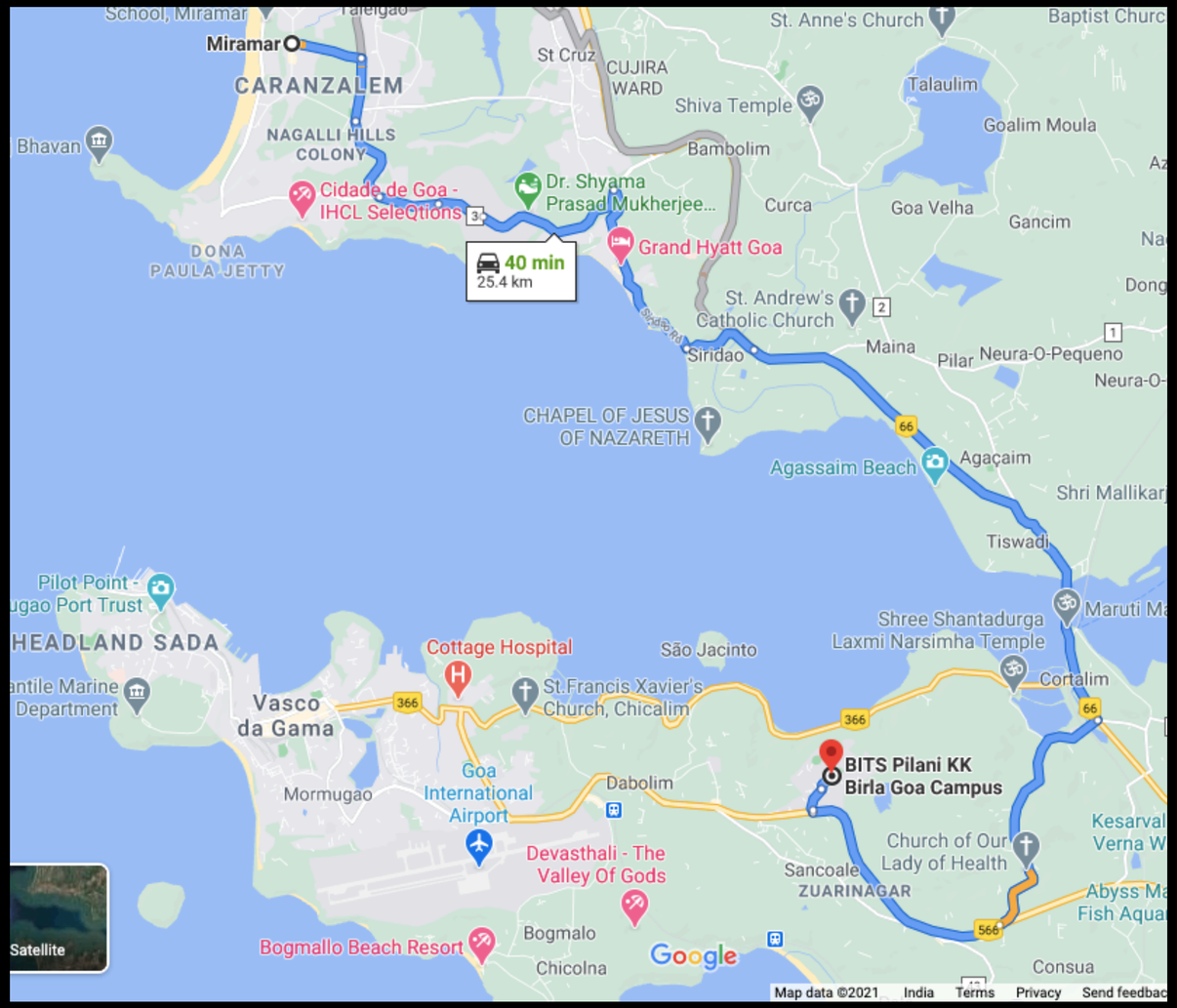
Iterative procedure

1. Where am I?
2. Where do I go next?
3. What should I do?

Key assumption: Operation in known environments

Maps and planning

Global planning: Go from campus to Miramar



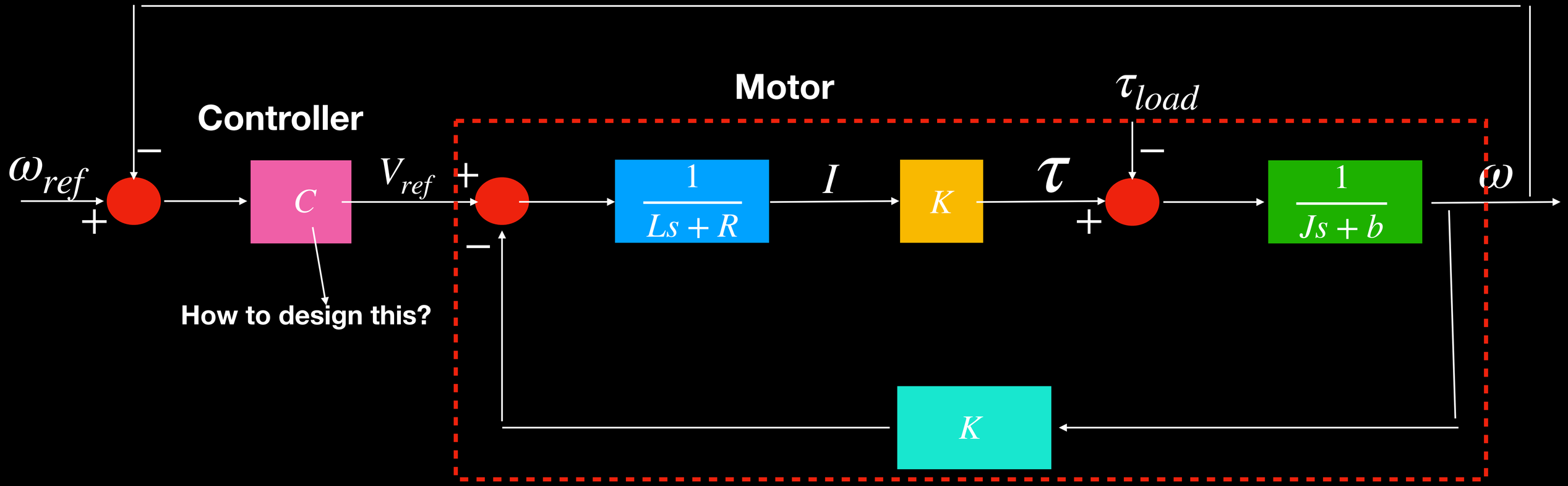
Useful for High-level estimates

Locally safe planning

Vehicle action in immediate time horizon

- More annotations/ Semantically richer
- Suited for quick Replanning

Motor control: Magic of closed-loop feedback



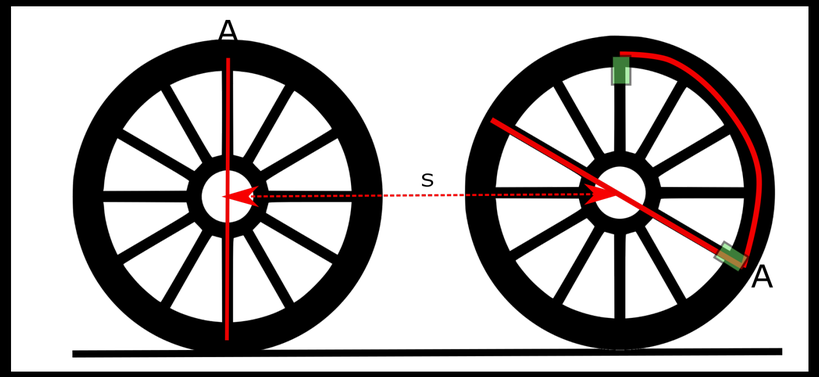
Desired properties

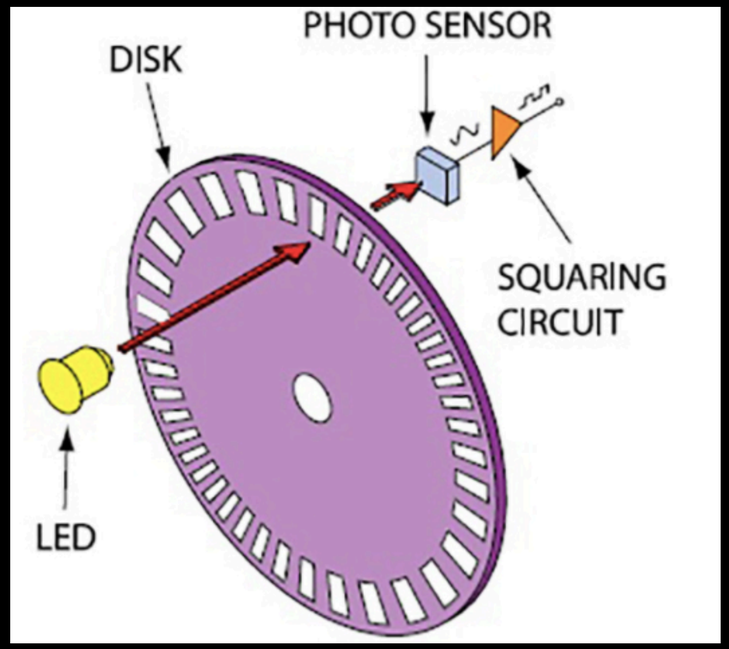
1. Need to track reference
2. Should not be dependent on model parameters
3. Provide stable performance (BIBO)

Controllers

1. Bang-Bang
2. Proportional (P)
3. Proportional-Integral (PI)

Wheel encoders





How fast left/ right wheel is rotating?

ω_l or $\omega_r = 2\pi r \frac{n}{N}$

From ω_r, ω_l get vehicle v, ω

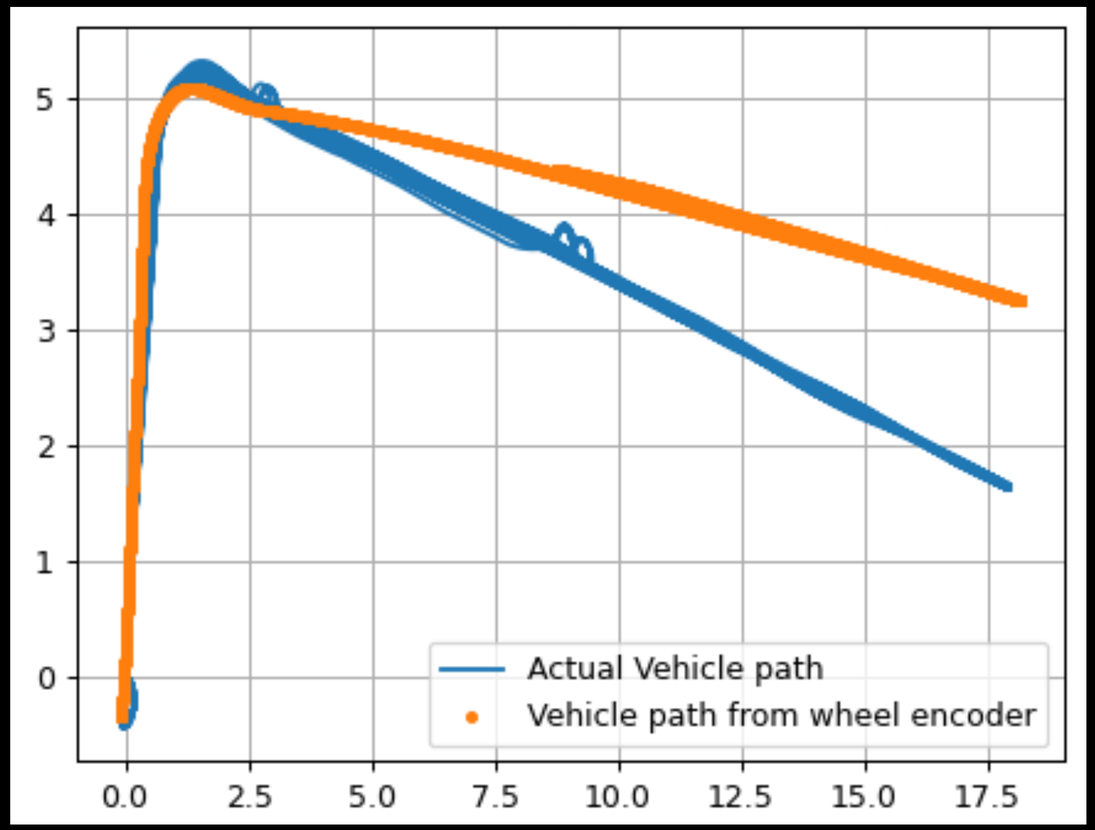
Incremental change in robot pose

$$\Delta x = v\cos\theta\Delta t$$

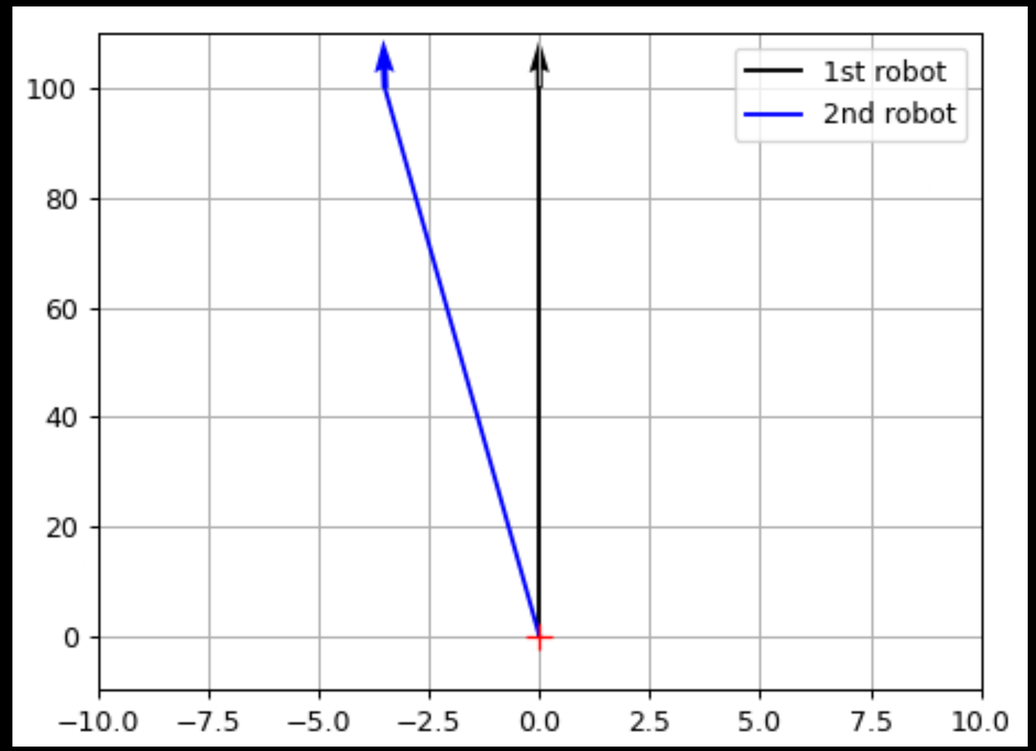
$$\Delta y = v\sin\theta\Delta t$$

$$\Delta\theta = \omega\Delta t$$

Drift in estimates from Wheel encoders



- v, ω estimates have noise
- Integrating over long time periods/ long stretches causes drift in robot pose estimate
- Heading angle estimate of robot is very sensitive



- 2 robots moving 100 meters straight
  - First robot's heading angle is correct 90 degrees
  - Second robot's heading angle has small error ~ 92 degrees
- **Robot is estimated to be 3.5m to left of where it should have been!**
  - **Error = 100*tan(2)**

Lidar

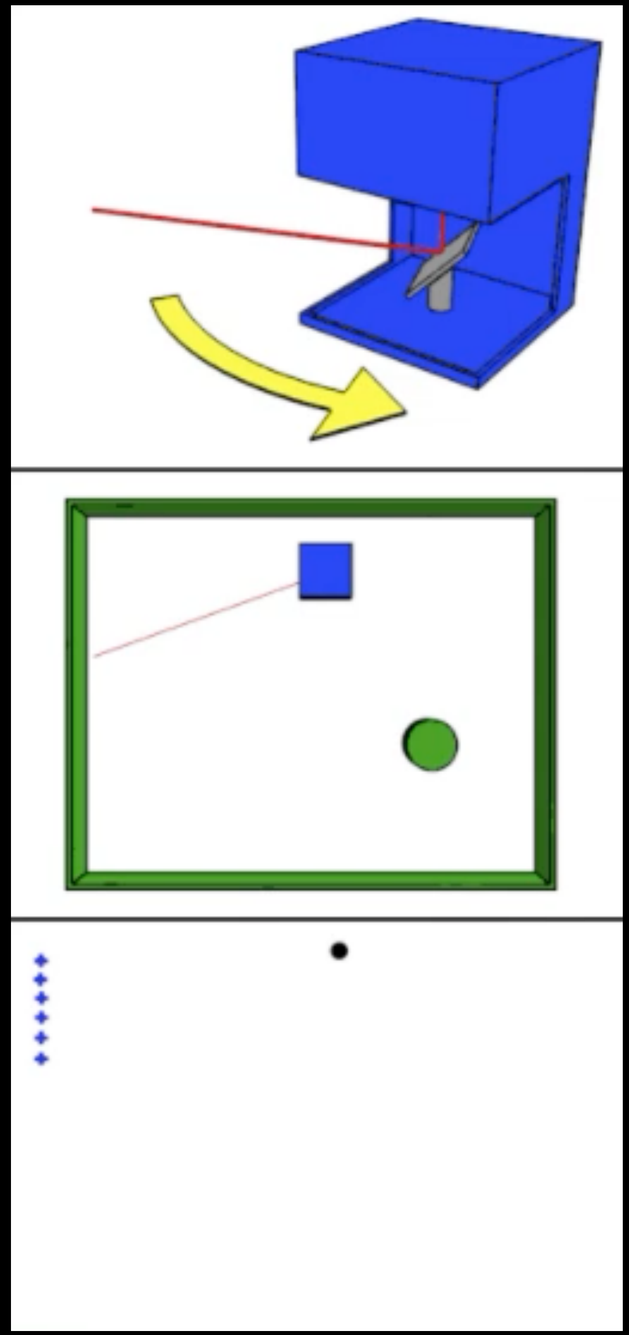
Light detection and ranging

Visible or Near-IR light used to image objects

Object distance calculated from time-of-flight estimates

Several advantages

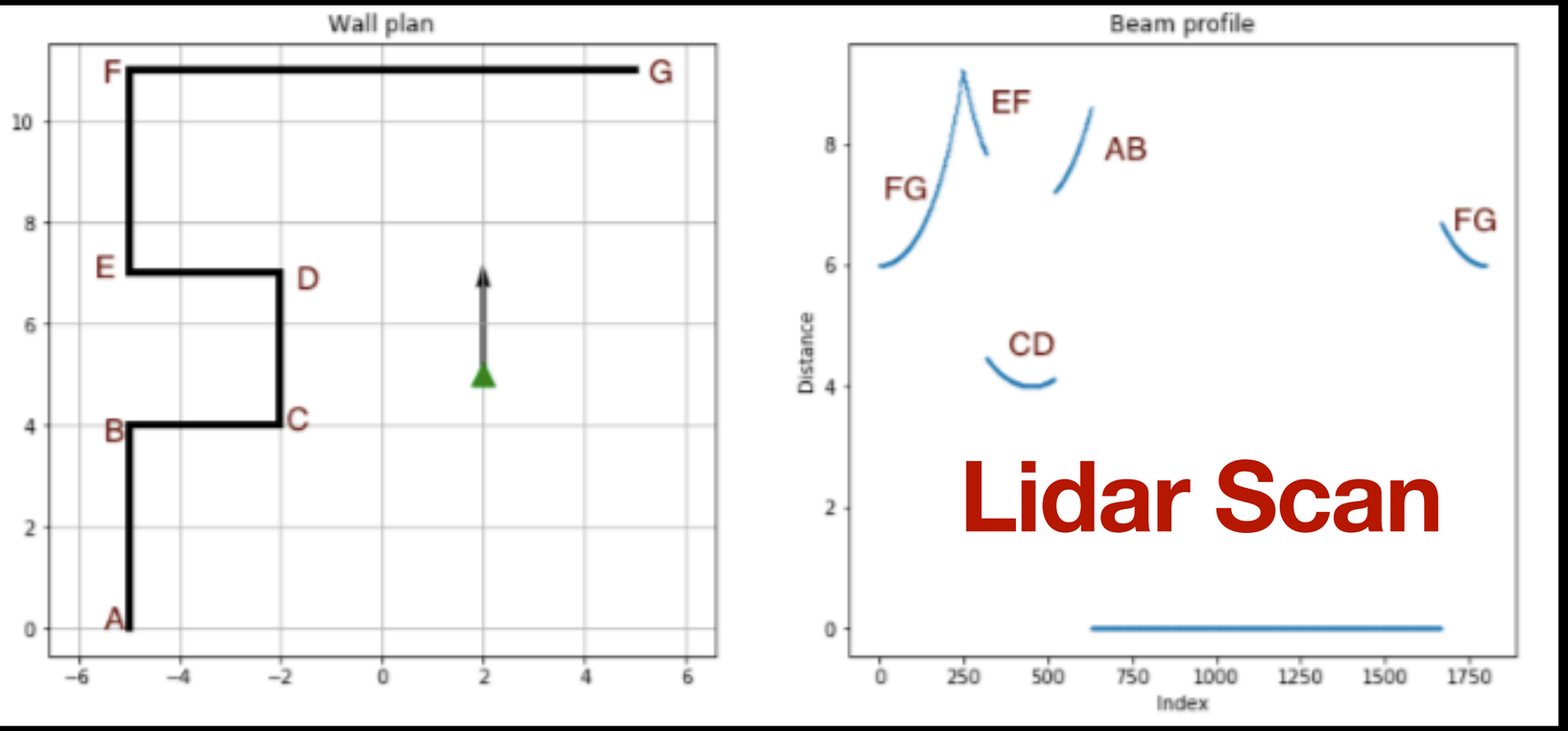
- Low Noise floor
- Independent of ambient lighting
- “Ready-made” 3-D imaging

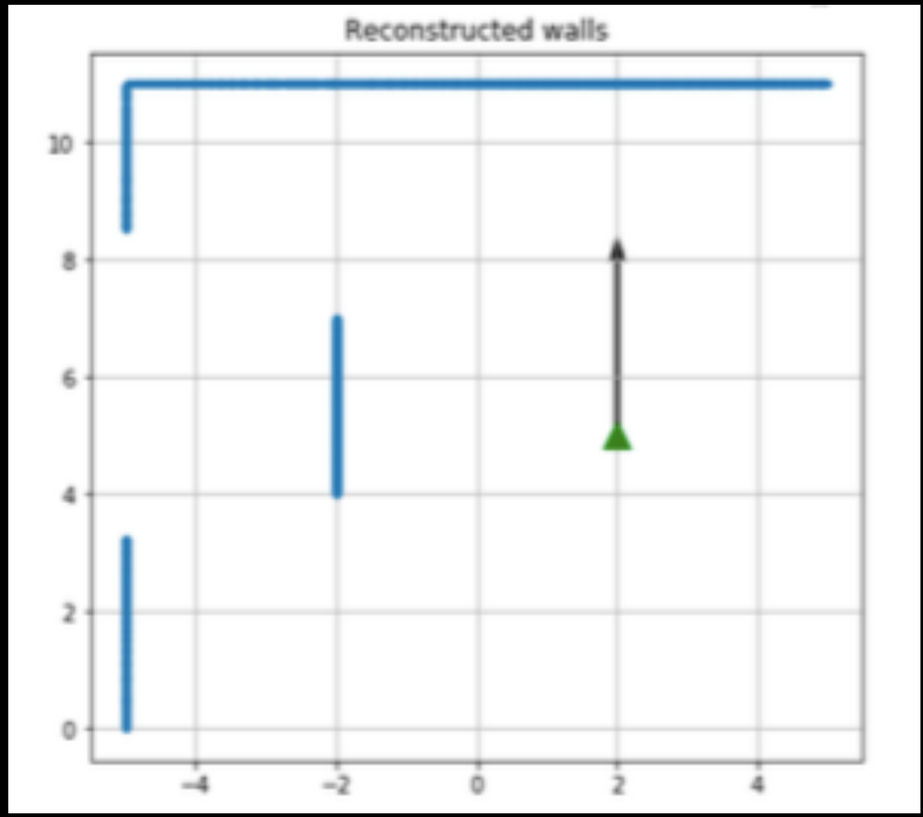


Video: https://en.wikipedia.org/wiki/Lidar

2-D Lidar

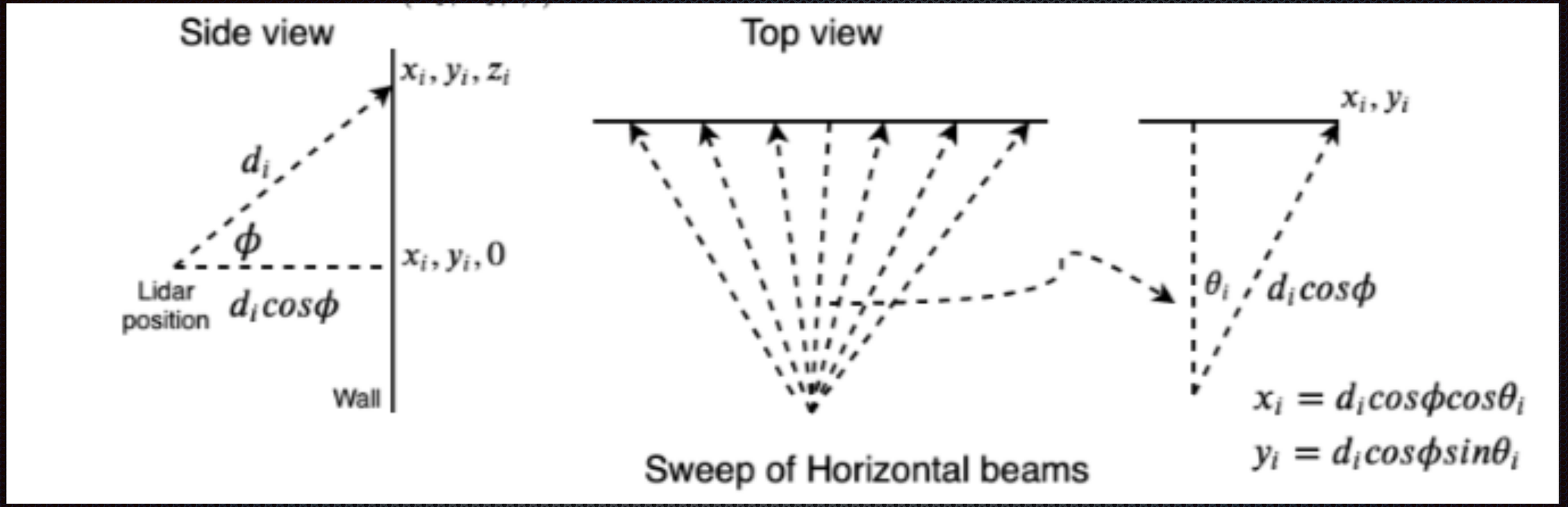
Single Laser beam rotating 360 degrees
Return reflections d recorded along with azimuth angle θ





2-D Lidar simulator animation in Jupiter notebook

3-D Lidar



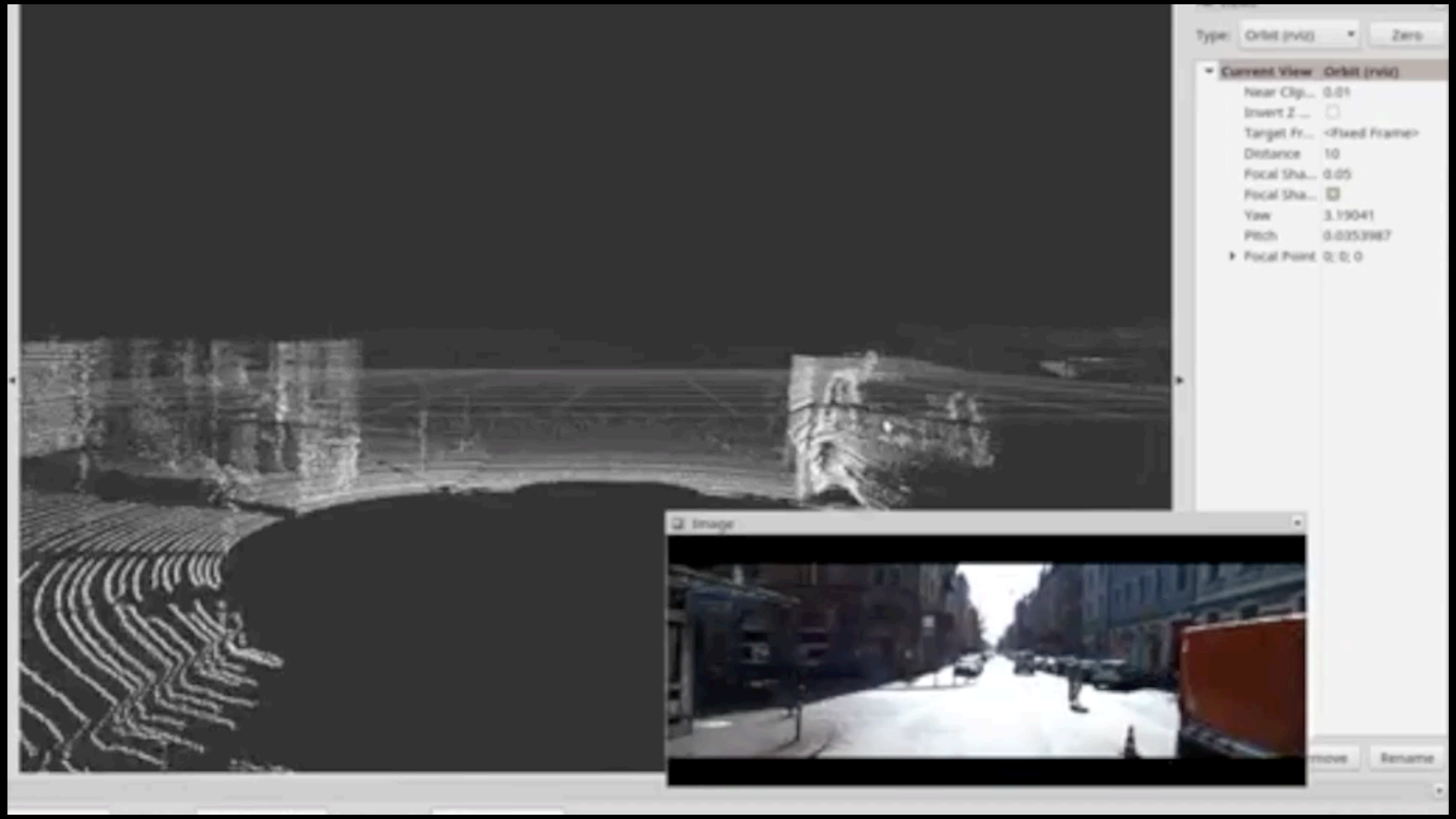
N-laser beams rotating 360 degrees
Span different elevation angles ϕ

Object which reflects is at (d, θ, ϕ)

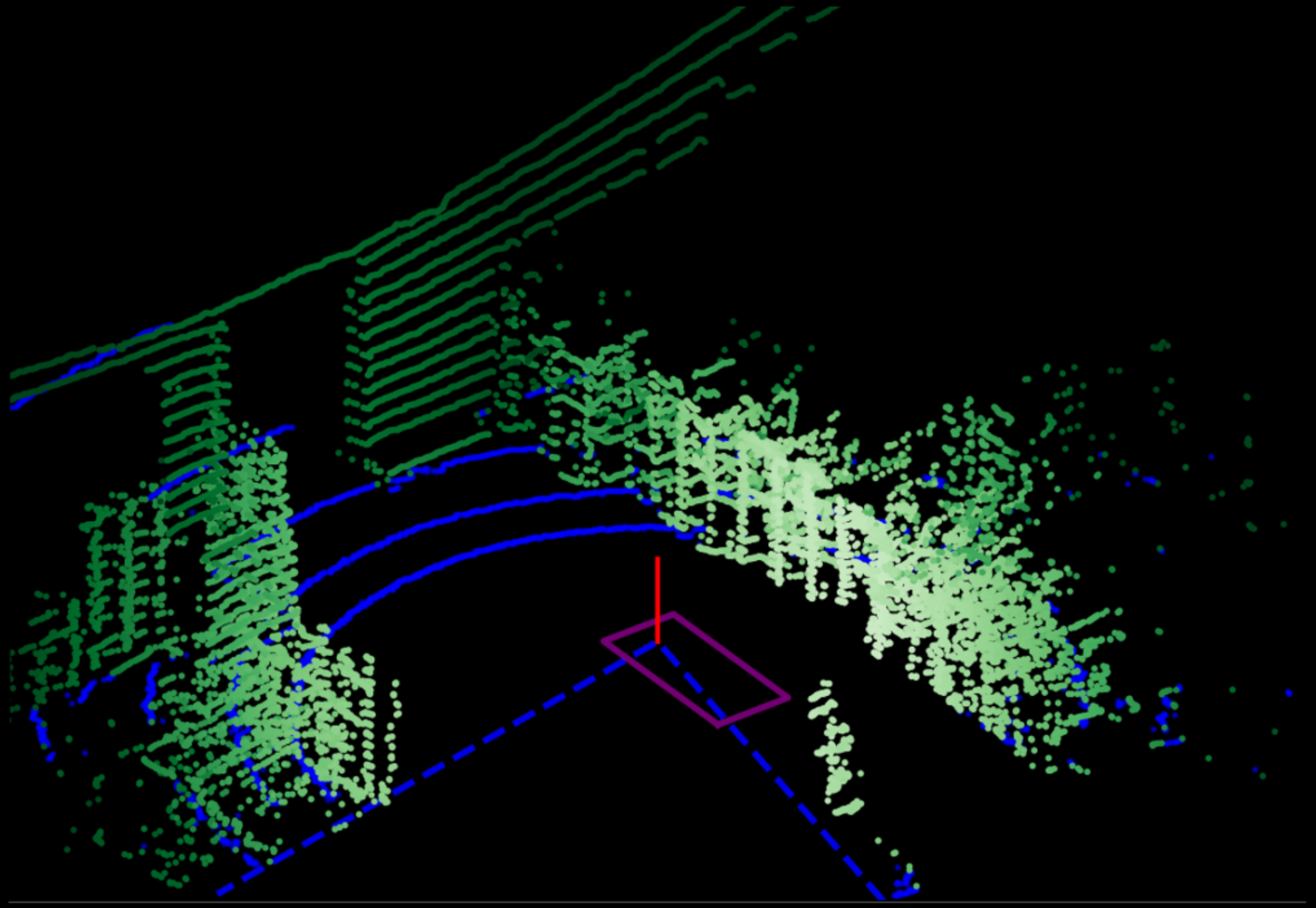
- **(θ, ϕ) is given by beam position**
- **d is calculated by time-of-flight**

Can be converted to cartesian coordinates (x, y, z)

3-D Lidar visualization

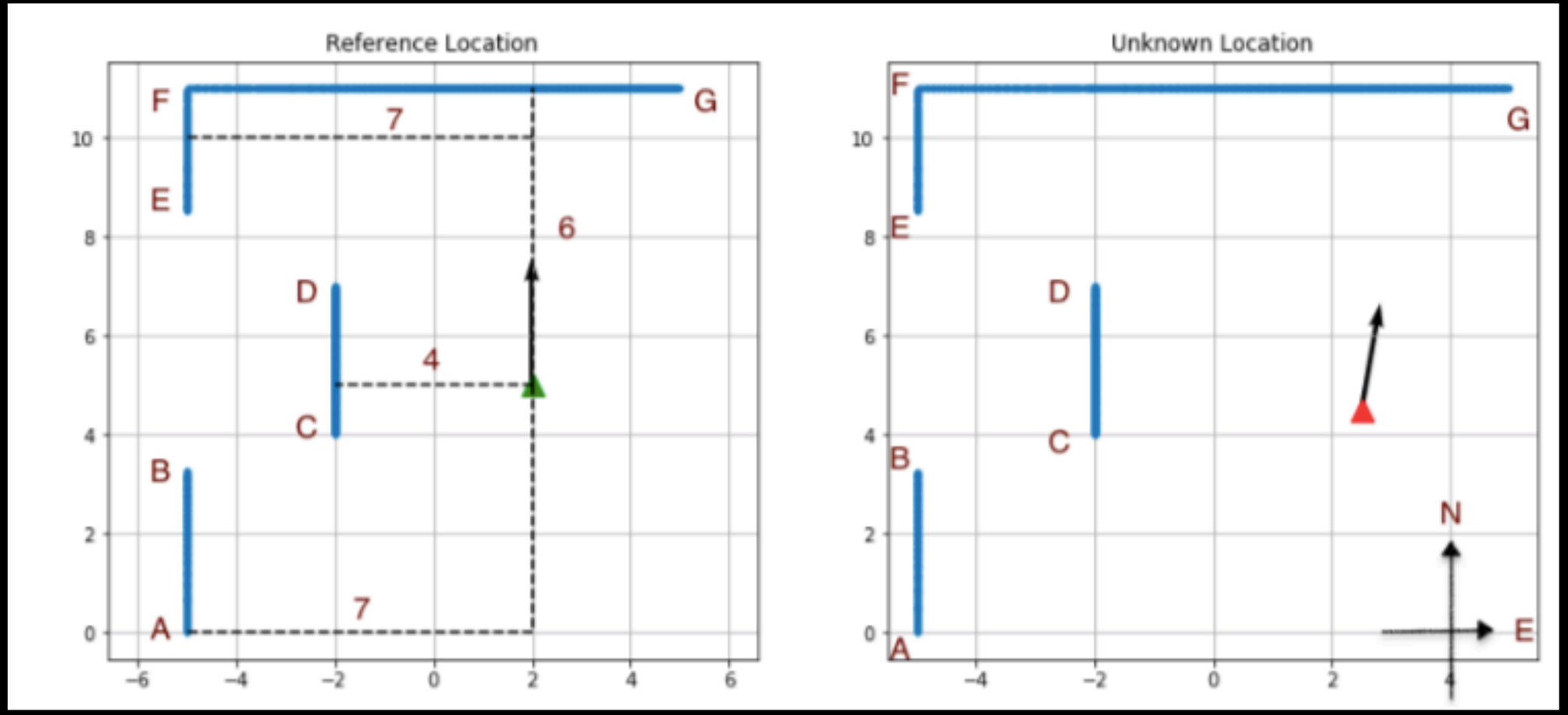


Count the number of vertical beams in this Lidar scan



Localization in 2D maps

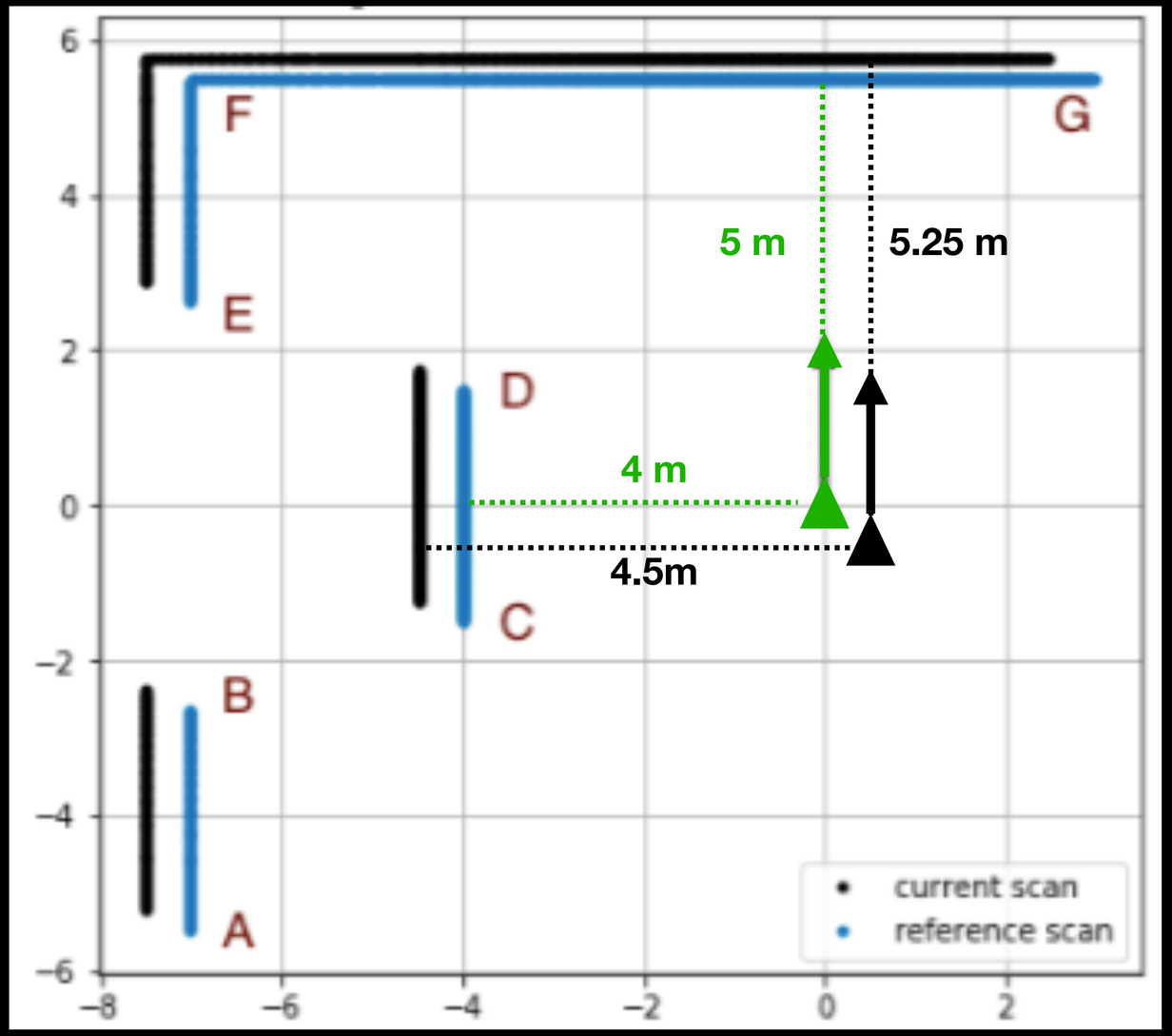
Where is the robot in the known map?



Scan-Matching

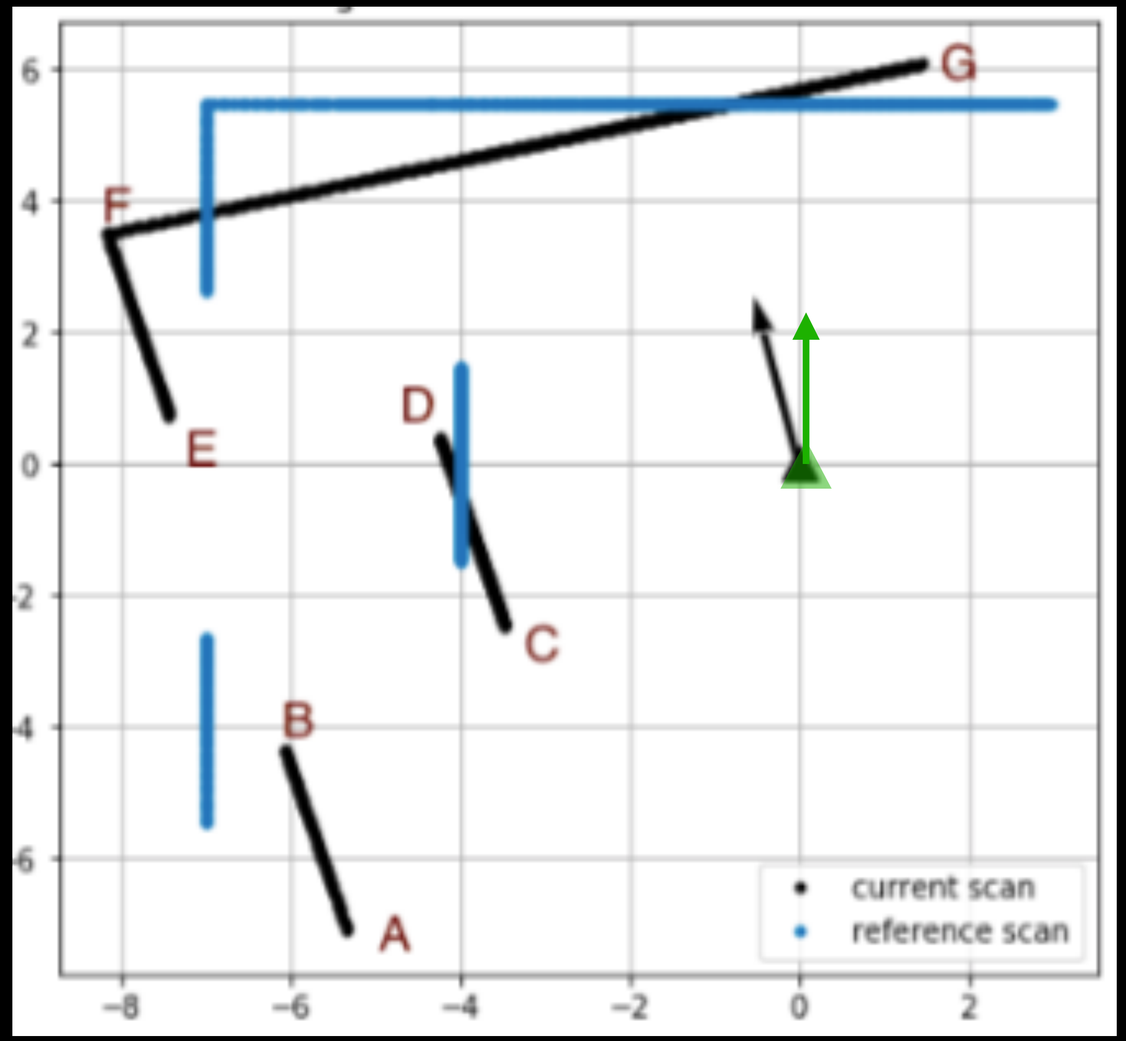
- Wall positions relative to “green” robot position known
- Can the “red” robot position be determined? Yes
  - Lidar scan at red position is made available
  - Distance/Orientation between green and red positions is not widely different

Pure translation



- Blue walls - observed when robot is at $(0,0)$
- Black walls - observed when $(0.5, -0.25)$
- Walls EF, CD, AB have moved 0.5m away
- Wall FG has moved 0.25m away

Pure rotation



- Vehicle stays at $(0,0)$
- There is a rotation of 15 degrees (somewhat high)
- Current observation
  - All 4 walls have also been rotated counter-clockwise 15 degrees
  - Note that other hidden walls may appear

Robot motion: translation and rotation

- Assume a lidar scan has observations for M points
- Pure translation of (t_1, t_2)

$$\begin{pmatrix} x_i^q \\ x_i^q \end{pmatrix} = \begin{pmatrix} x_i^p \\ y_i^p \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} \; \forall i = 1,2,\cdots,M$$

- Pure rotation of angle θ

$$\begin{pmatrix} x_i^q \\ x_i^q \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_i^p \\ y_i^p \end{pmatrix} \; \forall i = 1,2,\cdots,M$$

Iterative Closest point (simple version)

- Two lidar scans P and Q
  - P is reference scan of robot
  - Q is the new scan of robot
- Say, each scan has M points

$$P = \{(x_1^p, y_1^p), (x_2^p, y_2^p), \cdots, (x_M^p, y_M^p)\}$$

$$Q = \{(x_1^q, y_1^q), (x_2^q, y_2^q), \cdots, (x_M^q, y_M^q)\}$$

- Pure translation

$$Q = P + t$$

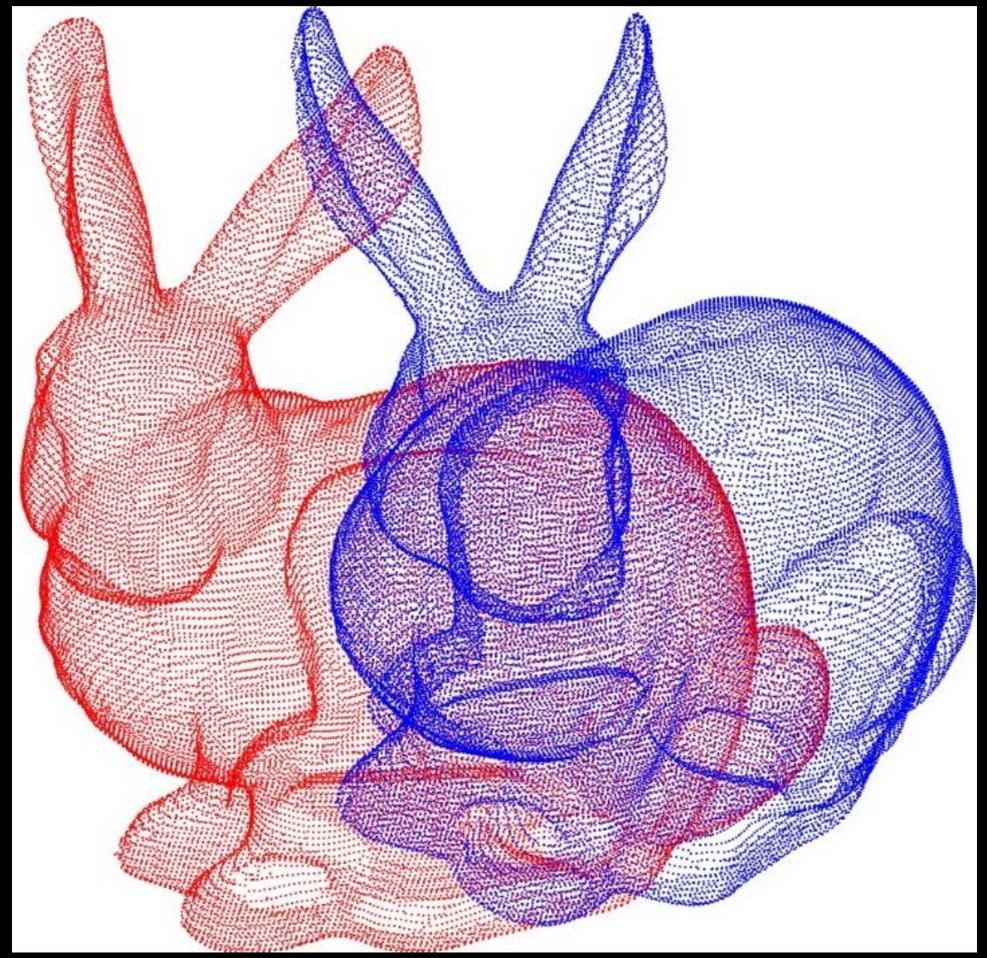
- Pure rotation

$$Q = RP$$

- ICP

$$\min_{R,t} ||RP + t - Q||^2$$

How much did observer move between red and blue scans of the rabbit?



What is R and t ?

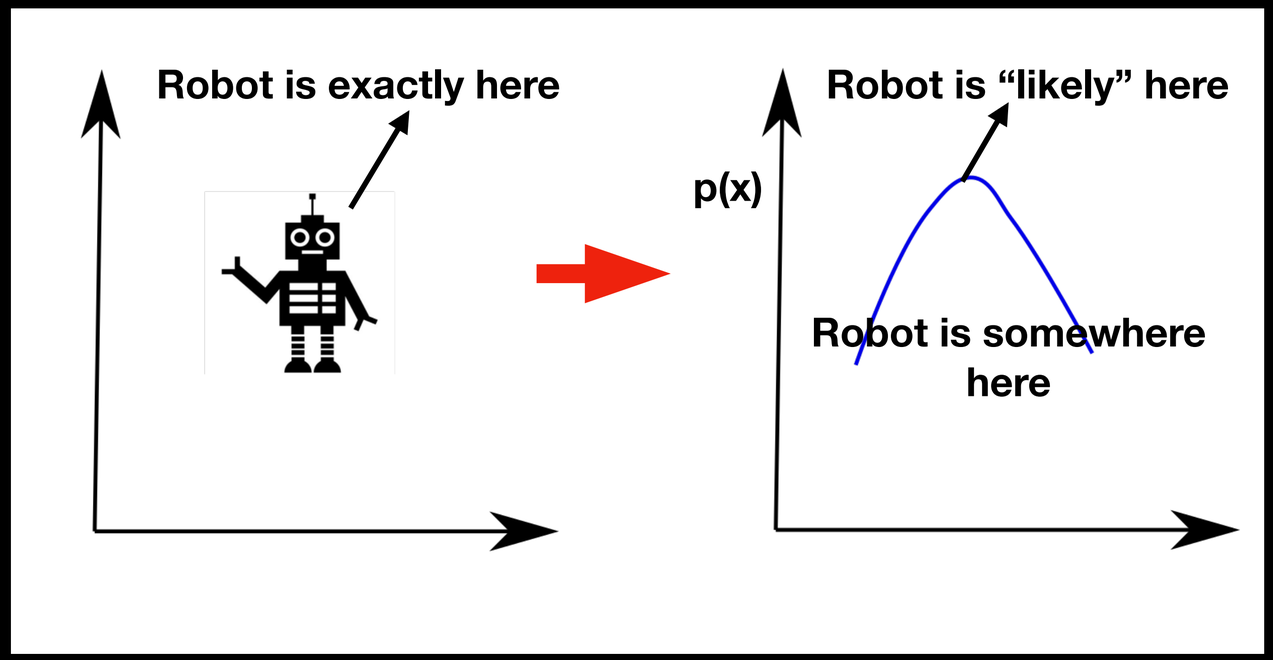
Scan matching: relative motion between P and Q

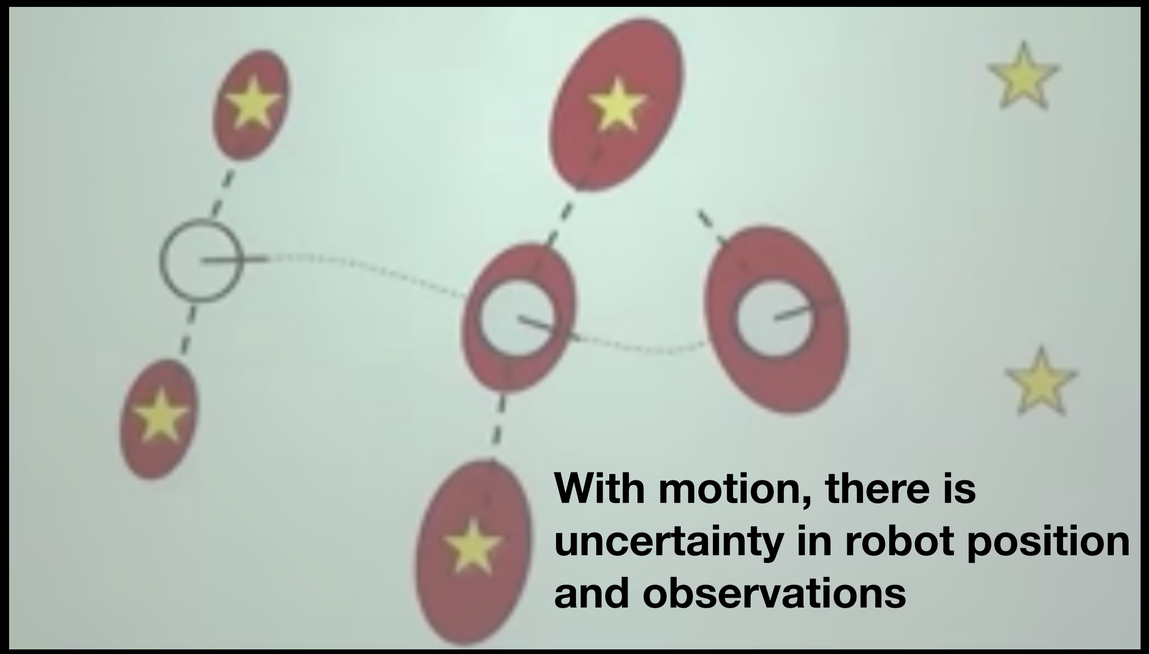
Localization

Need for probabilistic approaches

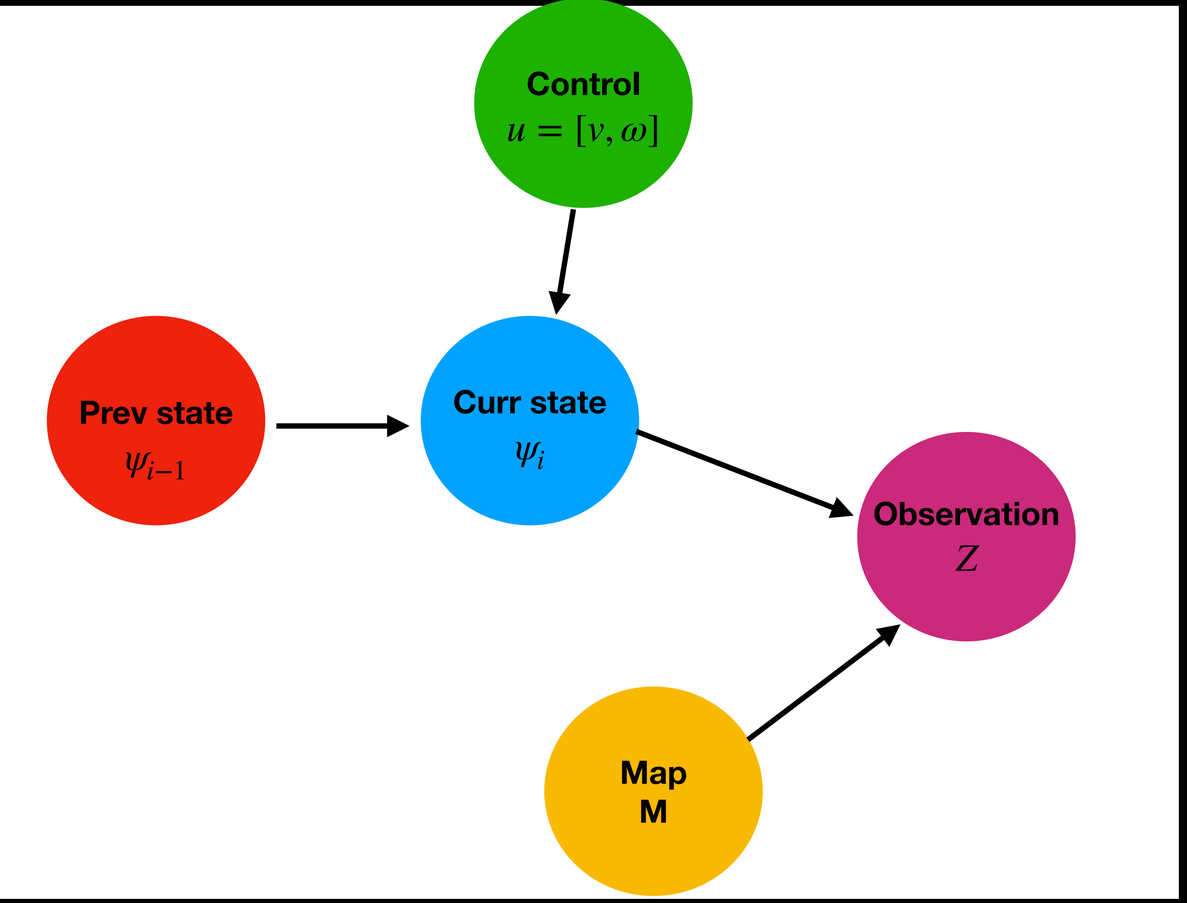
Uncertainty in both robot's motion as well as sensor observations

Probabilistic techniques to explicitly model uncertainties





Localization: Graphical model view



Robot state $\psi_i = (x, y, \theta)$ at epoch i

Control $u = [v, \omega]$ applied at epoch $i - 1$

Robot states are hidden

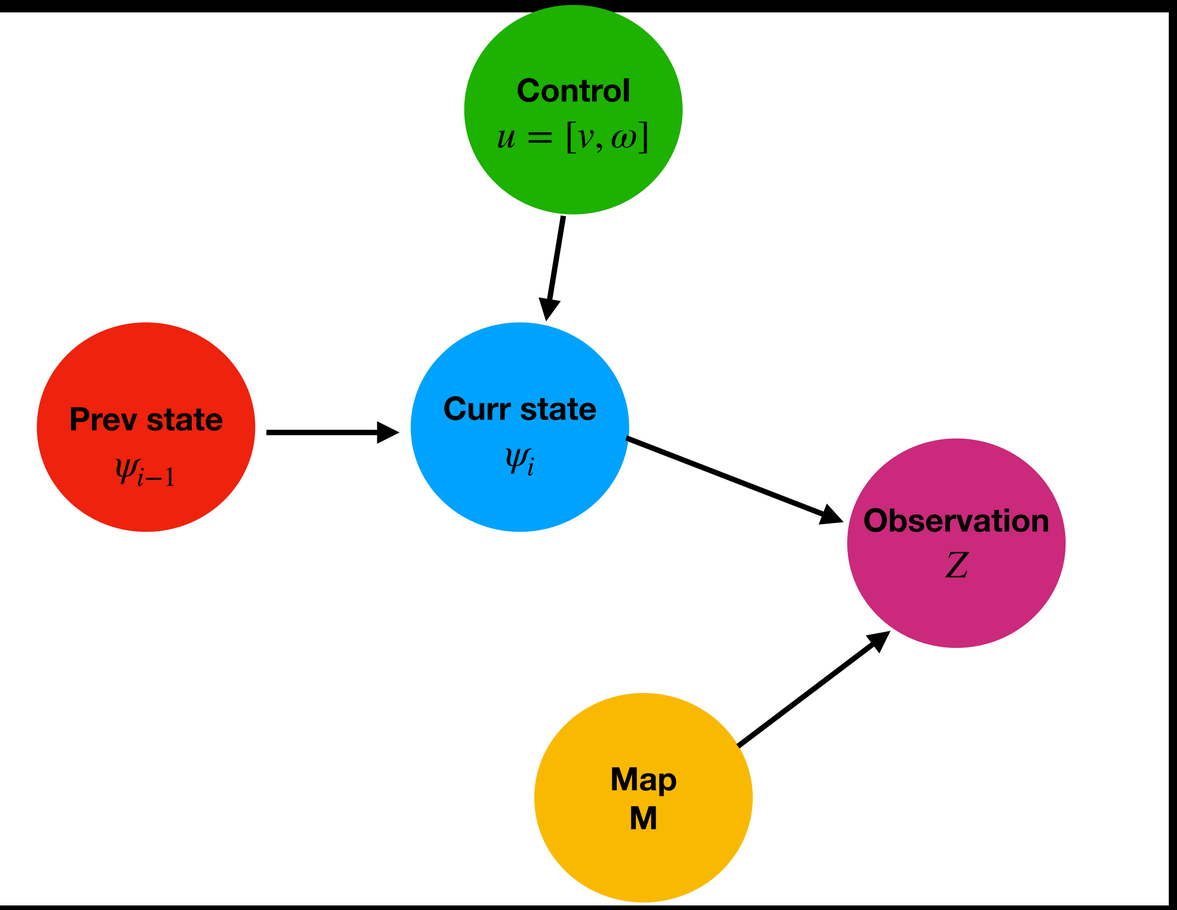
Access to observations (Lidar scans)

Pose evolution is Markovian given control u

- ψ_i not dependent on trajectory taken to reach ψ_{i-1}
- $P(\psi_i \mid u, \psi_{i-1}, \psi_{i-2}, \cdots) = P(\psi_i \mid u, \psi_{i-1})$

Fig source: EB Olson, “Real-time Correlative scan matching”, ICRA 2009

Localization



Our Belief of robot state

- Most probable state at *i* given all measurements made till *i*

Posterior can be shown to decompose:

$$p(\psi_i \mid \psi_{i-1}, u, z, m) \propto p(\psi_i \mid \psi_{i-1}, u) p(z \mid \psi_i, m)$$

Motion model: $p(\psi_i \mid \psi_{i-1}, u)$ ~ prior

- Ex: from wheel encoder measurement

Observation model: $p(z \mid \psi_i, m)$ ~ likelihood

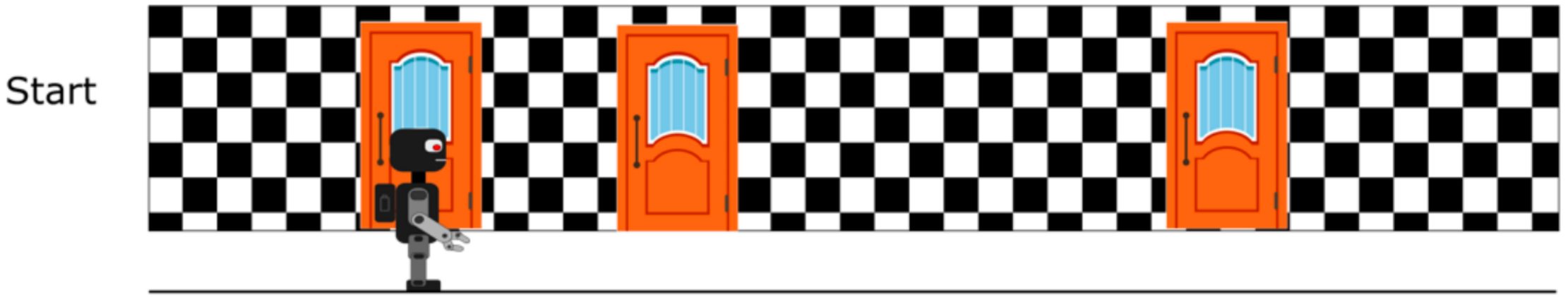
- Ex: from Lidar measurement

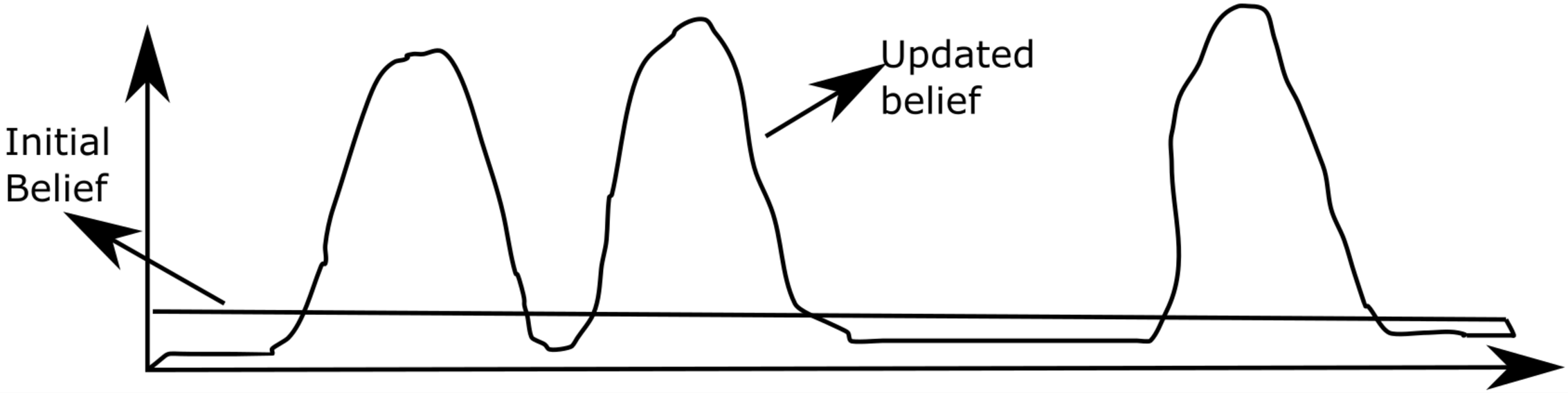
- Recursive refinement of our belief

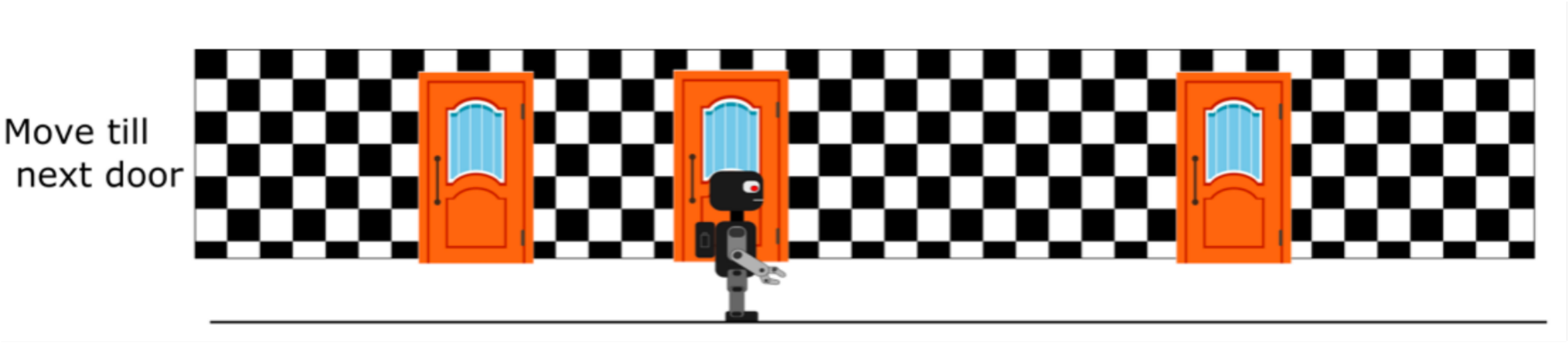
Bayesian filtering

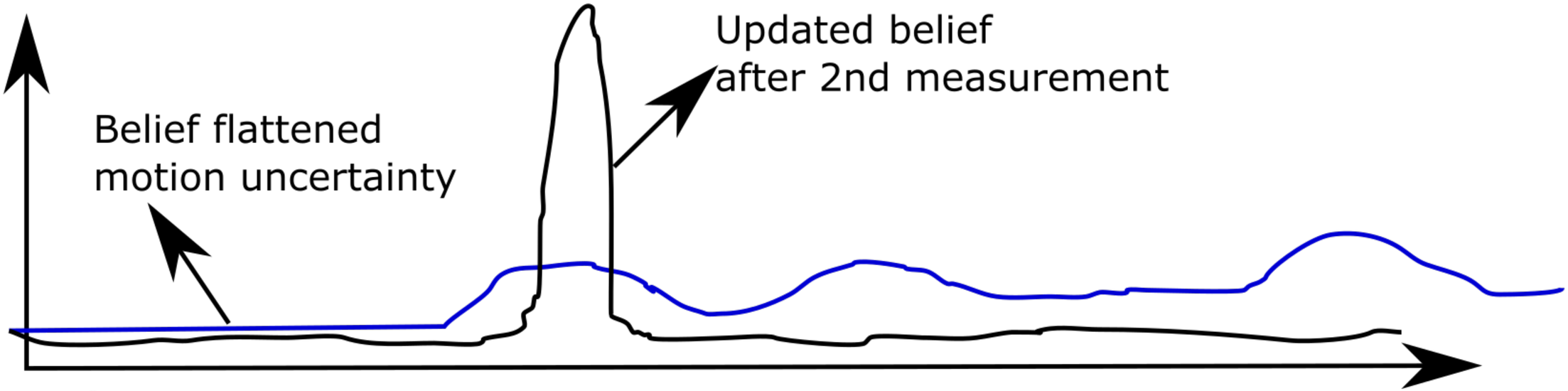
Recursive Localization example

1-D robot world, Robot can tell if it sees door or not









Source: Thrun, Burgard, Fox, “Probabilistic Robotics”

Descriptive Statistics

Random variables, Mean, Variance, Gaussian distribution

- Random Variables
    - Bit of a misnomer
    - Function maps outcomes of a random experiment to real-values
- Say, X is the random variable
    - Coin toss => $X \in$ {Heads, Tails}
    - Die thrown => $X \in \{1, 2, 3, 4, 5, 6\}$
    - Political party of choice in a survey => $X \in$ {Party A, Party B, Party C, … }
    - Temperature of people entering an office
        - Continuous
        - $X \in [96, 103]$
- **Self-paced exercises in Python notebook**

Sensor fusion

Lidar measurements still have noise (all sensors do!)

Scan matches are not perfect

- Dynamic changes (occlusion)
- “Bad” static objects (Glass, mesh, reflective surfaces)
- Sparsity of scatterers

Wheel encoders

- Smooth outputs in the short-term
- Drifts over long periods of time

Lidar

- Noisier output
- Stable over long runs

Kalman filter - combine the pose estimates from multiple sensors

1-D Kalman filter

Two thermometers have following standard deviations

Therm 1: 3 deg C

Therm 2: 1 deg C

[Calibration could have been done by measuring and comparing against known objects]

One particular person was measured 101 and 98 on Therm 1 and Therm 2

a. Does he have fever? How to combine the 2 readings?

b. Is the combined reading “better” than the individual reading?

$$\sigma_1^2 = 9 \text{ and } \sigma_2^2 = 1$$

$$\hat{T} = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} T_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} T_2$$

$$\hat{T} = 0.1 * 101 + 0.9 * 98 = 98.3$$

$$\hat{\sigma}^2 = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2}$$

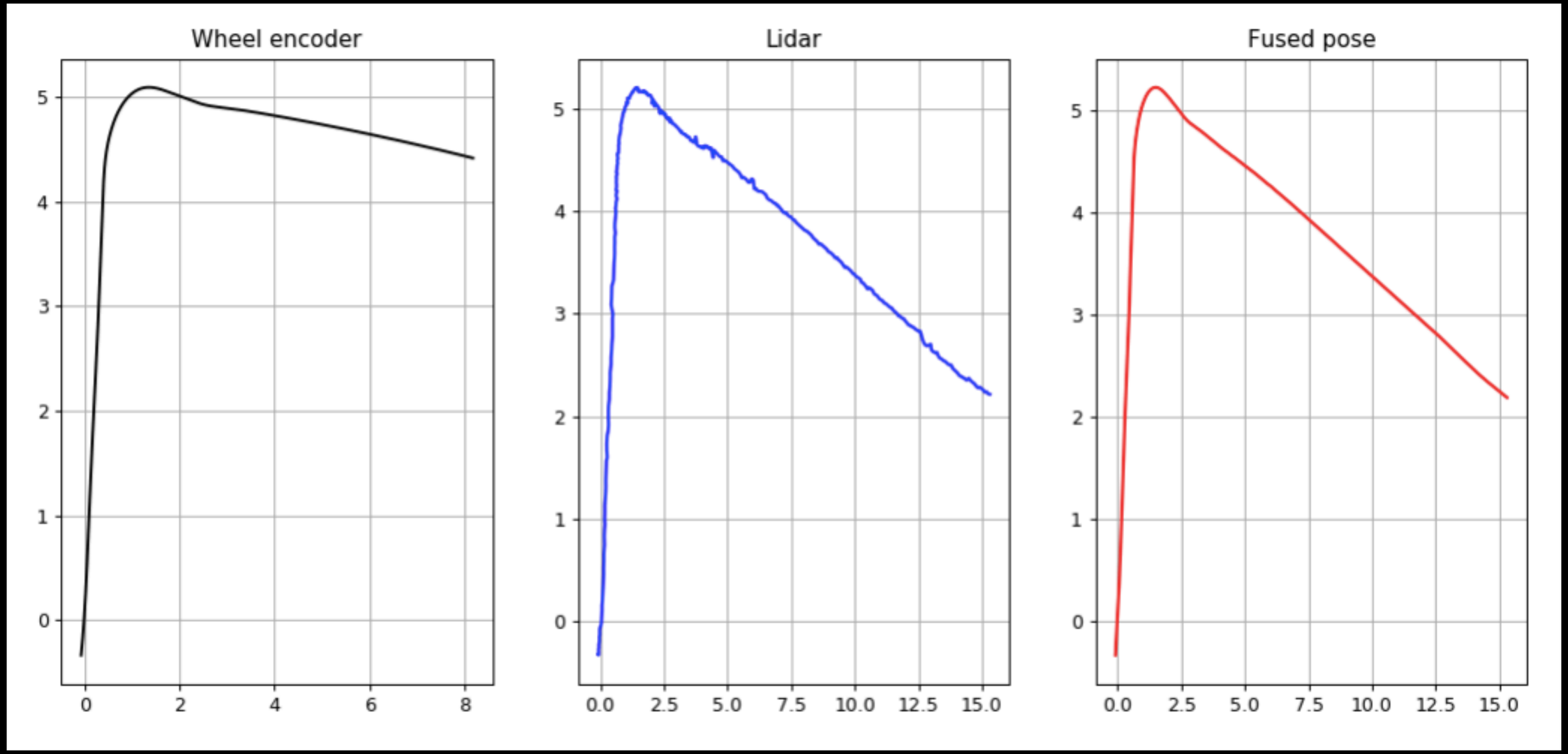
$$\hat{\sigma}^2 = 0.9$$

Better than a single measurement!

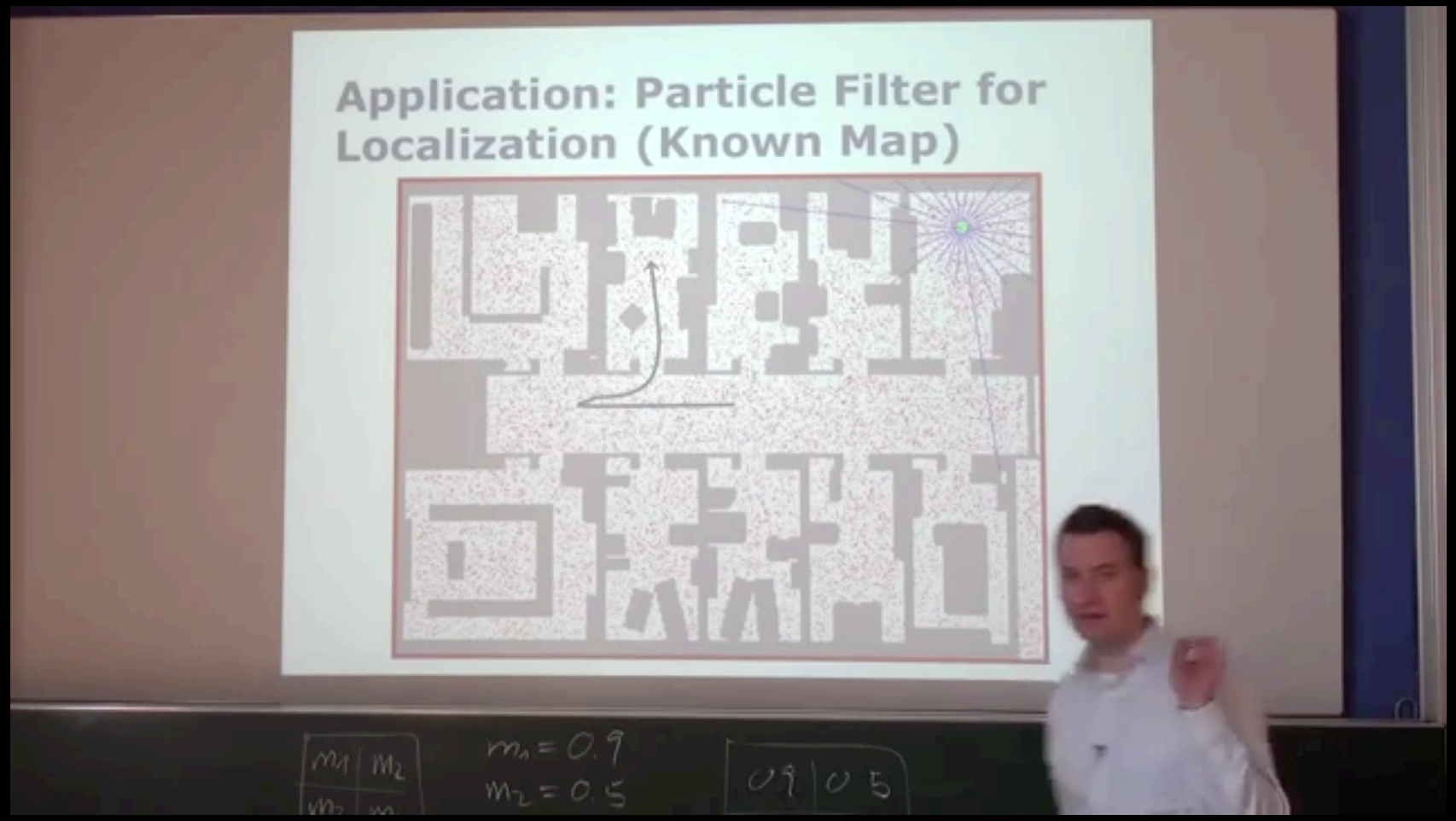
Extensions to higher-dimensional Kalman filter

- State is not scalar temperature
    - 3-tuple robot pose: (x, y, θ)
- Wheel encoder measures v, ω
    - Successive states can be estimated using unicycle kinematics model
- Lidar estimates robot pose by scan-matching
- Kalman filter
    - Fused pose weighs relative confidence of Lidar and wheel encoder estimates
    - Covariance used instead of standard deviation

Robot localization using Kalman filter

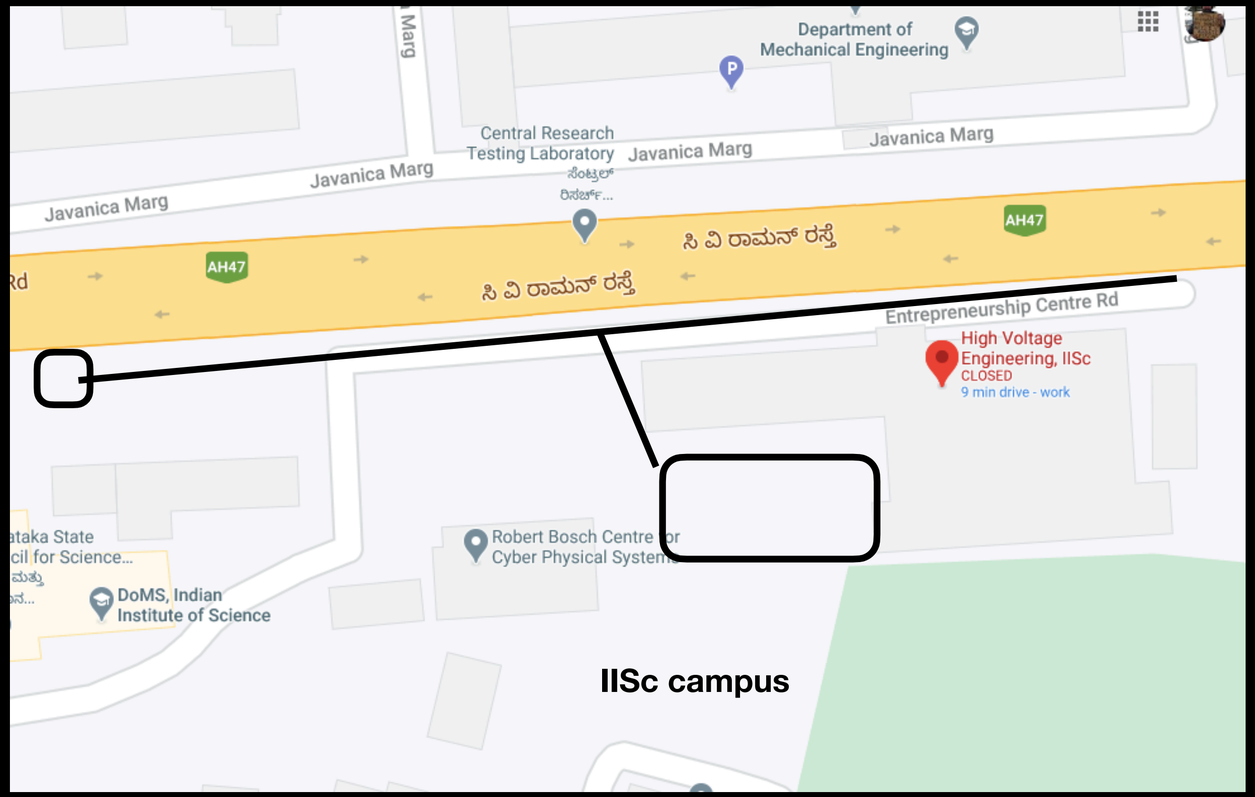


Localization in a large map using Particle filter

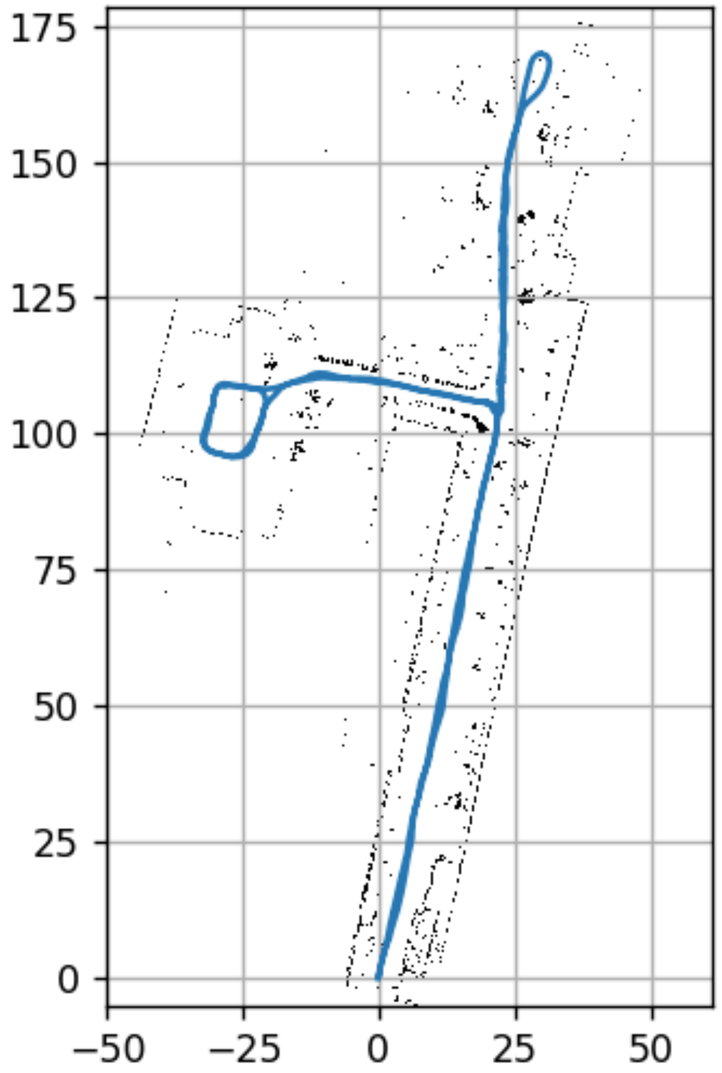


Source: C. Stachniss, SLAM lectures

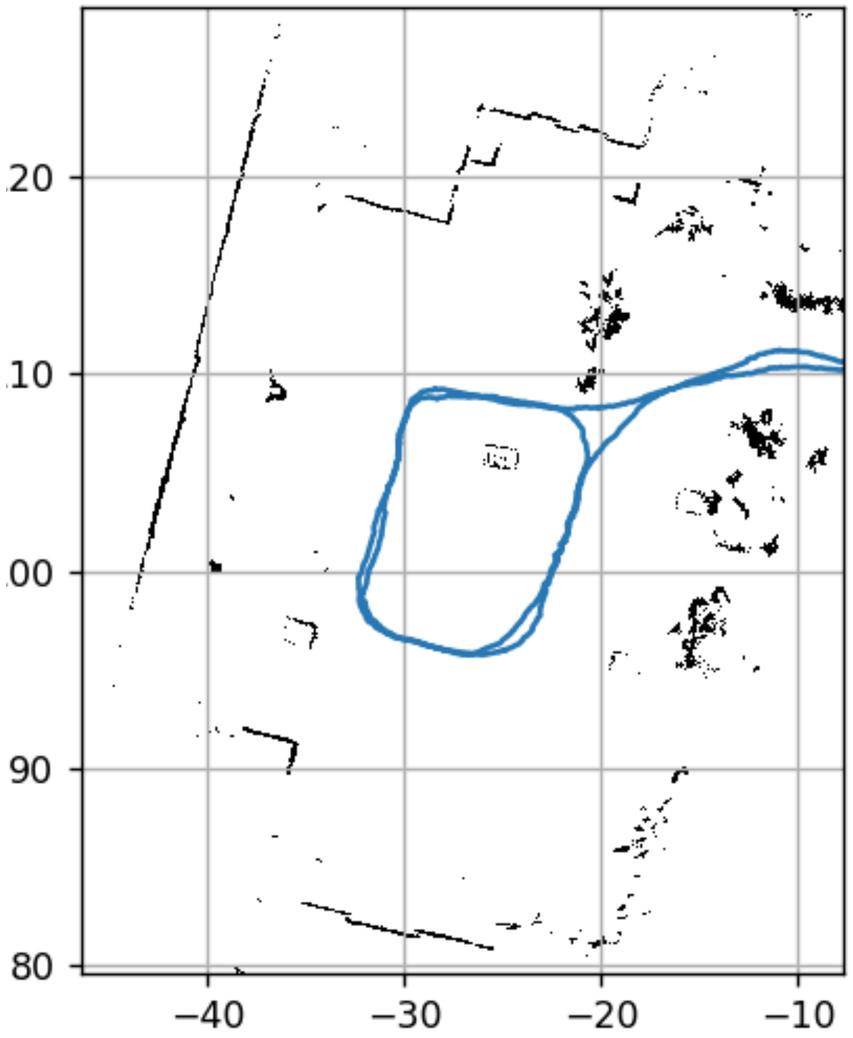
How is mapping done?



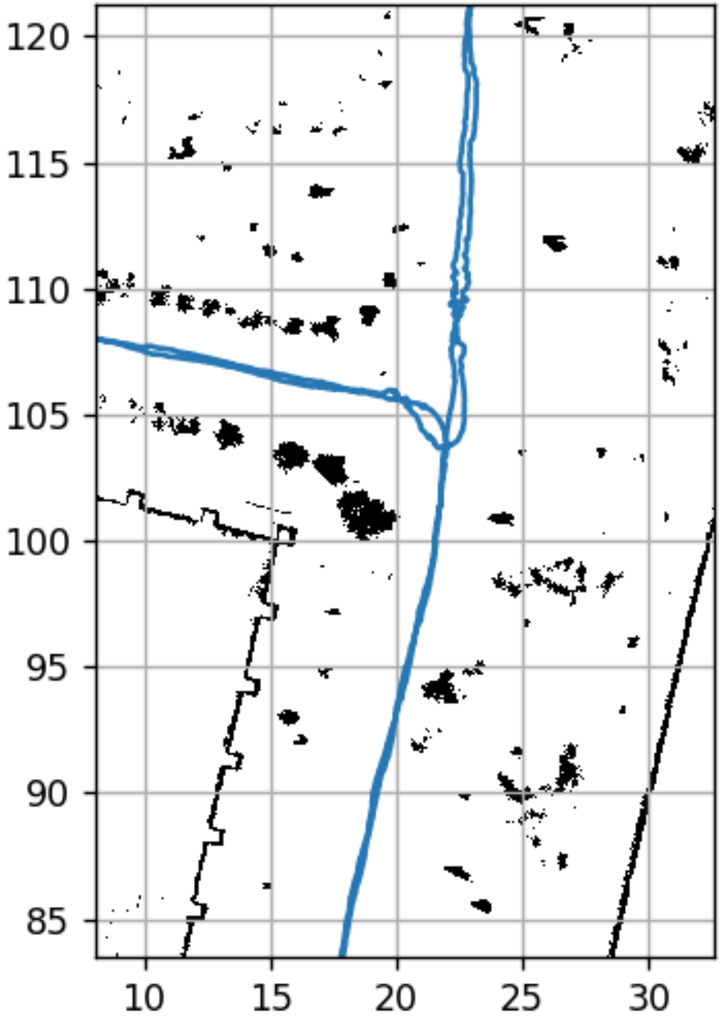
Overall map



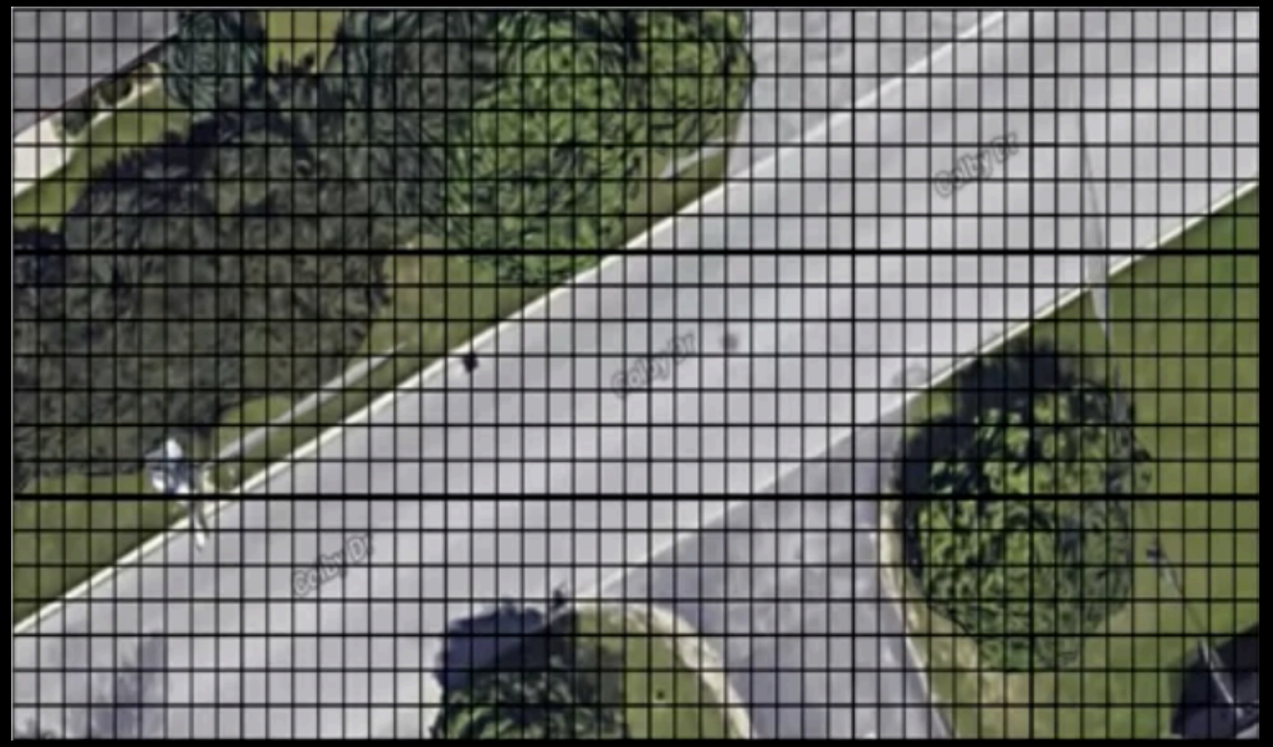
Parking lot

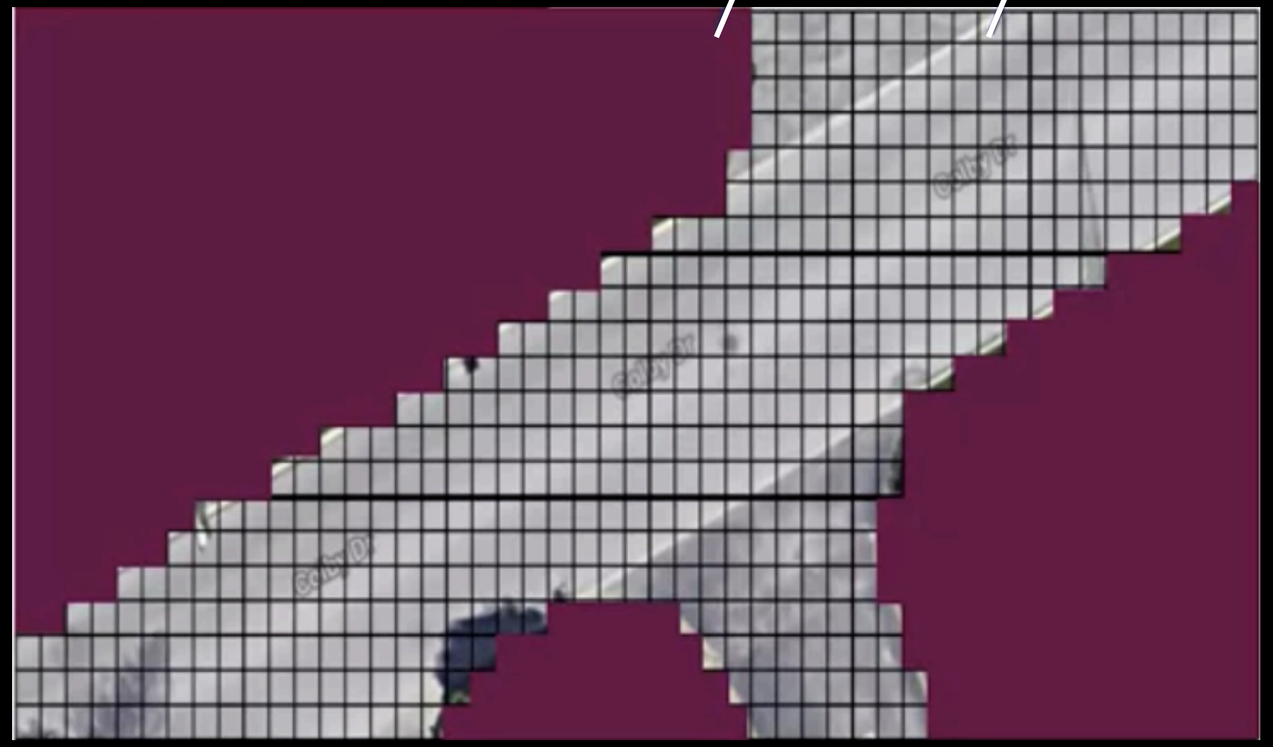


Cross-over loop



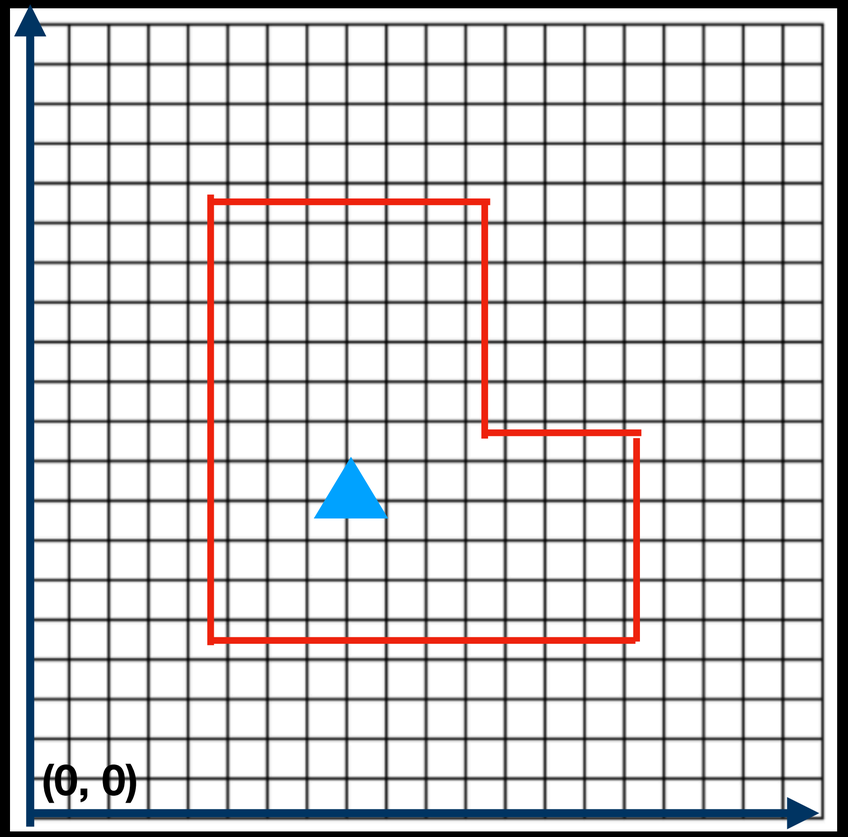
Grid map representation

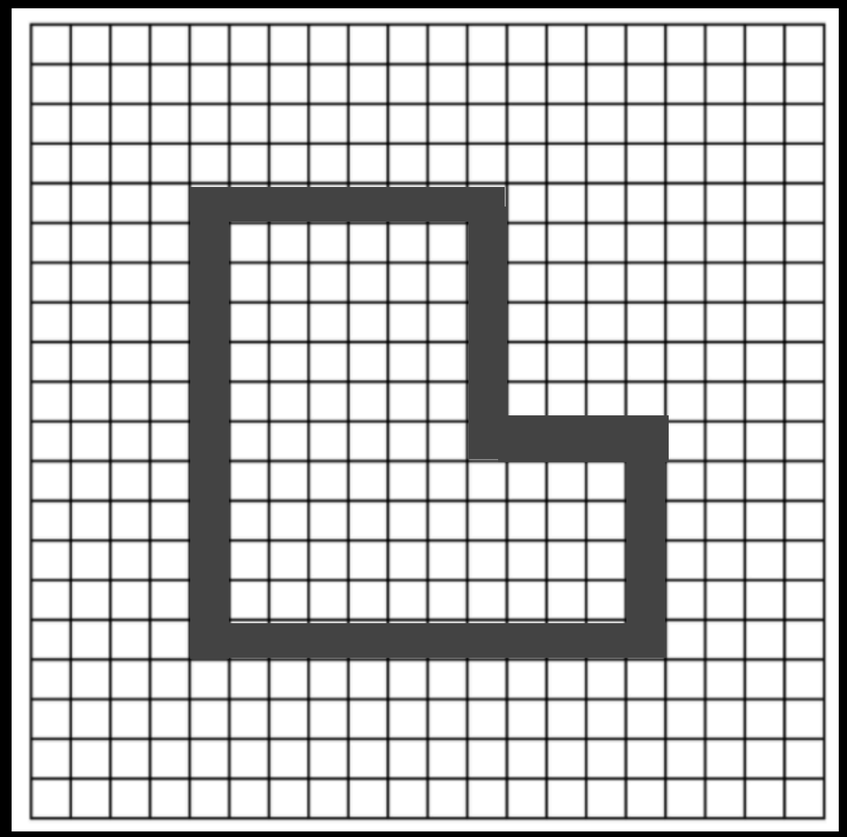




2-D array of 0’s and 1’s

Occupancy grid maps





Robot's operating environment (red polygon room in example)

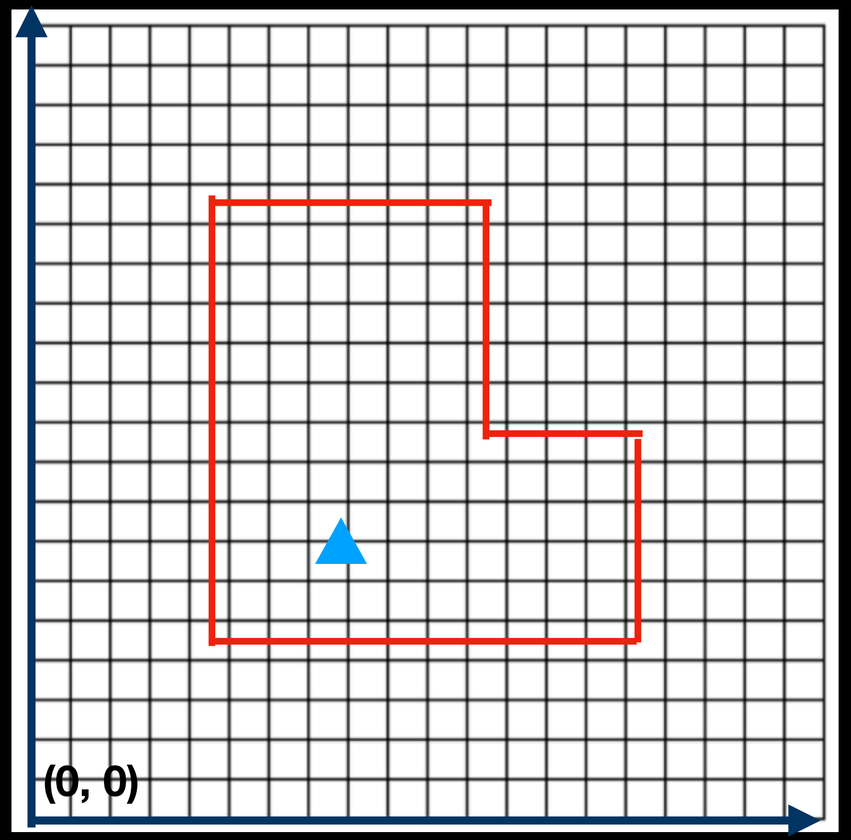
- Represented in form of grid
- Each grid value = occupied/ unoccupied

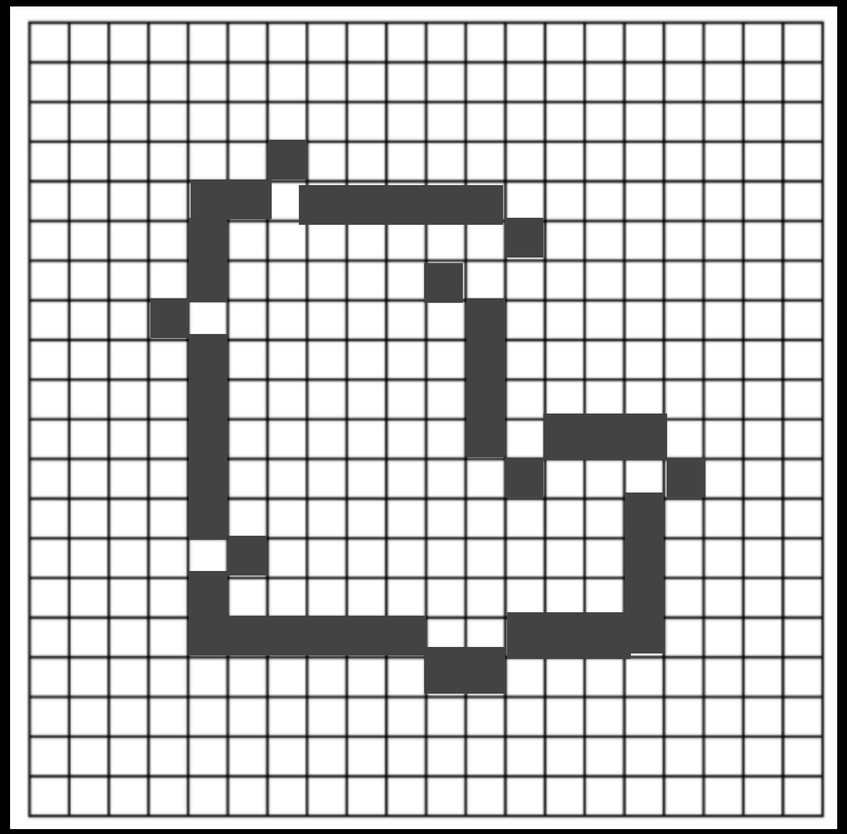
Robot's sensors differentiate free space and obstacle locations

Map precision depends on

- Grid resolution
- Sensor noise

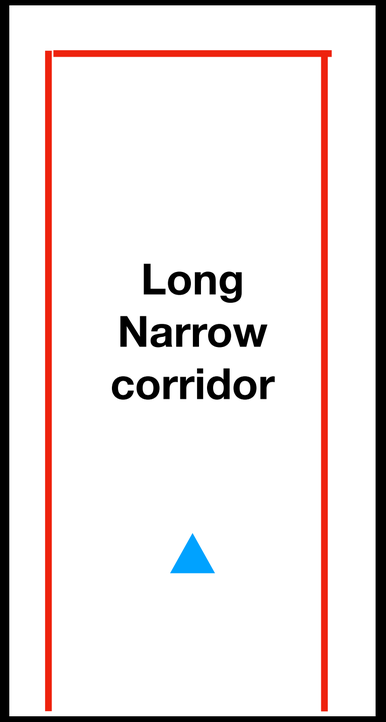
Error sources in mapping

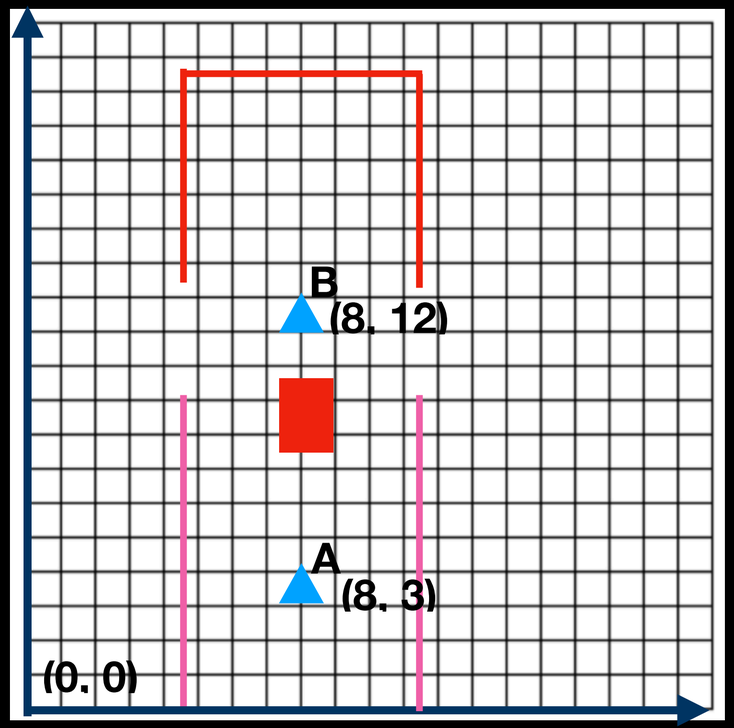


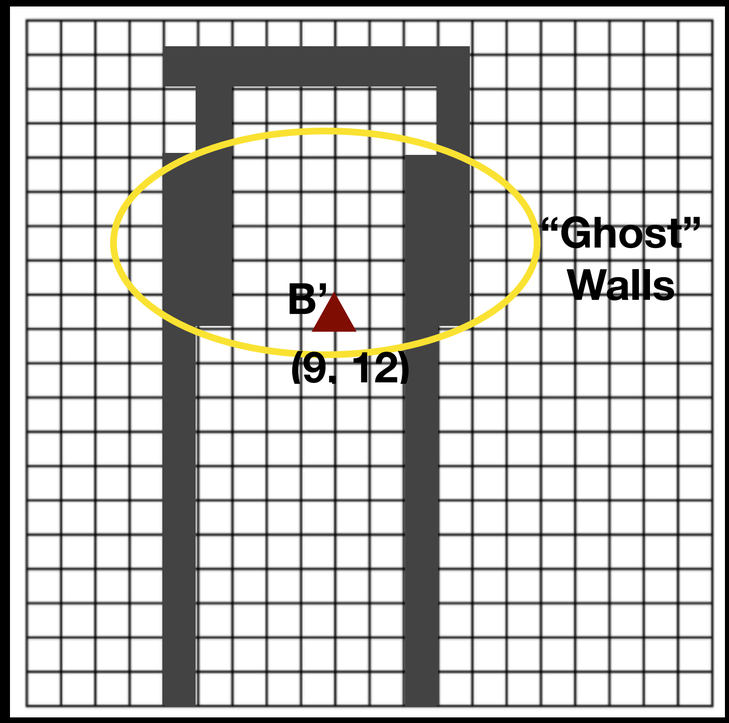


Sensor Noise gives rise to fuzzy maps

Motion drift during mapping





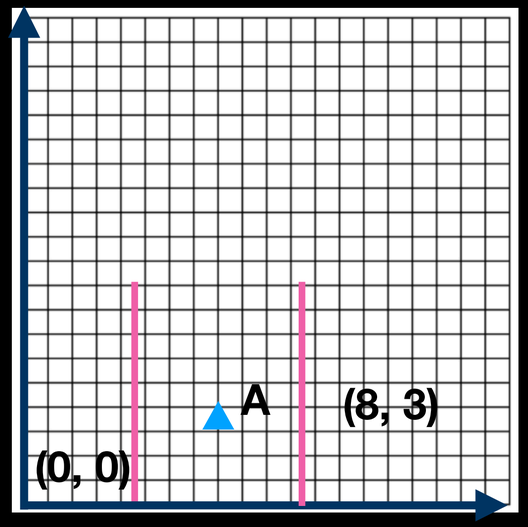


Entire robot environment not visible in single frame
Robot has to move/explore
Long Narrow corridor example

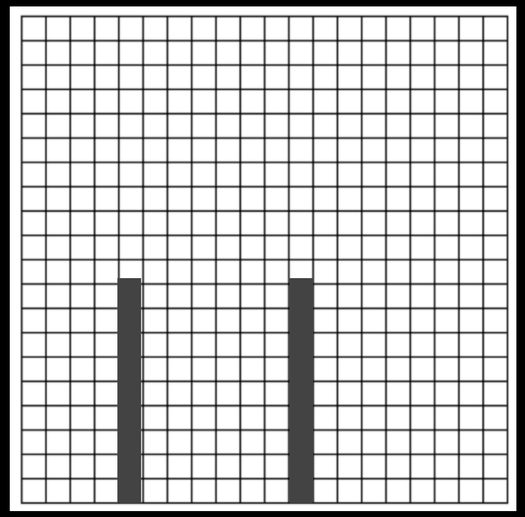
- Robot is initially at position A
- Grid populated based on partial measurements (shown in pink)
- Final section of corridor is visible when robot reaches position B
- Say, B is estimated to be at (9, 12) instead of (8, 12)
- New measurements are populated in wrong grids
- Map building is sensitive to errors in motion estimate makes

Simultaneous Localization and Mapping

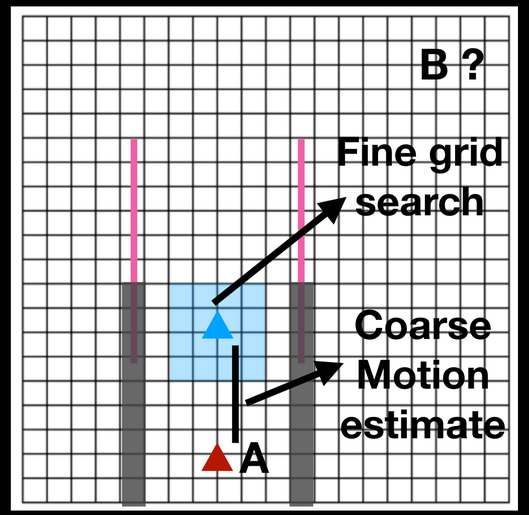
Known starting position



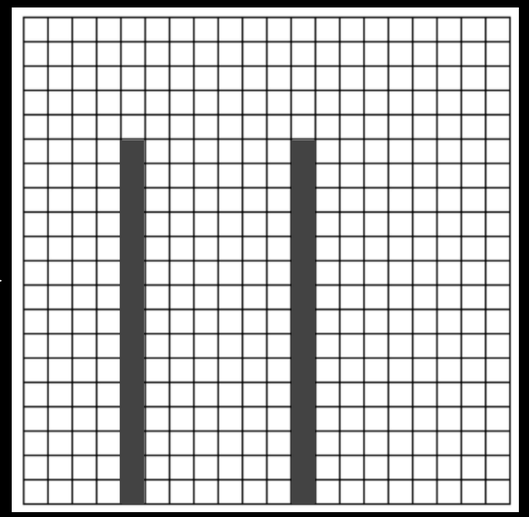
Map from measurements@ A



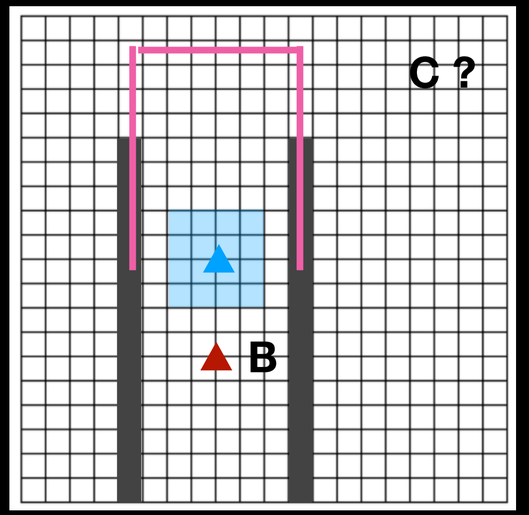
Robot has moved to B



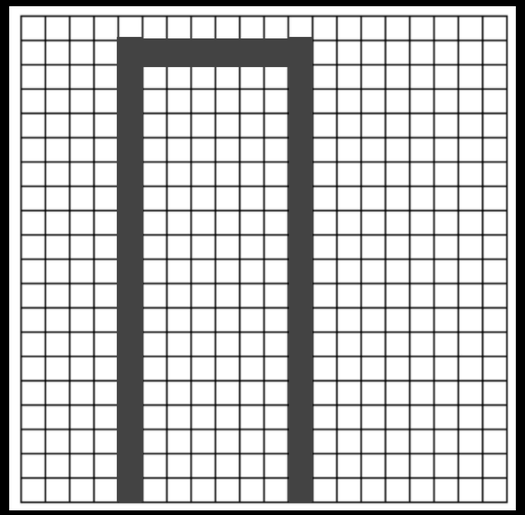
Map from measurements@ A and B



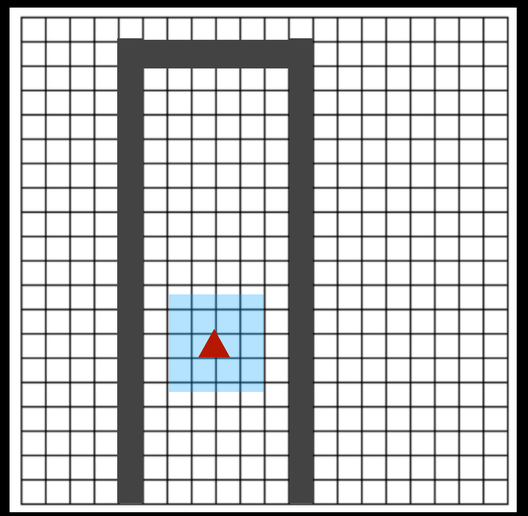
Robot has moved to C



Final Map

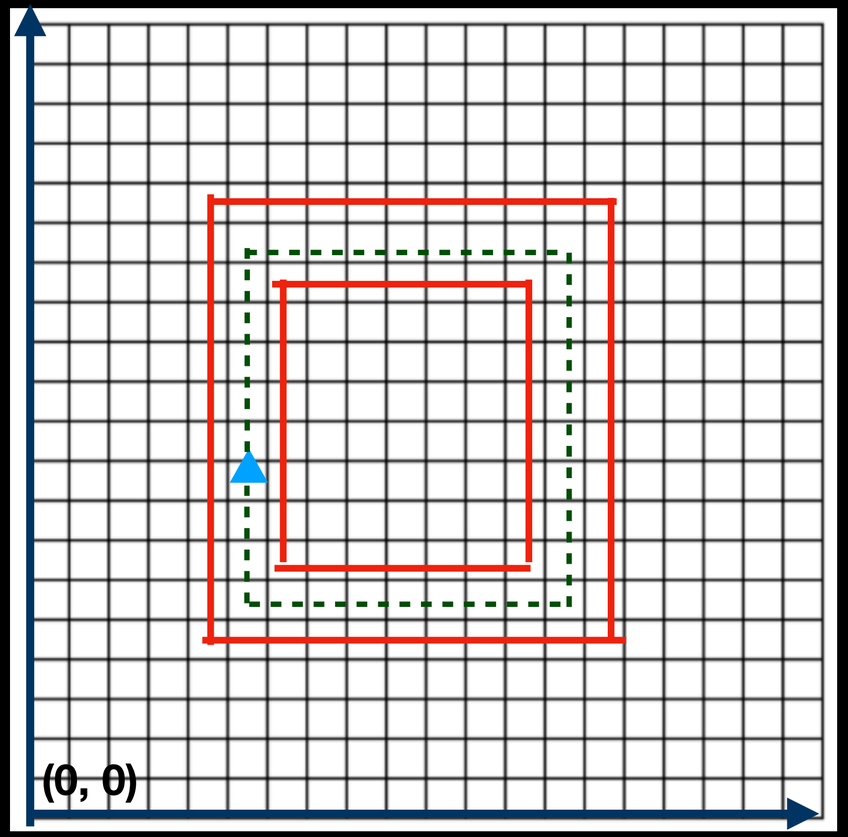


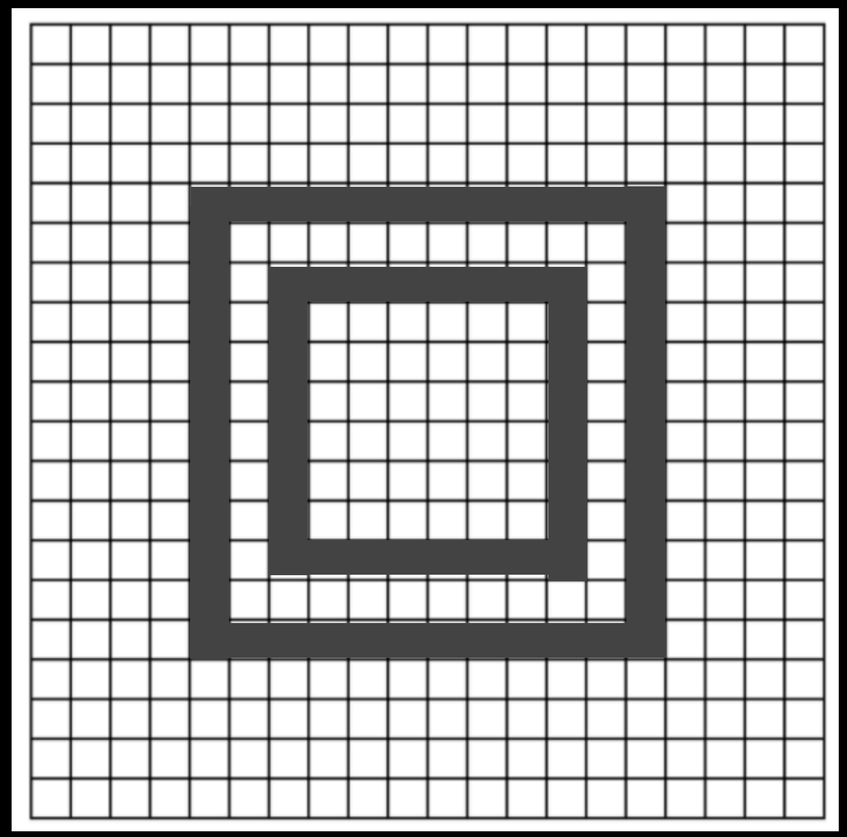
Use this map for future



How do we know this is a good map?

Ground truth and Loop closure





Not practical to manually verify generated maps
How do we know the built map reflects reality?

- Recognizing robot came back to same location => Loop closure

- Example: Consider robot moving around an annular square room
- SLAM - alternatively finds the new position of the robot and updates the map of the room
- Loop closed => Estimate of final robot position is same as the starting position
- Map is consistent with the ground reality

Probabilistic Occupancy grid

Occupancy probability of grid cell i

$$m^i \in \{0,1\}$$

Belief map

$$bel_t(m^i) = p(m^i \mid z_t)$$

$bel(m^i)$ = occupancy belief of cell i

z_t = sensor measurements

Use multiple measurements to build a robust belief map

Recursive Bayesian update:

$$bel_t(m^i) = \eta bel_{t-1}(m^i) p(z_t \mid m_i)$$

Previous Belief

Measurement model

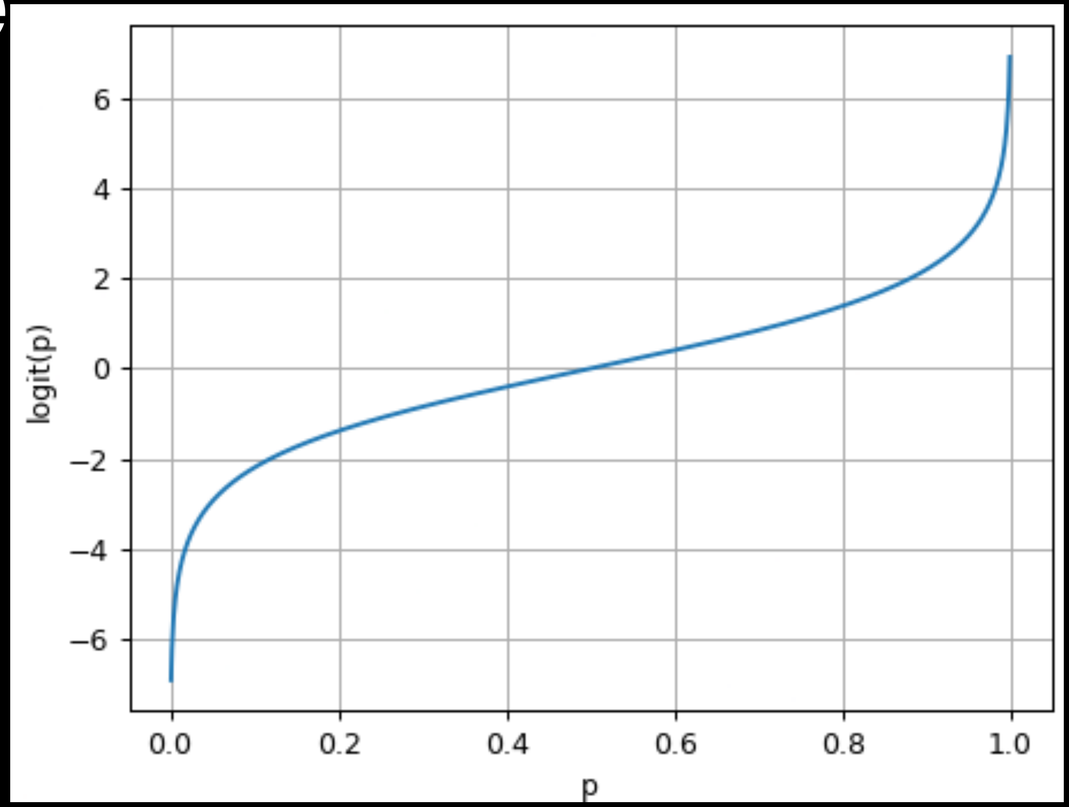
Key assumptions

- Cells are independent
- Static environment
- Robot position is known

Mapping: Log-odds update

- Floating point problems with repeated multiplication
- Move to log-odds (logit) transformation

$$logit(p) = log\left(\frac{p}{1-p}\right)$$

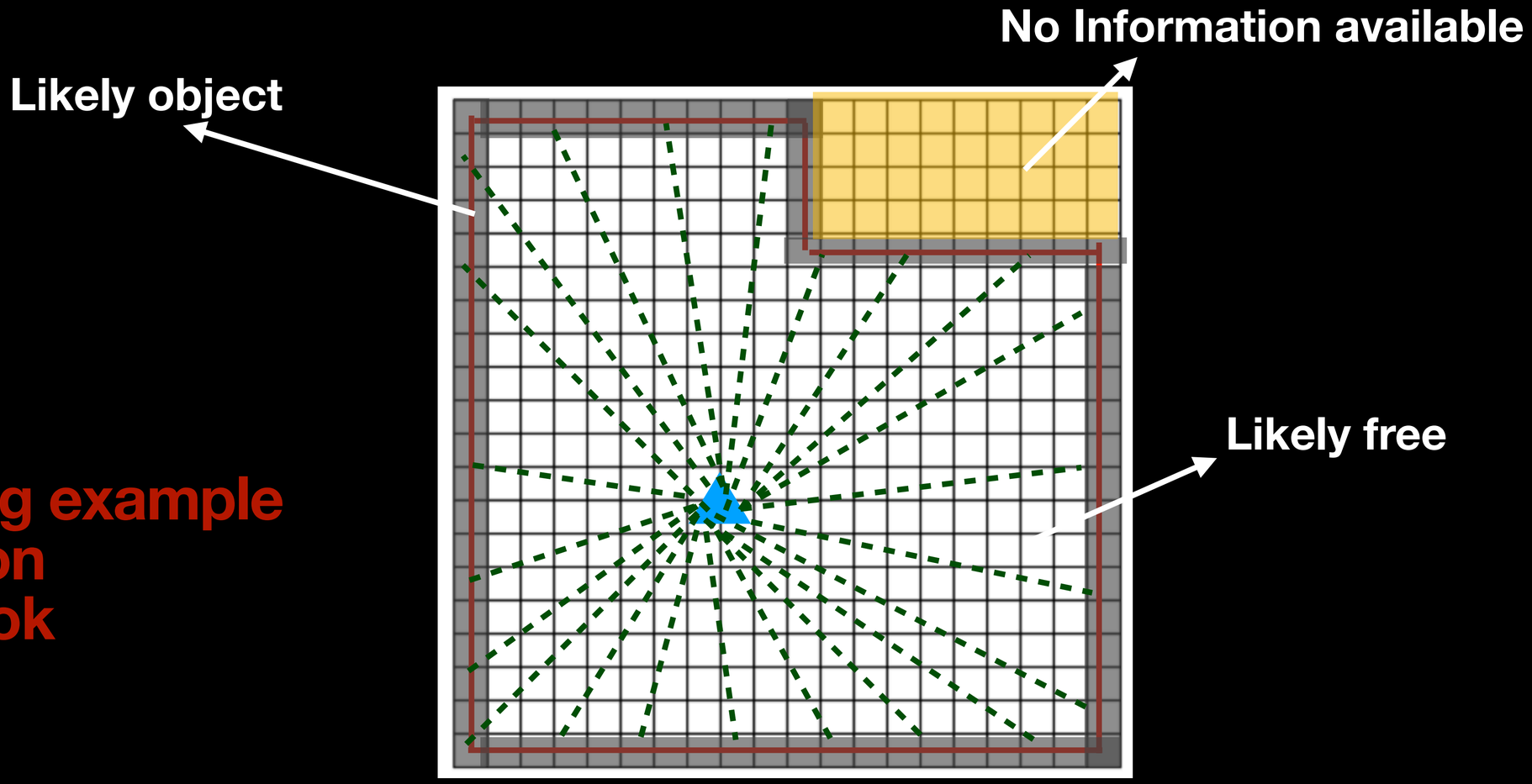


$$l_i(t) = logit(p(m^i \mid z_t)) + l_i(t-1)$$

Log-odds of occupancy belief at time t

Inverse measurement model of Lidar

Inverse measurement model



Mapping example in python notebook

3 possible regions from Lidar measurements