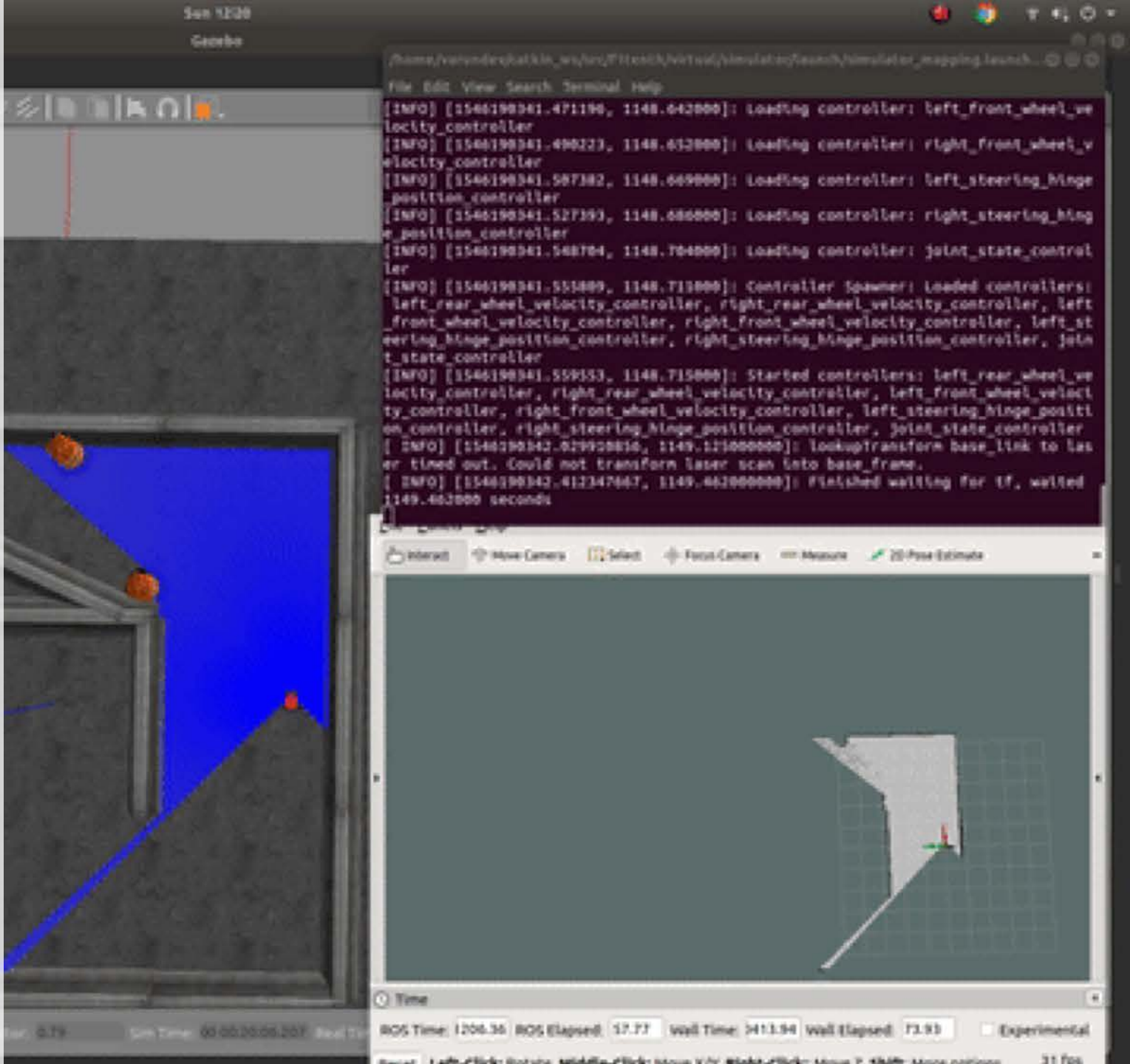
F1/10
Autonomous Racing

Simultaneous Localization And Mapping (SLAM)

Prof. Madhur Behl
Computer Science

CS 4501
University of Virginia

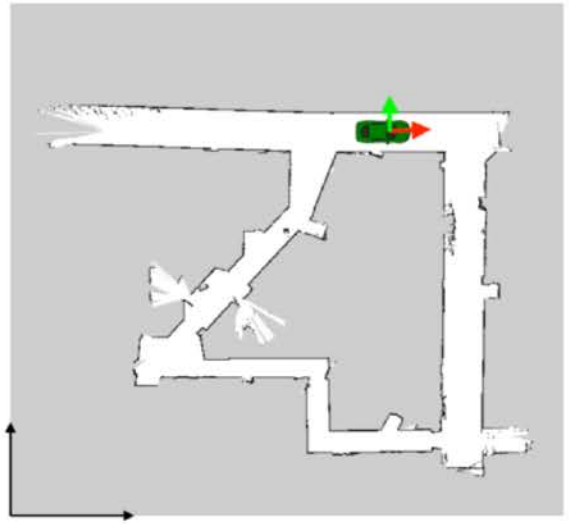


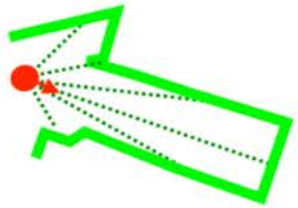
Hector SLAM

Objectives

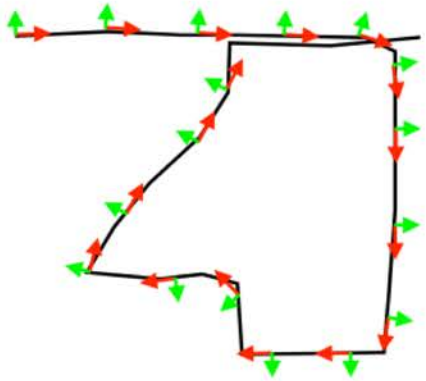
- What is Simultaneous Localization and Mapping ?
- What are Occupancy Maps ?
- How are SLAM and Scan Matching related ?
- SLAM in ROS – Hector Mapping
- Google Cartographer SLAM Overview

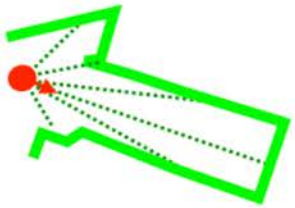
Problem Setting



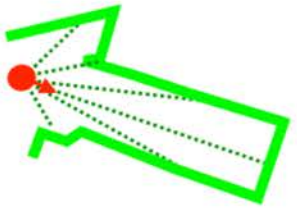


Localization: given a ***map***, use *sensor data* to estimate the current *pose* of the robot





Mapping: given robot pose at each time (**trajectory**), use *sensor data* to build map



Simultaneous Localization and Mapping (SLAM): use sensor data to build map and estimate robot trajectory

A brief history of SLAM

Why do we need a map?

- In order to support path planning
- Limiting the error in state estimates, by providing the opportunity to 'reset'
- Later… do we really need a map?

Historical Development (1986-2004): *Probabilistic Foundations*

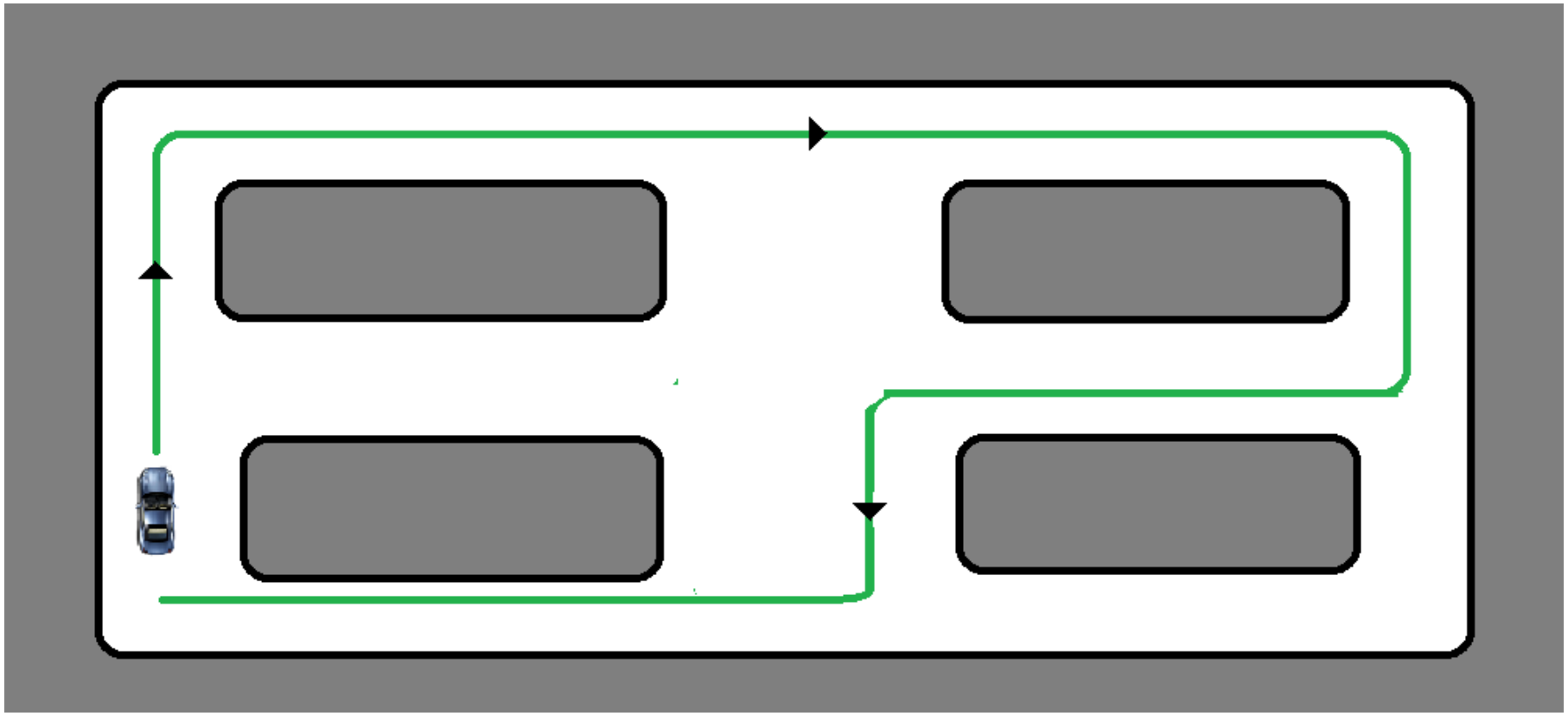
- EKF (you will still find this in visual inertial odometry)
- Particle Filter (very efficient localization)
- We will cover these methods next class in the context of localization

Modern Era (2004-Now): *Algorithmic Improvements*

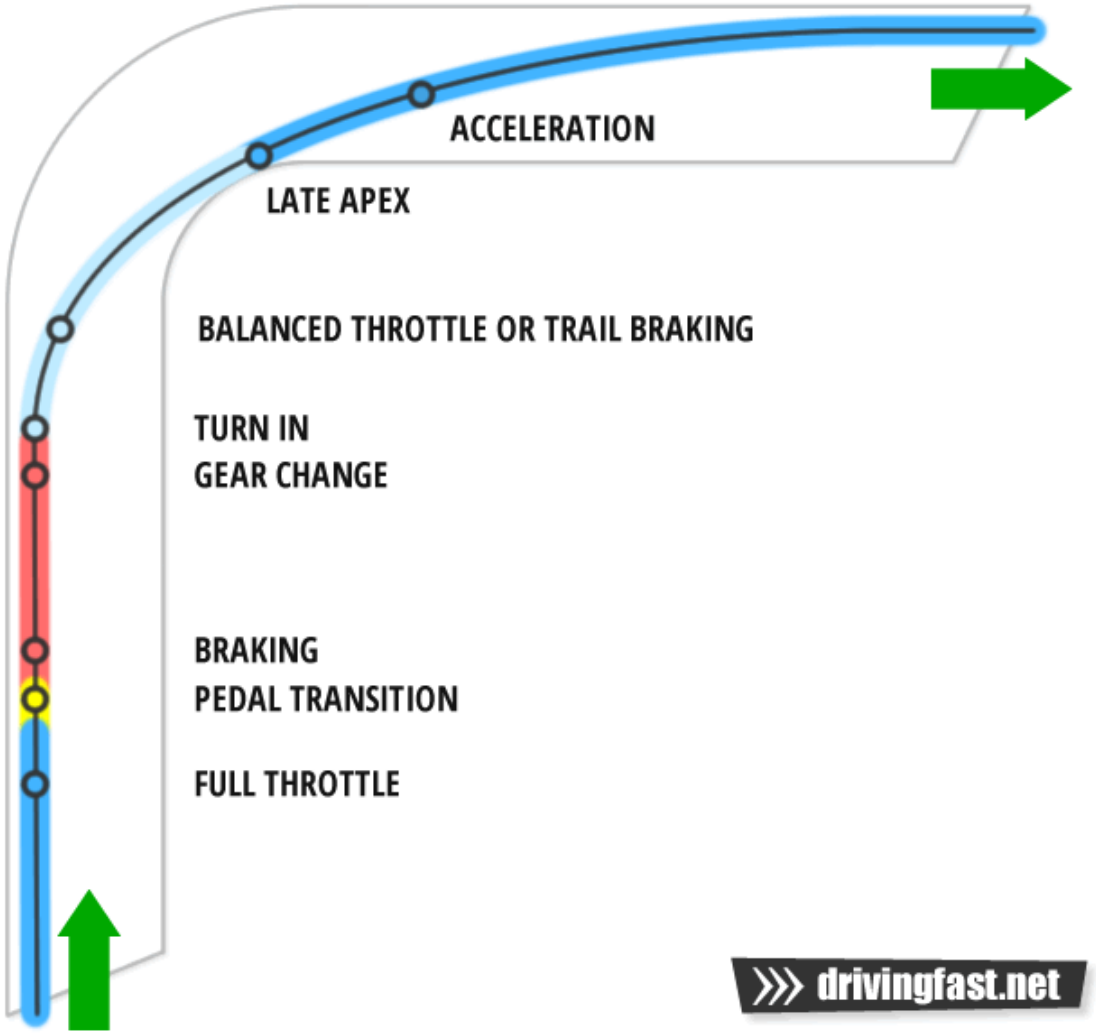
- Maximum a-posteriori Estimation
- Other names: factor graph optimization, graph-SLAM, smoothing and mapping (SAM), bundle adjustment

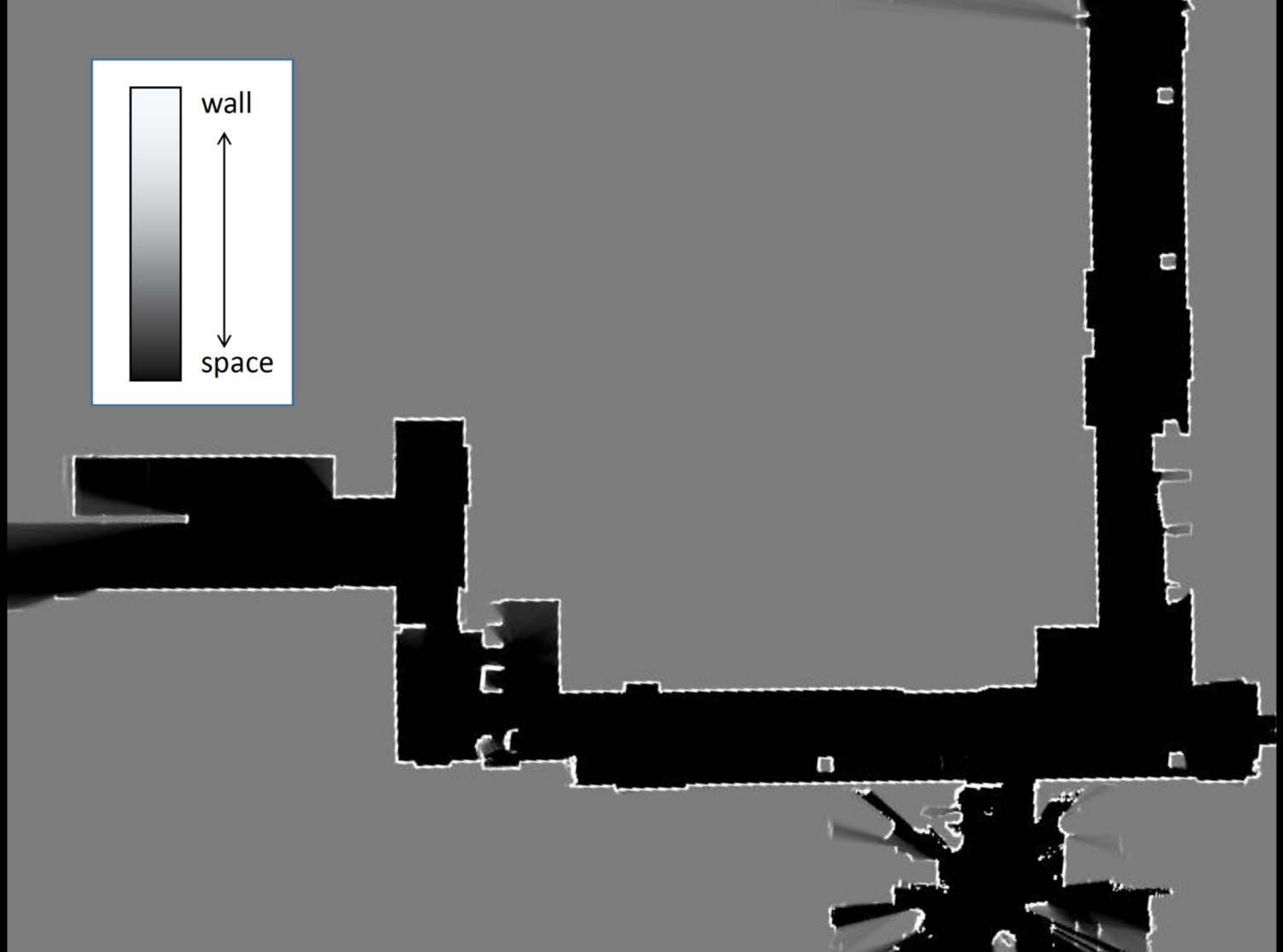
Limitations : Basic Path Planning

- High Level Path Assignments
  - 2nd right, 2nd right, 1st right, 1st left, 1st right



Race Lines


ACCELERATION
LATE APEX
BALANCED THROTTLE OR TRAIL BRAKING
TURN IN
GEAR CHANGE
BRAKING
PEDAL TRANSITION
FULL THROTTLE
drivingfast.net


wall
space

Occupancy Grid Mapping

- Occupancy: binary R.V.

$m_{x,y}: \{free, occupied\} \rightarrow \{\, 0\,, 1\,\}$

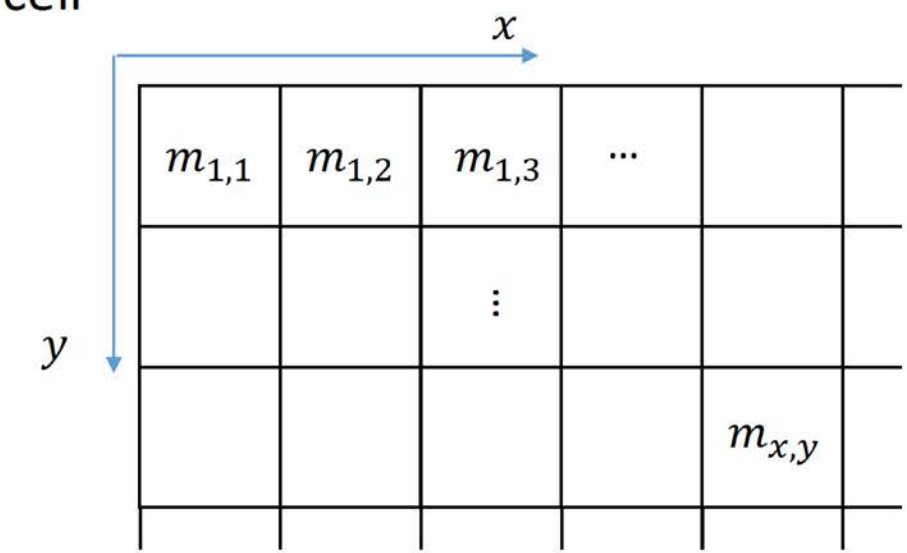
[Review – Into Probability]
Given some probability space (Ω, P),
a **random variable** $X: \Omega \rightarrow R$ is a *function* that maps the sample space to the reals.

Occupancy Grid Mapping

- Occupancy: binary R.V.

$$m_{x,y}: \{free, occupied\} \rightarrow \{\, 0\,, 1\,\}$$

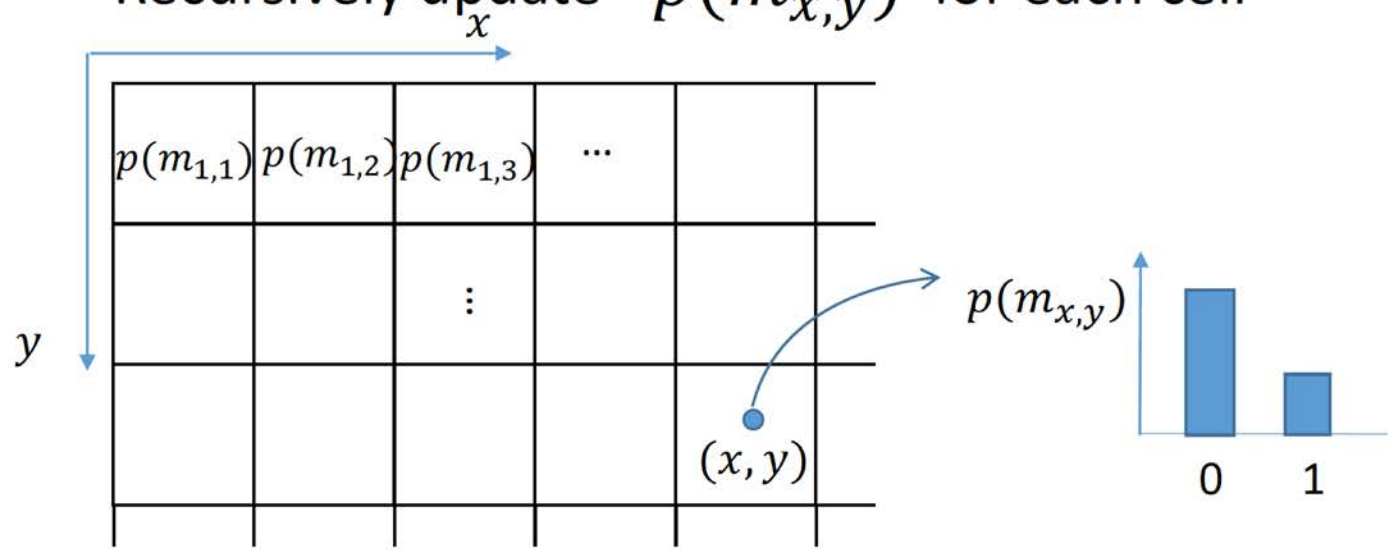
- Occupancy grid map
  : fine-grained grid map where an occupancy variable associated with each cell



Occupancy Grid Mapping

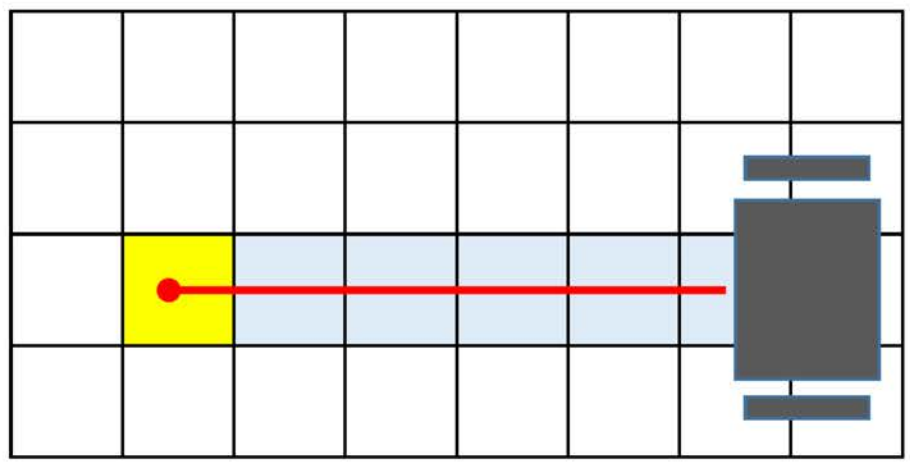
- Occupancy grid mapping
  : A Bayesian filtering to maintain a occupancy grid map.

Recursively update $p(m_{x,y})$ for each cell



Occupancy Grid Mapping

- Measurement



a range sensor

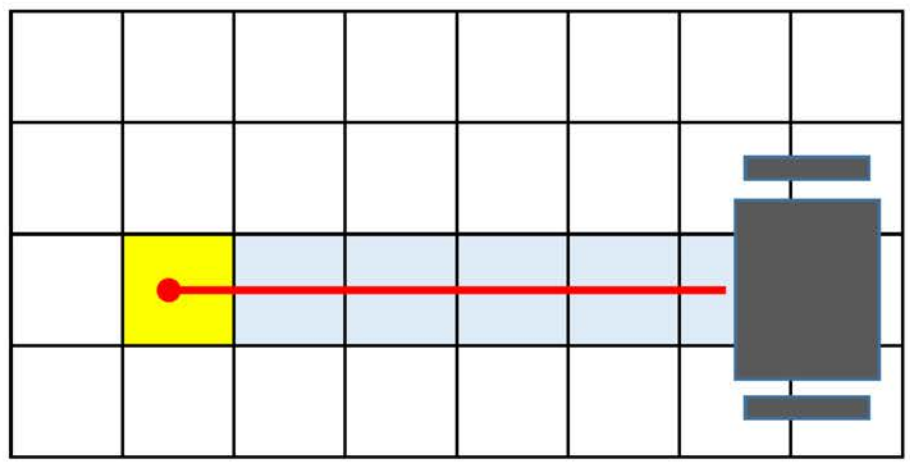
Occupancy Grid Mapping

- Measurement

Free

Occupied

$$z \sim \{0, 1\}$$



a range sensor

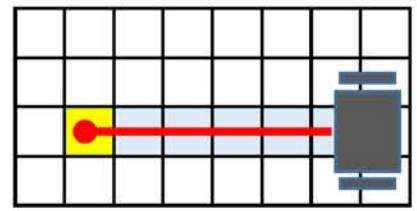
Occupancy Grid Mapping

- Measurement

Free

Occupied

$$z \sim \{0, 1\}$$



- Measurement model

$$p(z|m_{x,y})$$

$p(z = 1|m_{x,y} = 1)$: True **occupied** measurement

$p(z = 0|m_{x,y} = 1)$: False **free** measurement

$p(z = 1|m_{x,y} = 0)$: False **occupied** measurement

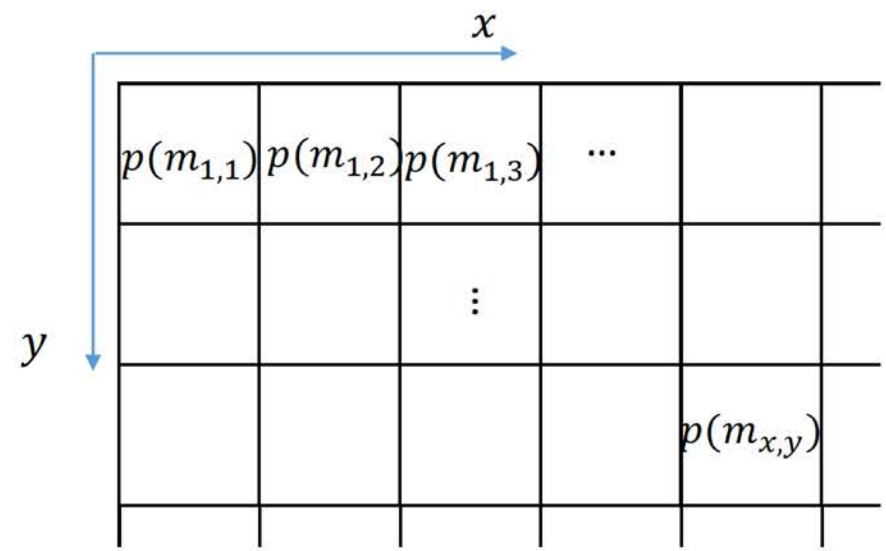
$p(z = 0|m_{x,y} = 0)$: True **free** measurement

Occupancy Grid Mapping

Measurement Model

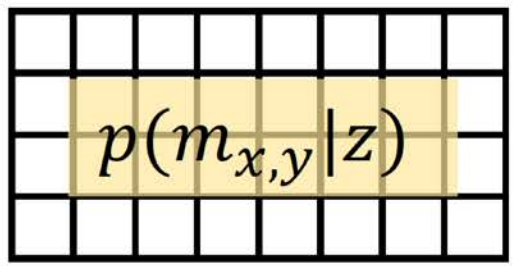
$$p(z|m_{x,y})$$

Map



Occupancy Grid Mapping

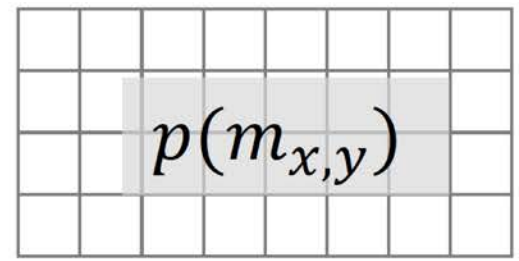
Posterior Map



Measurement Model

$p(z|m_{x,y})$

Prior Map



Likelihood Prior

Posterior

$$p(m_{x,y}|z) = \frac{p(z|m_{x,y})p(m_{x,y})}{p(z)}$$

Evidence

Occupancy Grid Mapping

$$Odd := \frac{(X\ happens)}{(X\ not\ happens)} = \frac{p(X)}{p(X^c)}$$

- More convenient when we use “Odd”

$$Odd((m_{x,y}= 1)\ given\ z) = \frac{p(m_{x,y} = 1|z)}{p(m_{x,y} = 0|z)}$$

Occupancy Grid Mapping

- Odd

(Bayes' Rule)

$$p(m_{x,y}=1|z)=\frac{p(z|m_{x,y}=1)p(m_{x,y}=1)}{p(z)}$$

$$Odd=\frac{p(m_{x,y}=1|z)}{p(m_{x,y}=0|z)}=\frac{p(z|m_{x,y}=1)p(m_{x,y}=1)/p(z)}{p(m_{x,y}=0|z)}$$

Occupancy Grid Mapping

- Odd

$$Odd = \frac{p(m_{x,y} = 1|z)}{p(m_{x,y} = 0|z)} = \frac{p(z|m_{x,y} = 1)p(m_{x,y} = 1)/\cancel{p(z)}}{p(z|m_{x,y} = 0)p(m_{x,y} = 0)/\cancel{p(z)}}$$

$$p(m_{x,y} = 0|z) = \frac{p(z|m_{x,y} = 0)p(m_{x,y} = 0)}{p(z)}$$

(Bayes' Rule)

Occupancy Grid Mapping

- Take the log!

Odd:
$$\frac{p(m_{x,y}=1|z)}{p(m_{x,y}=0|z)}=\frac{p(z|m_{x,y}=1)p(m_{x,y}=1)}{p(z|m_{x,y}=0)p(m_{x,y}=0)}$$

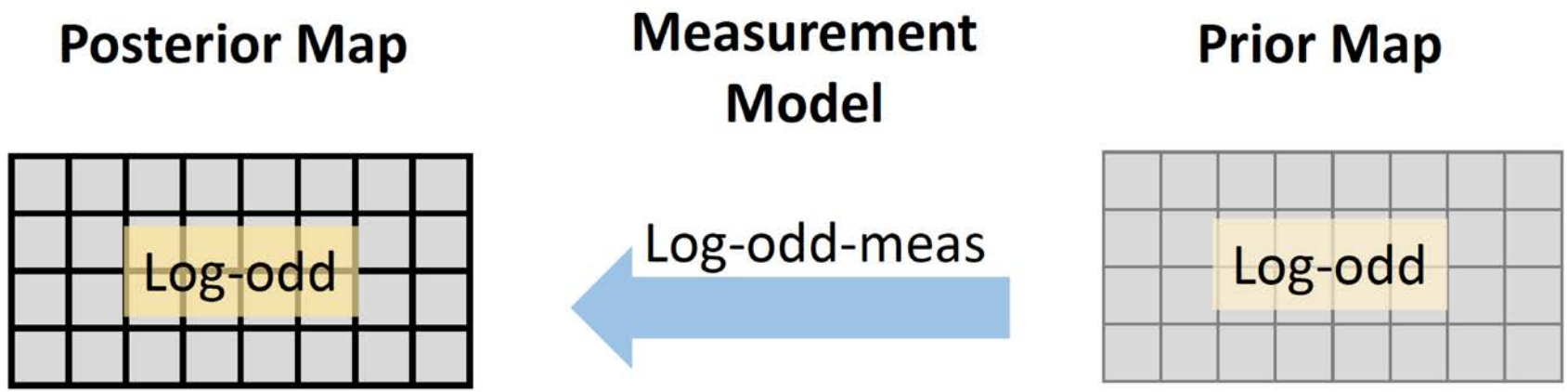
Log-Odd:
$$\log\frac{p(m_{x,y}=1|z)}{p(m_{x,y}=0|z)}=\log\frac{p(z|m_{x,y}=1)p(m_{x,y}=1)}{p(z|m_{x,y}=0)p(m_{x,y}=0)}$$

$$=\log\frac{p\left(z|m_{x,y}=1\right)}{p\left(z|m_{x,y}=0\right)}+\log\frac{p\left(m_{x,y}=1\right)}{p\left(m_{x,y}=0\right)}$$

$$\log odd^{+}=\log odd\ meas+\log odd^{-}$$

Occupancy Grid Mapping

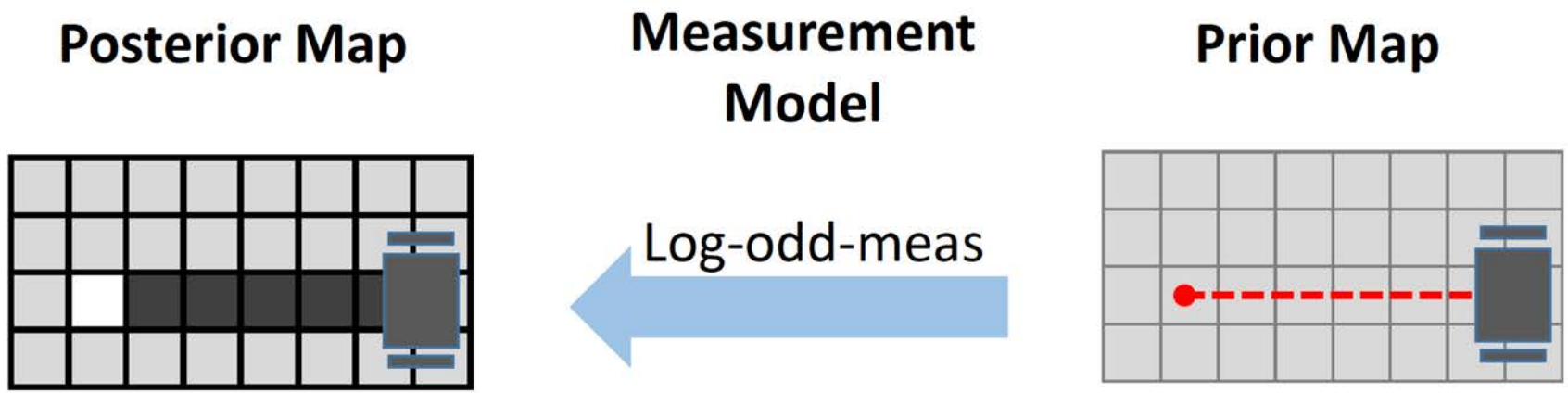
- Log-odd update



$$\log odd^{+} = \log odd\ meas + \log odd^{-}$$

Occupancy Grid Mapping

- Log-odd update



$$\log odd^{+} = \log odd\ meas + \log odd^{-}$$

Occupancy Grid Mapping

- Measurement model in log-odd form

$$\log \frac{p(z|m_{x,y}=1)}{p(z|m_{x,y}=0)}$$

- **Two possible measurement:**

Case I : cells with z=1

$$\log odd\_occ := \log \frac{p(z=1|m_{x,y}=1)}{p(z=1|m_{x,y}=0)}$$

Case II : cells with z=0

$$\log odd\_free := \log \frac{p(z=0|m_{x,y}=0)}{p(z=0|m_{x,y}=1)}$$

(Trivial Case : cells not measured)

Occupancy Grid Mapping

- Example

Constant Measurement Model

$\log odd\_occ := 0.9$

$\log odd\_free := 0.7$

Initial Map:

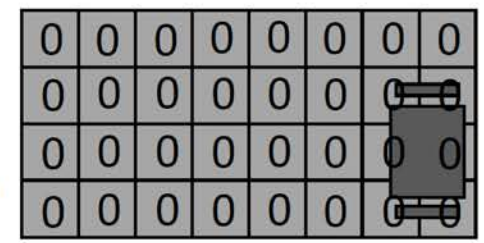
$\log odd = 0$ for all (x,y)

$p(m_{x,y} = 1) = p(m_{x,y} = 0) = 0.5$

Update Rule:

$\log odd \mathrel{+}= \log odd\_meas$

t0



Occupancy Grid Mapping

- Example

Constant Measurement Model

$\log odd\_occ := 0.9$

$\log odd\_free := 0.7$

Update

- Case I : cells with z=1

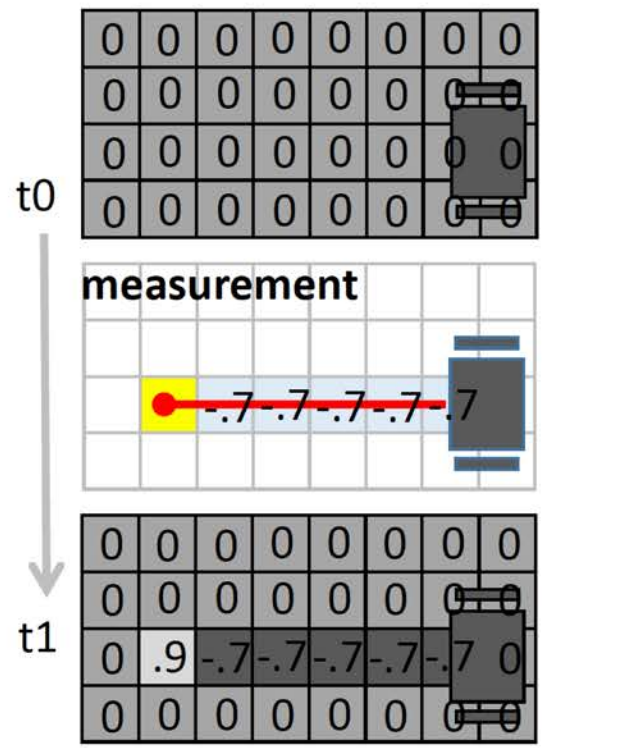
$\log odd \leftarrow 0 + \log odd\_occ$

- Case II : cells with z=0

$\log odd \leftarrow 0 - \log odd\_free$

Update Rule:

$\log odd \mathrel{+}= \log odd\_meas$



Occupancy Grid Mapping

- Example

Constant Measurement Model

$\log odd\_occ := 0.9$

$\log odd\_free := 0.7$

Update

- Case I : cells with z=1

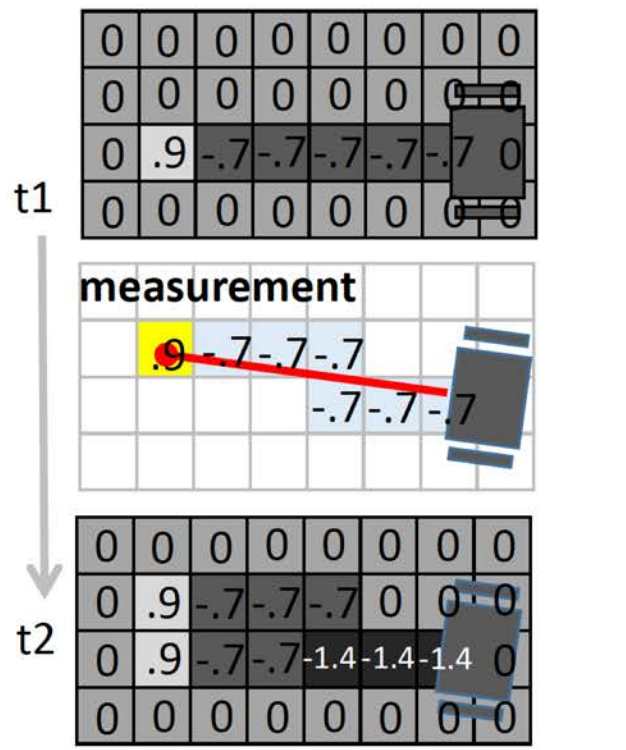
$\log odd \leftarrow 0 + \log odd\_occ$

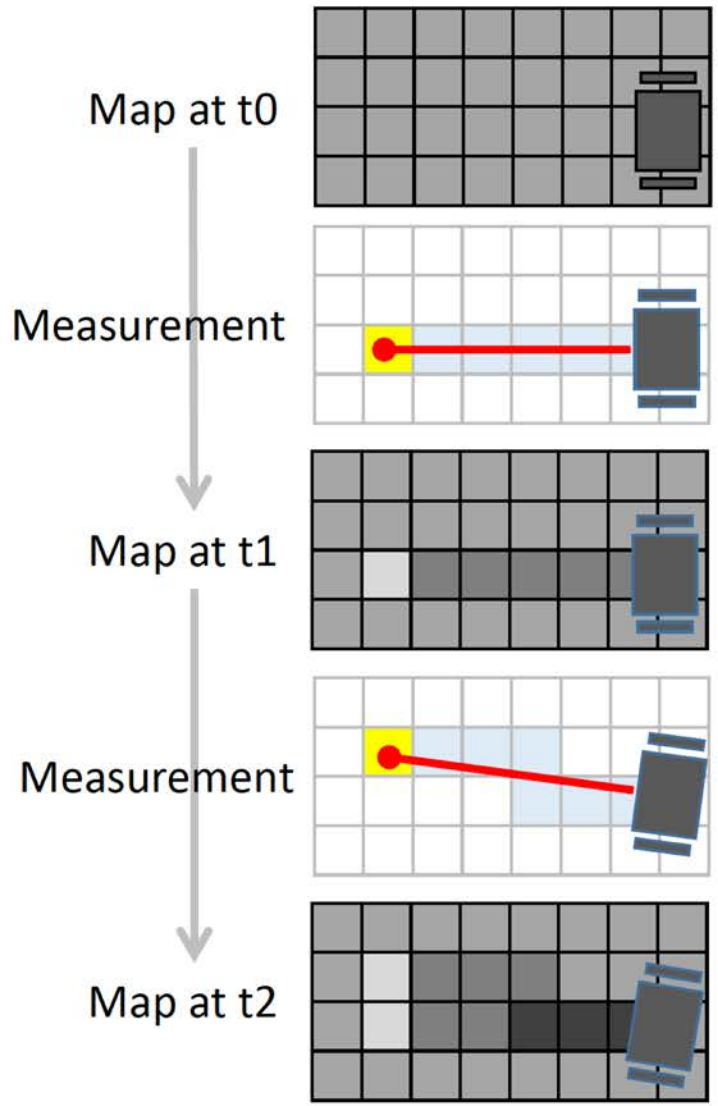
- Case II : cells with z=0

$\log odd \leftarrow 0 - \log odd\_free$

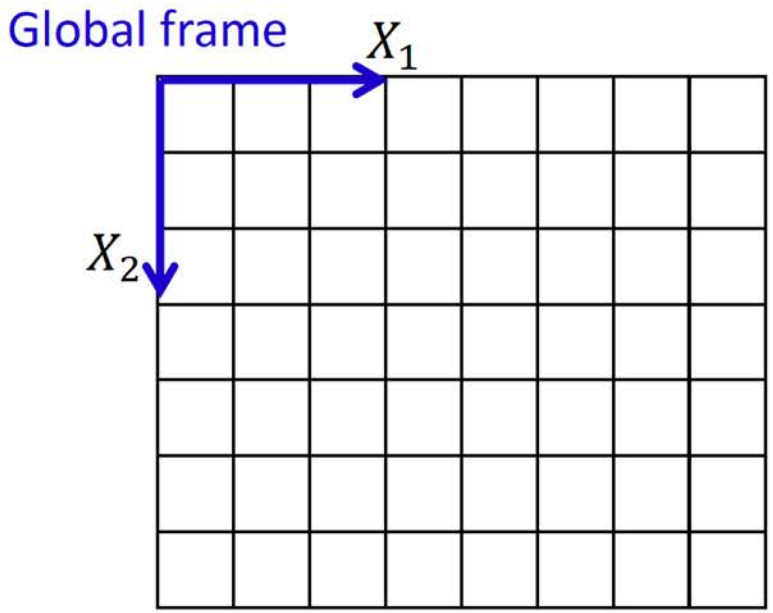
Update Rule:

$\log odd \mathrel{+}= \log odd\_meas$

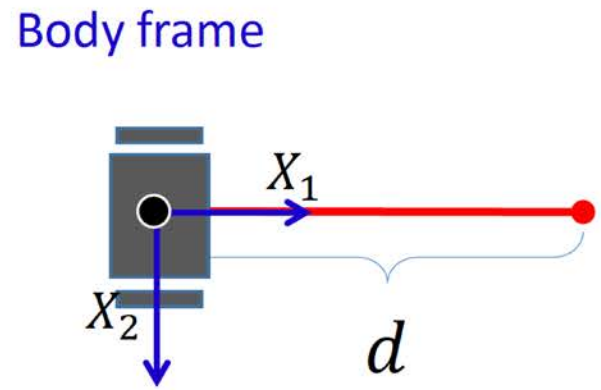



Map at t0
Measurement
Map at t1
Measurement
Map at t2

Handling Range Measurement on Grid

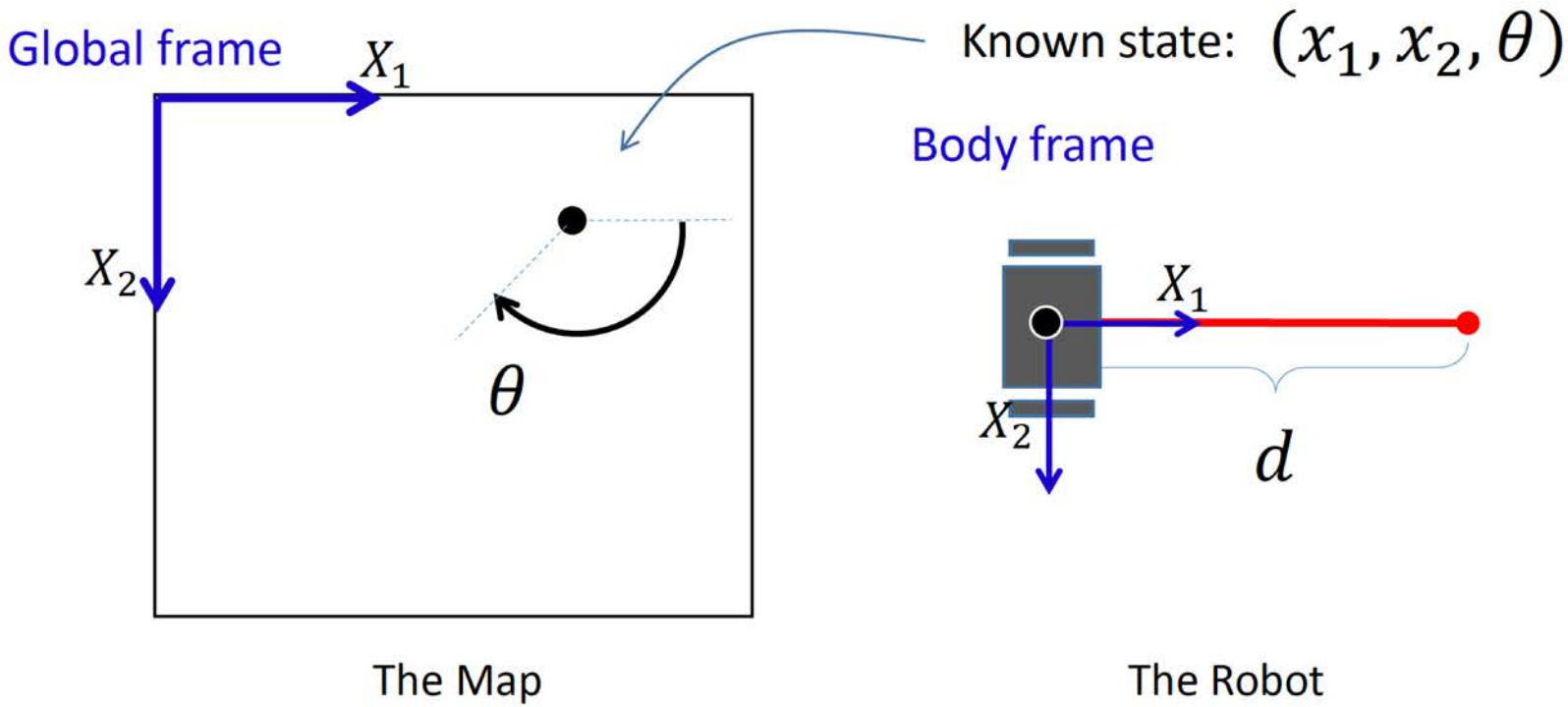


The Map

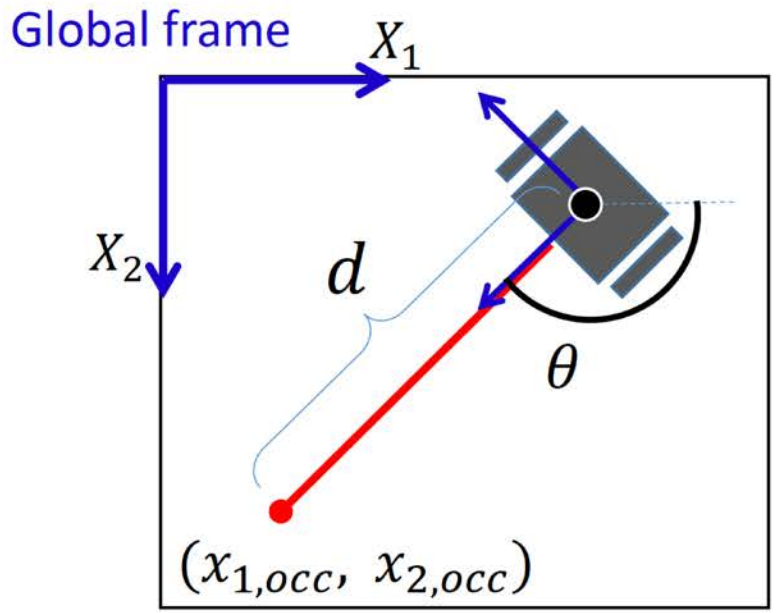


The Robot

Handling Range Measurement on Grid



Handling Range Measurement on Grid



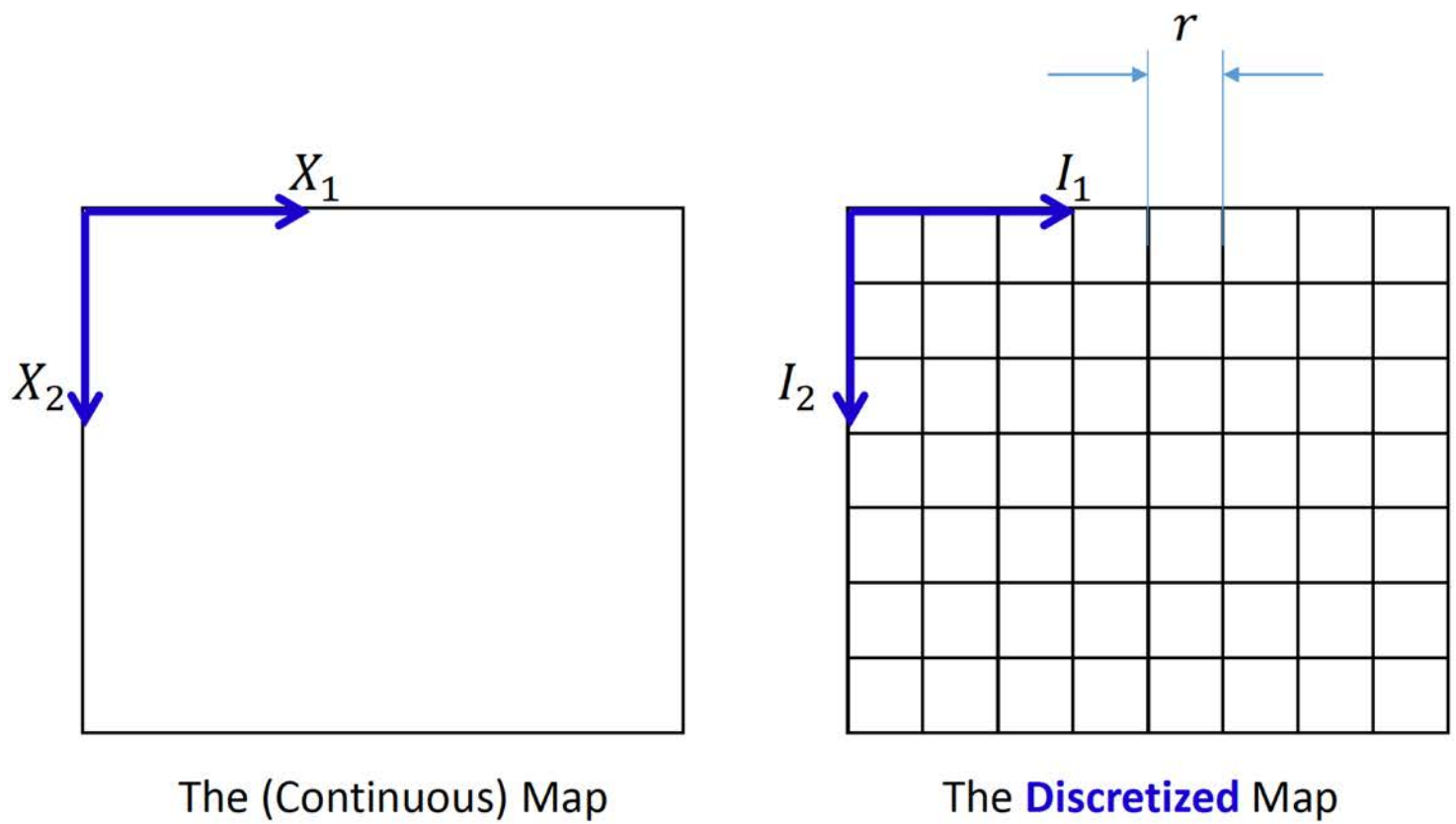
The Map

Distance measurement: d

Known state: (x_1, x_2, θ)

$$\begin{bmatrix} x_{1,occ} \\ x_{2,occ} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} d \\ 0 \end{bmatrix} + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Handling Range Measurement on Grid

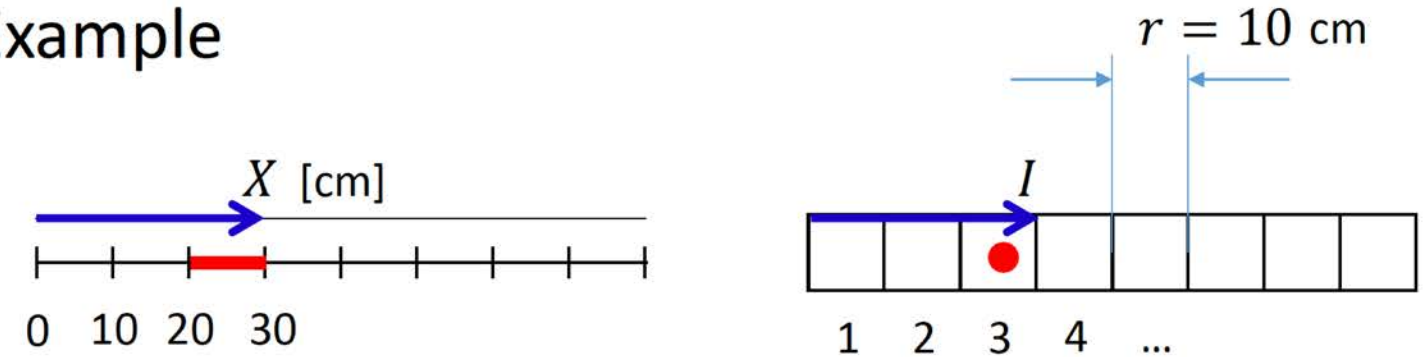


The (Continuous) Map

The **Discretized** Map

Handling Range Measurement on Grid

- Example



[cm] $0 < x \leq 10$ ⇨ $i = 1$ [index]

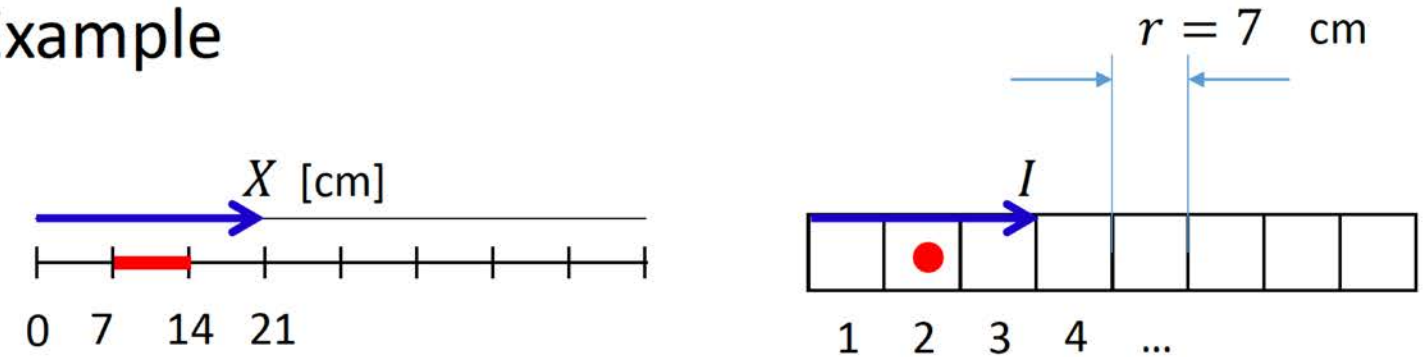
$10 < x \leq 20$ ⇨ $i = 2$

$20 < x \leq 30$ ⇨ $i = 3$

$\vdots$ $\vdots$

Handling Range Measurement on Grid

- Example

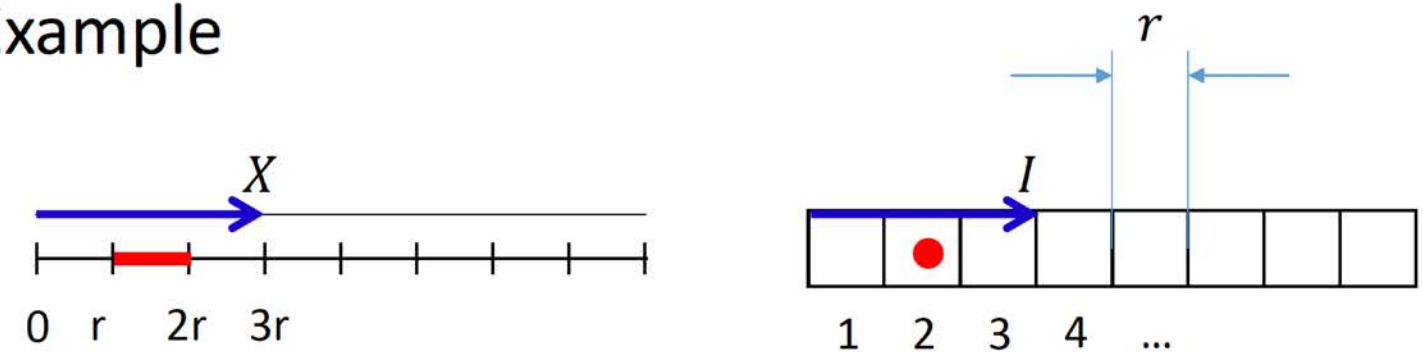


[cm] $0 < x \leq 7$ ⇨ $i = 1$ [index]

$7 < x \leq 14$ ⇨ $i = 2$

Handling Range Measurement on Grid

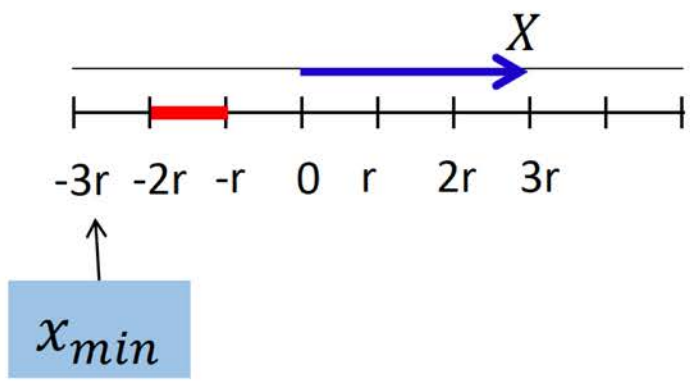
- Example

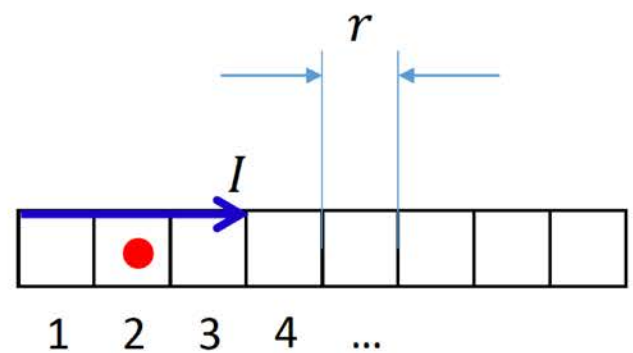


$$i = ceil(x/r)$$

Handling Range Measurement on Grid

- Example

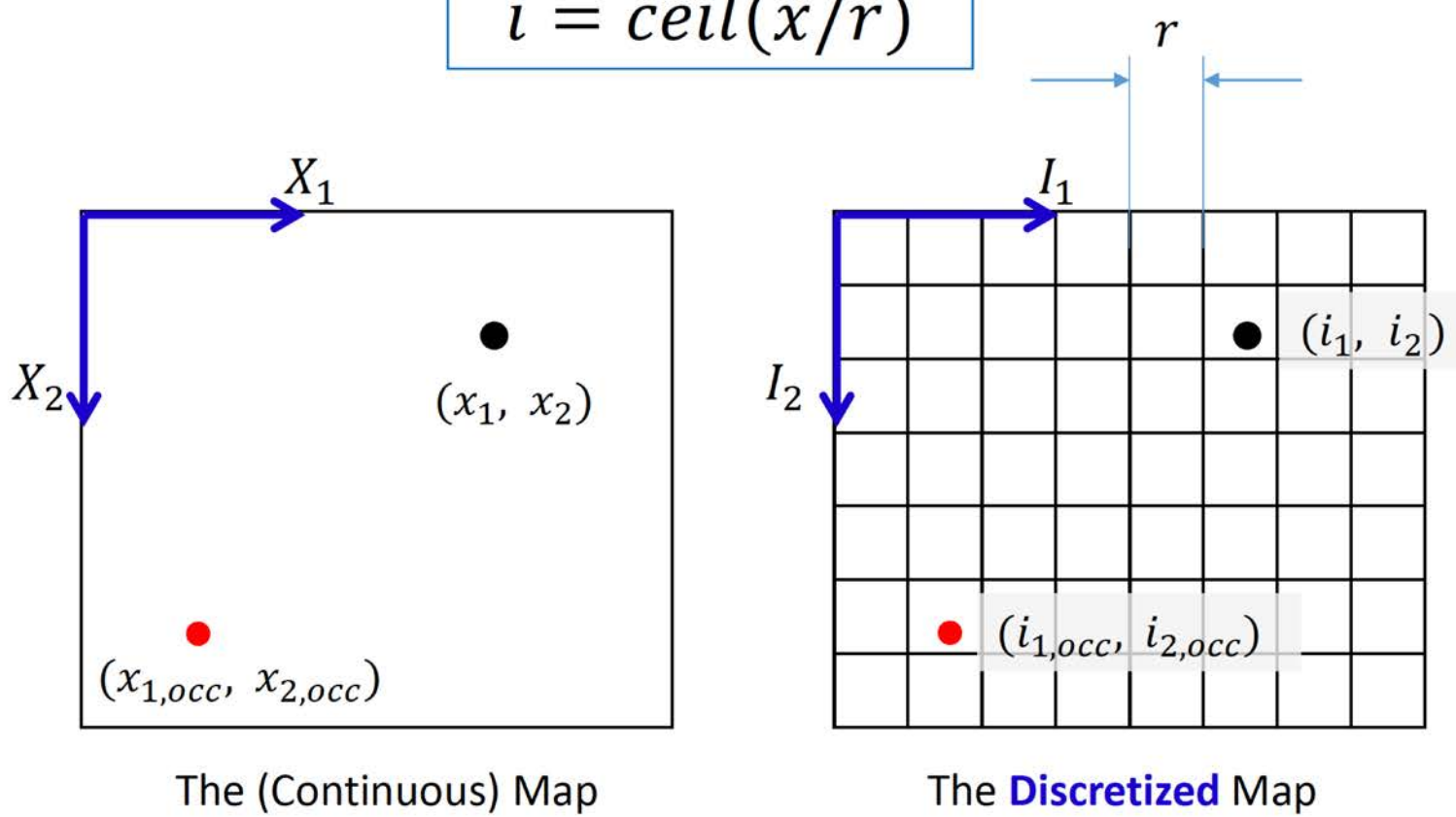




$$i = ceil((x - x_{min})/r)$$

Handling Range Measurement on Grid

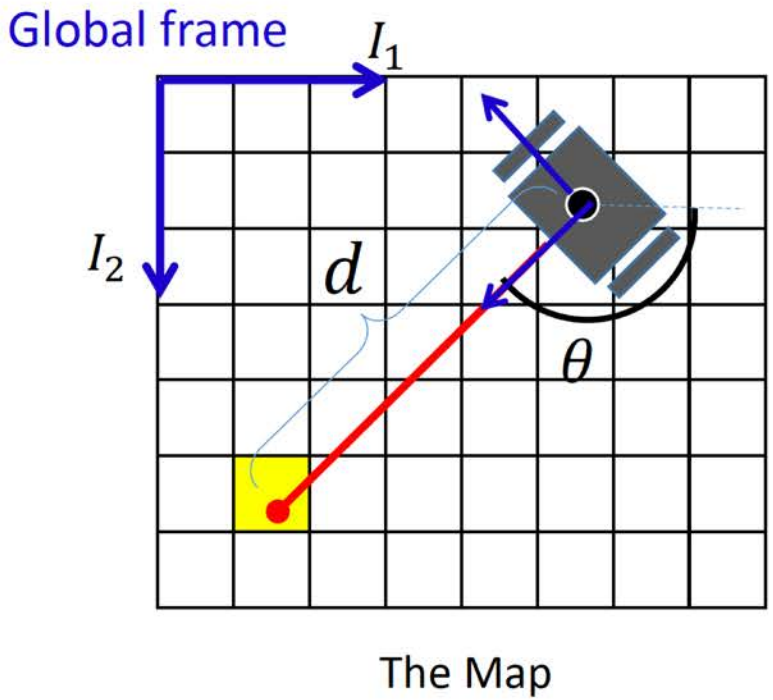
$$i = ceil(x/r)$$



The (Continuous) Map

The **Discretized** Map

Handling Range Measurement on Grid



The Map

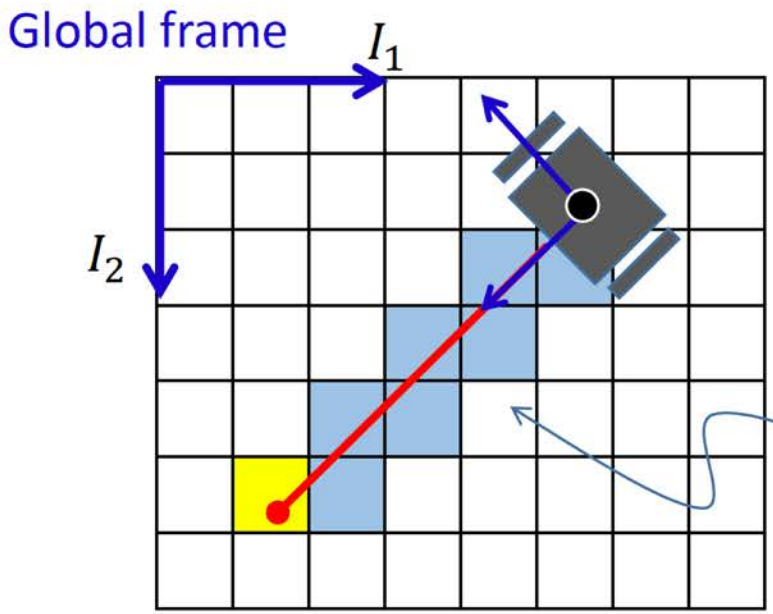
Distance measurement: d

Known state: (x_1, x_2, θ)

$$\begin{bmatrix} x_{1,occ} \\ x_{2,occ} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} d \\ 0 \end{bmatrix} + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\begin{bmatrix} i_{1,occ} \\ i_{2,occ} \end{bmatrix} = ceil\left(\frac{1}{r}\begin{bmatrix} x_{1,occ} \\ x_{2,occ} \end{bmatrix}\right)$$

Handling Range Measurement on Grid



The Map

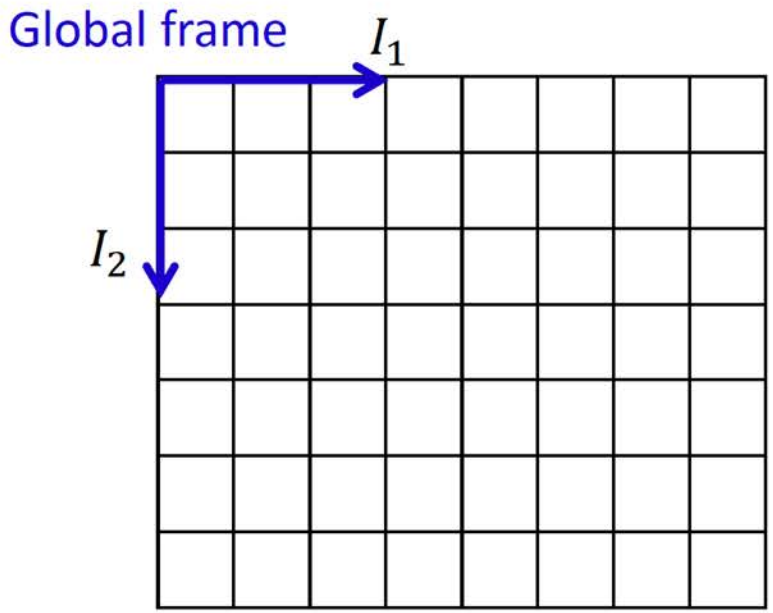
Distance measurement: d

Known state: (x_1, x_2, θ)

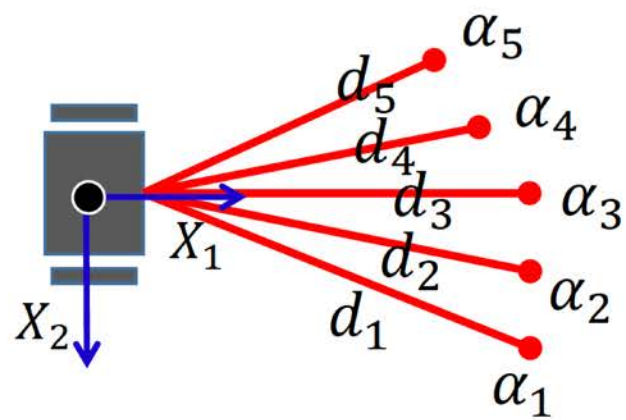
Occupied cell: $(x_{1,occ}, x_{2,occ})$

***Bresenham's line algorithm**

Handling Range Measurement on Grid

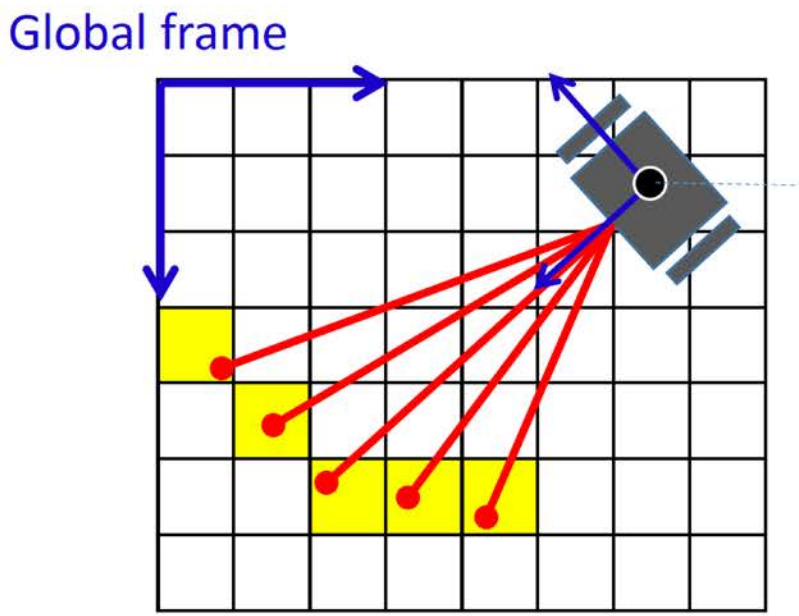


The Map



The Robot

Handling Range Measurement on Grid



The Map

Distance measurement:

$$(d_1, d_2, d_3, d_4, d_5)$$

Directions of rays:

$$(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$$

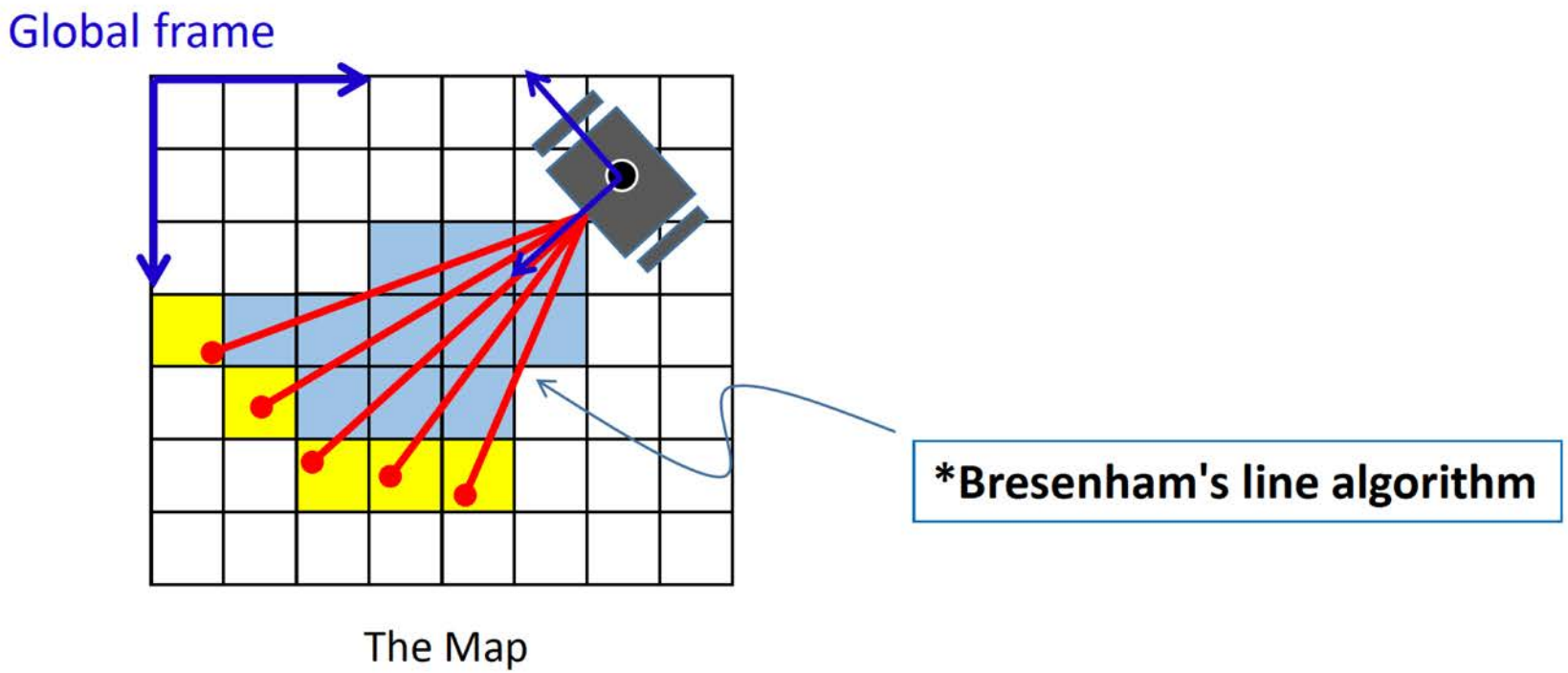
Known state: (x_1, x_2, θ)

For *k*-th occupied cell:

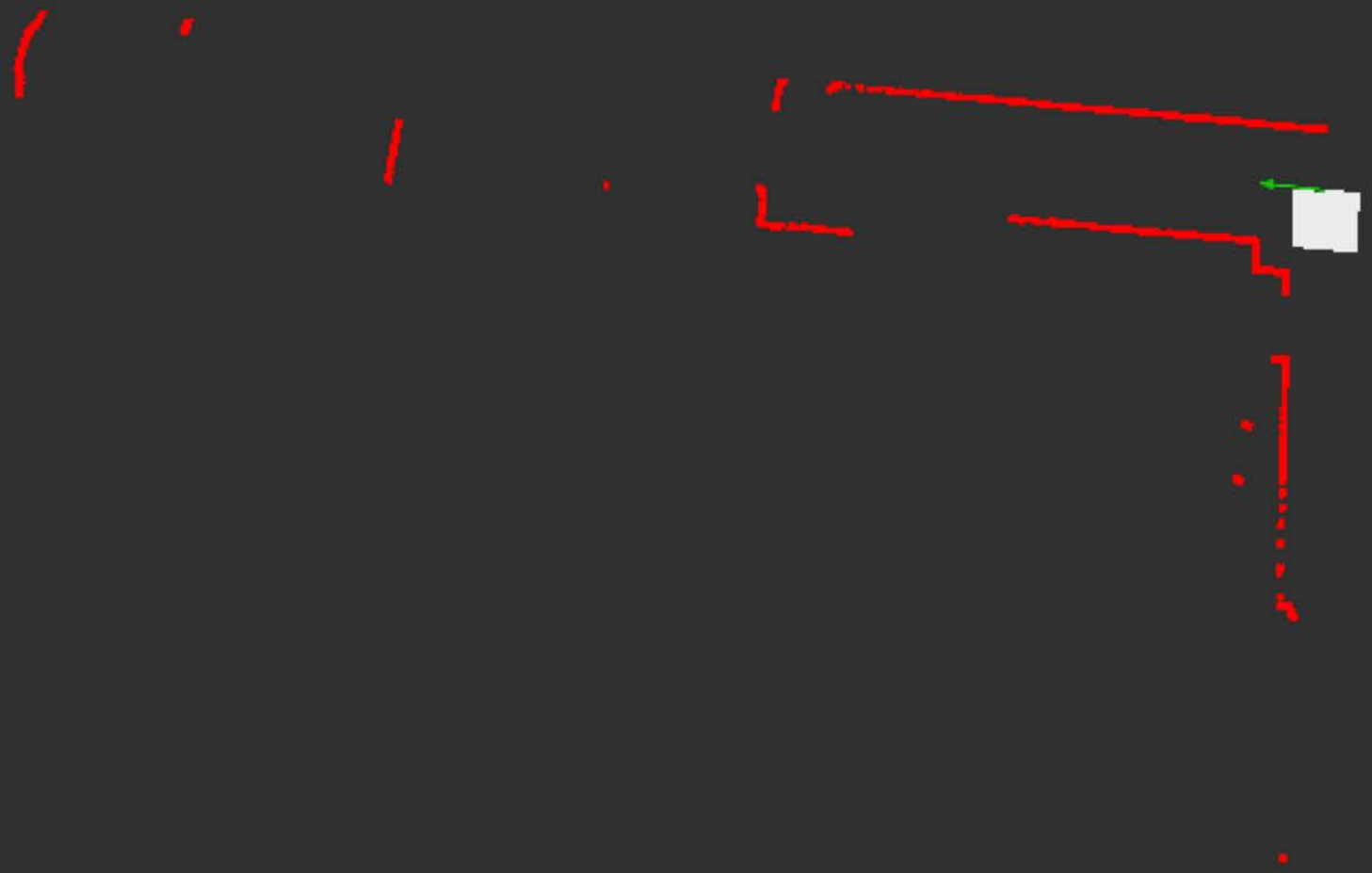
$$\begin{bmatrix} x_{1k} \\ x_{2k} \end{bmatrix} = \begin{bmatrix} d_k \cos(\theta + \alpha_k) \\ -d_k \sin(\theta + \alpha_k) \end{bmatrix} + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\begin{bmatrix} i_{1k} \\ i_{2k} \end{bmatrix} = ceil\left(\frac{1}{r}\begin{bmatrix} x_{1k} \\ x_{2k} \end{bmatrix}\right)$$

Handling Range Measurement on Grid



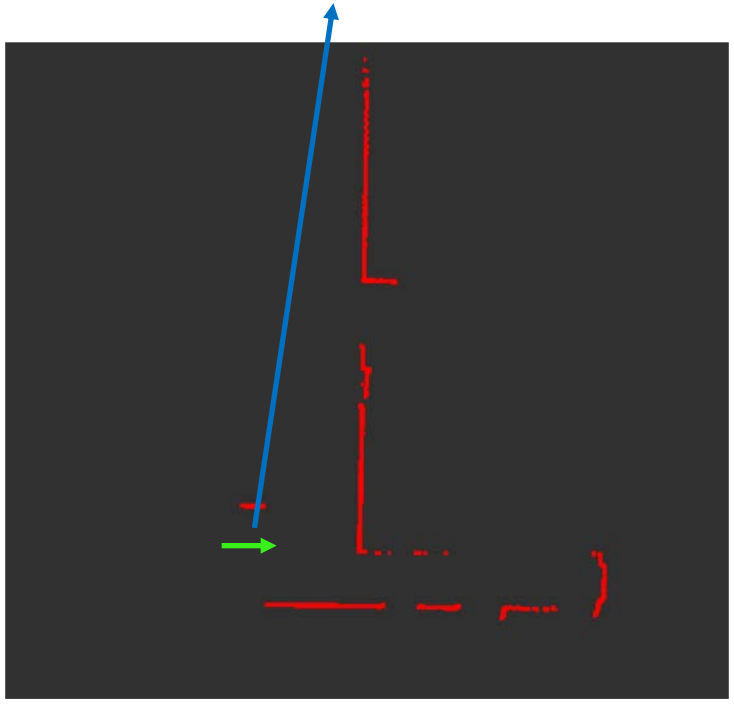
Registering the first Scan



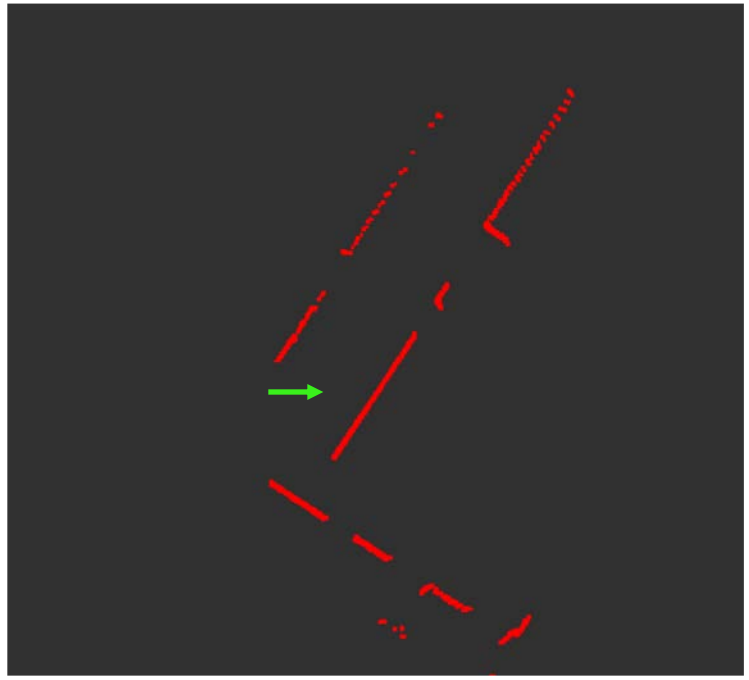
Registering the first Scan

Scan Matching

Pose of the Car at t = t1

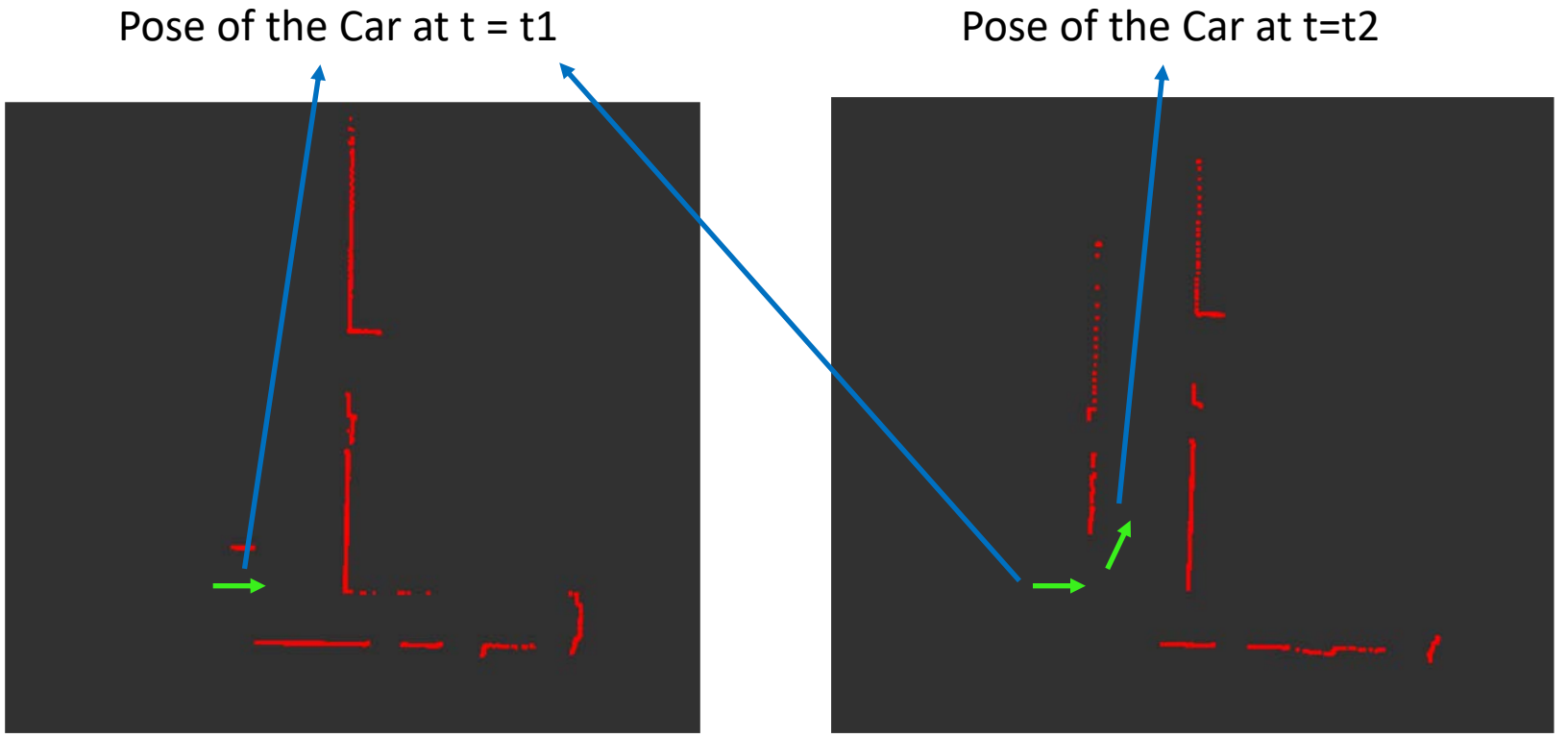


Laser Scans w.r.t car at Time t = t_1



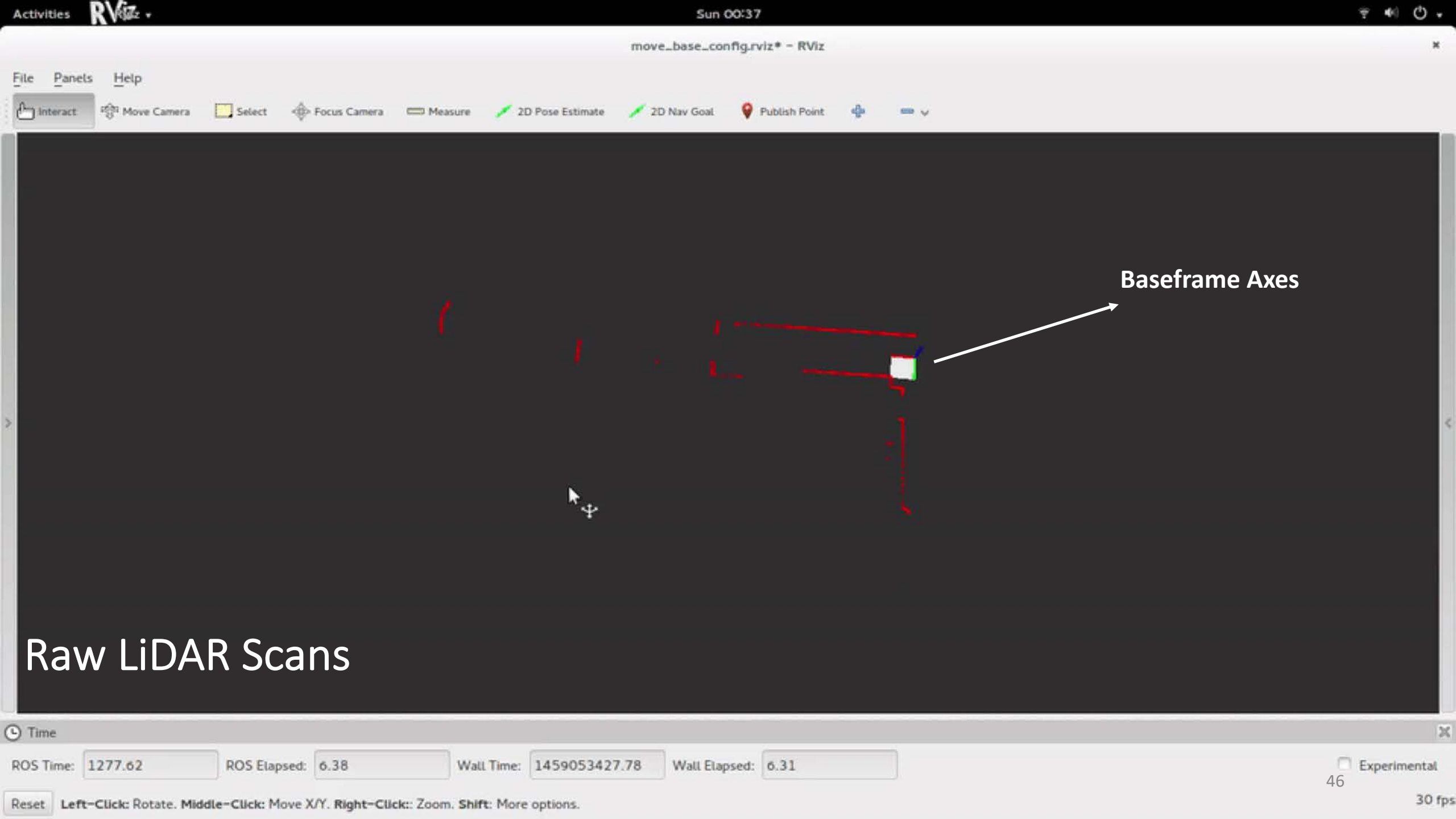
Laser Scans w.r.t car at Time t = t_2

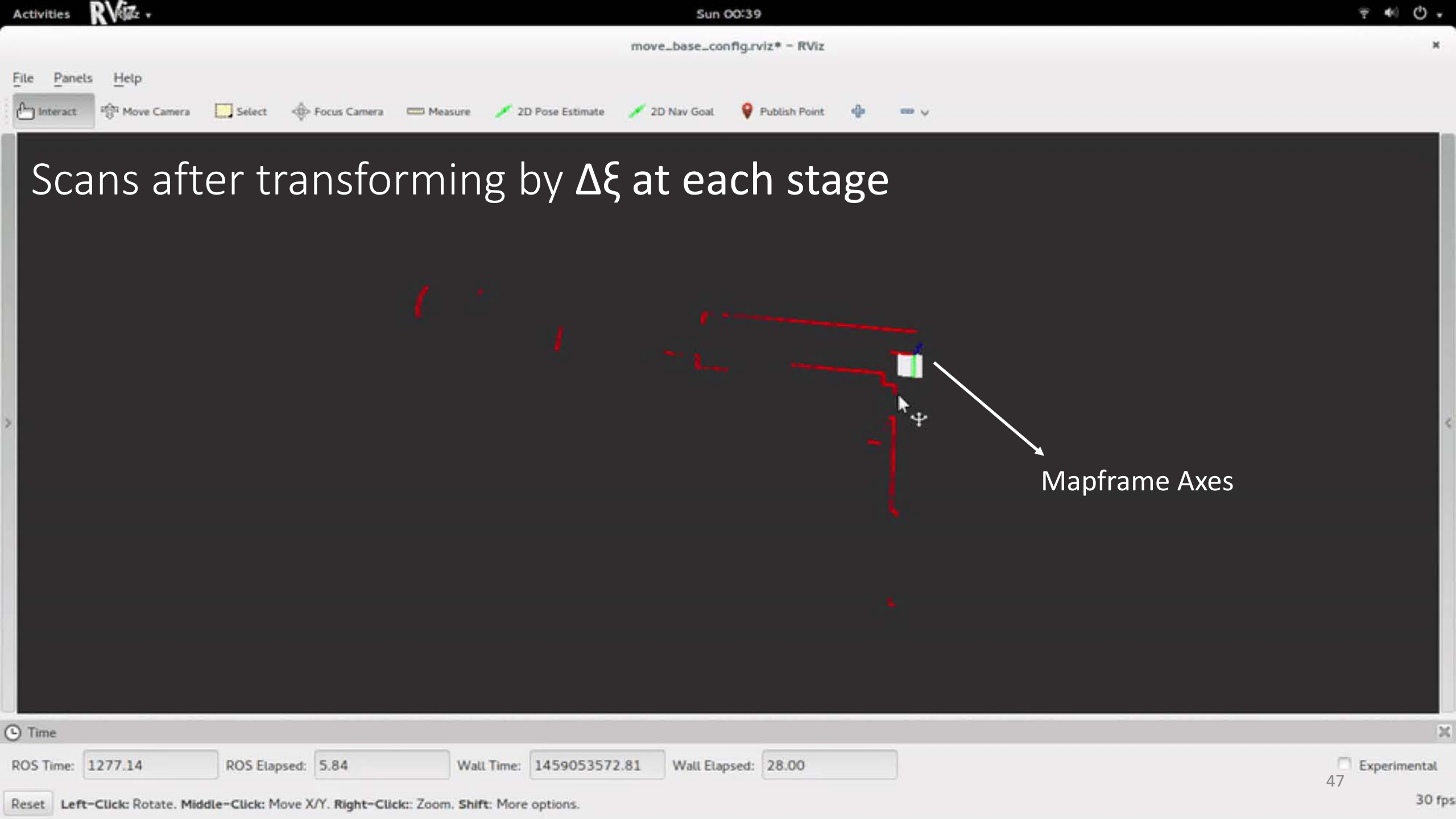
Scan Matching

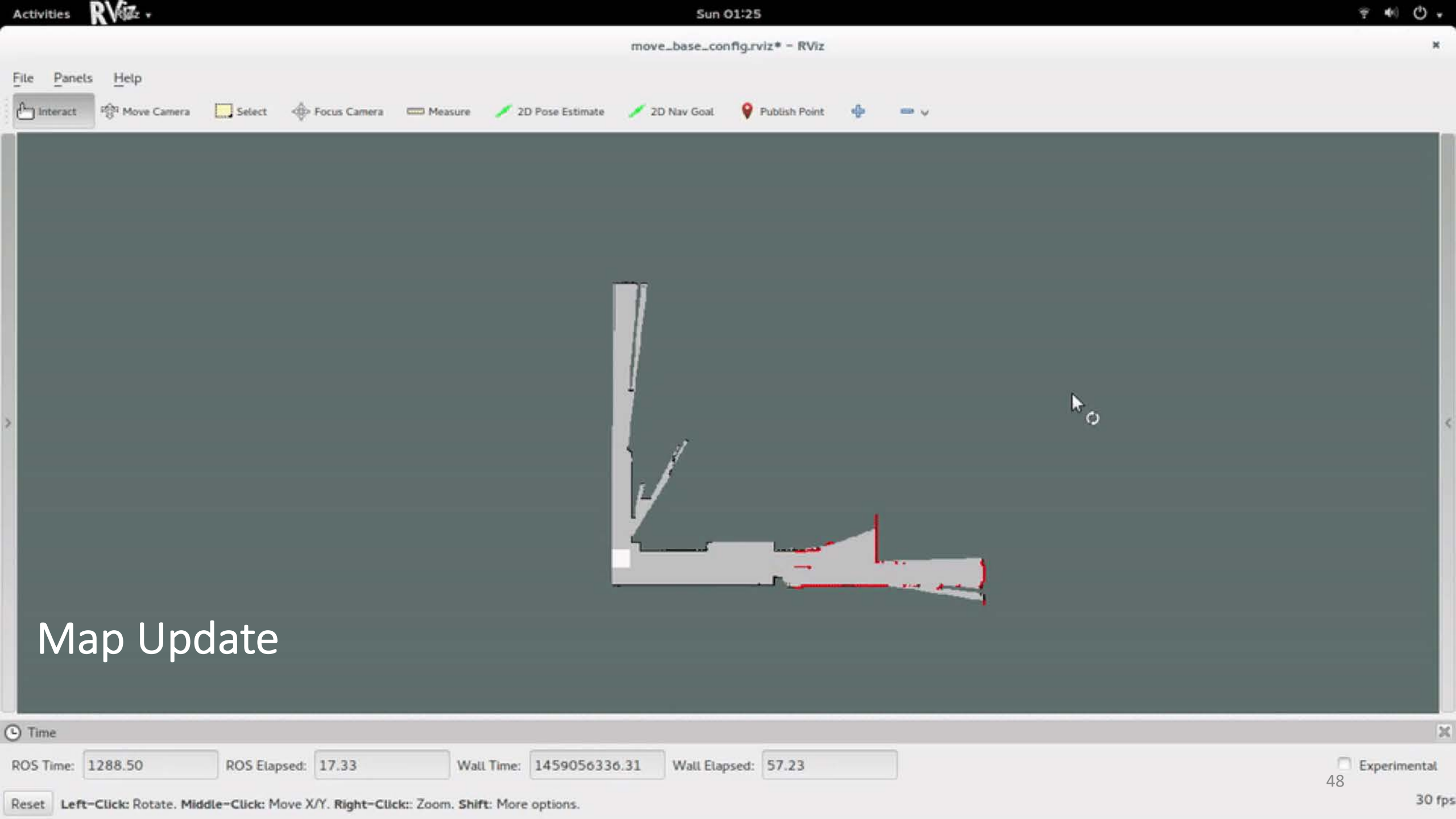


Laser Scans w.r.t car at Time t = t_1

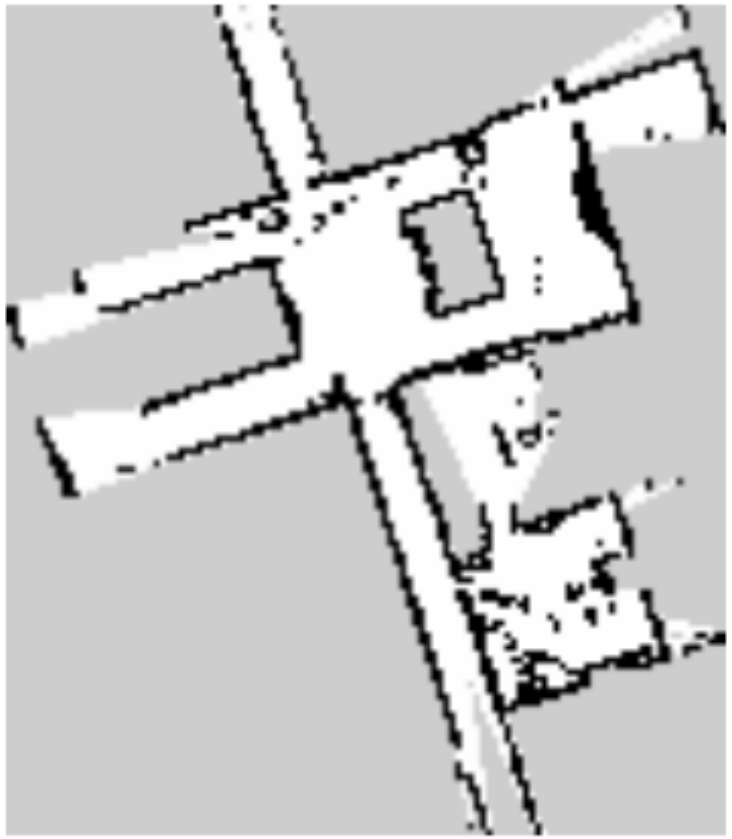
Laser Scans w.r.t car at Time t = t_2


Activities
Sun 00:37
move_base_config.rviz* - RViz
File
Panels
Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Baseframe Axes
Raw LiDAR Scans
Time
ROS Time: 1277.62
ROS Elapsed: 6.38
Wall Time: 1459053427.78
Wall Elapsed: 6.31
Experimental
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.
30 fps

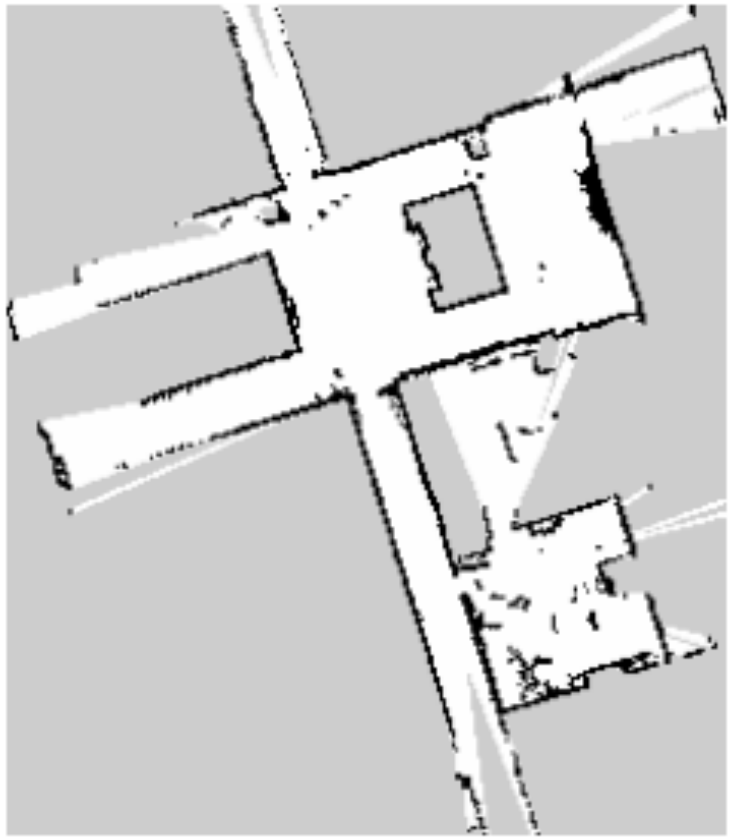

Activities
Sun 00:39
move_base_config.rviz* - RViz
File
Panels
Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Scans after transforming by Δξ at each stage
Mapframe Axes
Time
ROS Time: 1277.14
ROS Elapsed: 5.84
Wall Time: 1459053572.81
Wall Elapsed: 28.00
Experimental
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.
30 fps


Activities
Sun 01:25
move_base_config.rviz* - RViz
File
Panels
Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Map Update
Time
ROS Time: 1288.50
ROS Elapsed: 17.33
Wall Time: 1459056336.31
Wall Elapsed: 57.23
Experimental
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.
30 fps

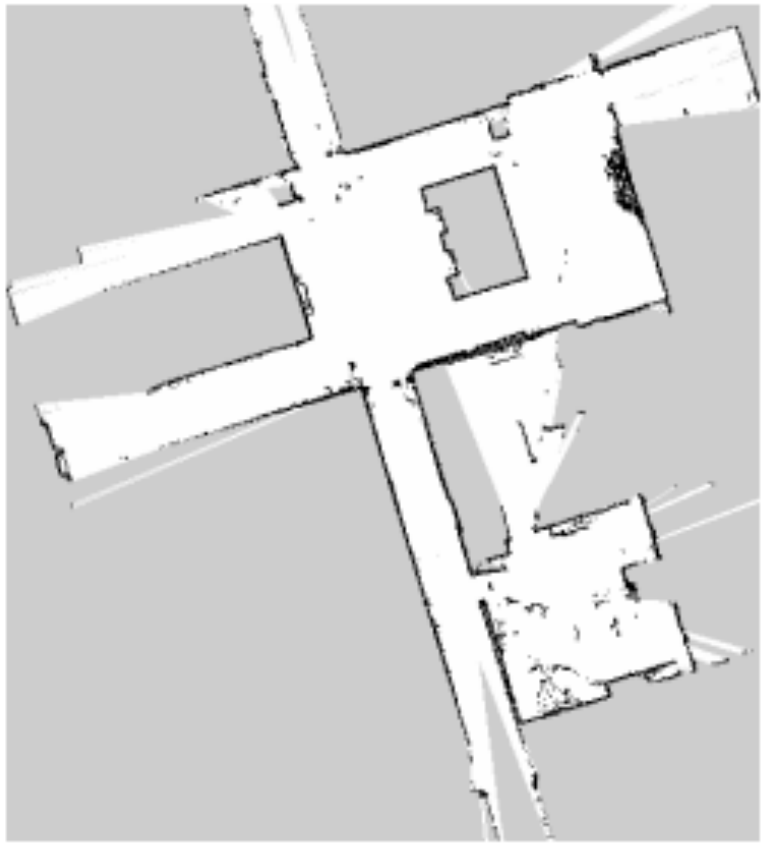
Multi-Resolution Map Representation



20 cm Grid Cell

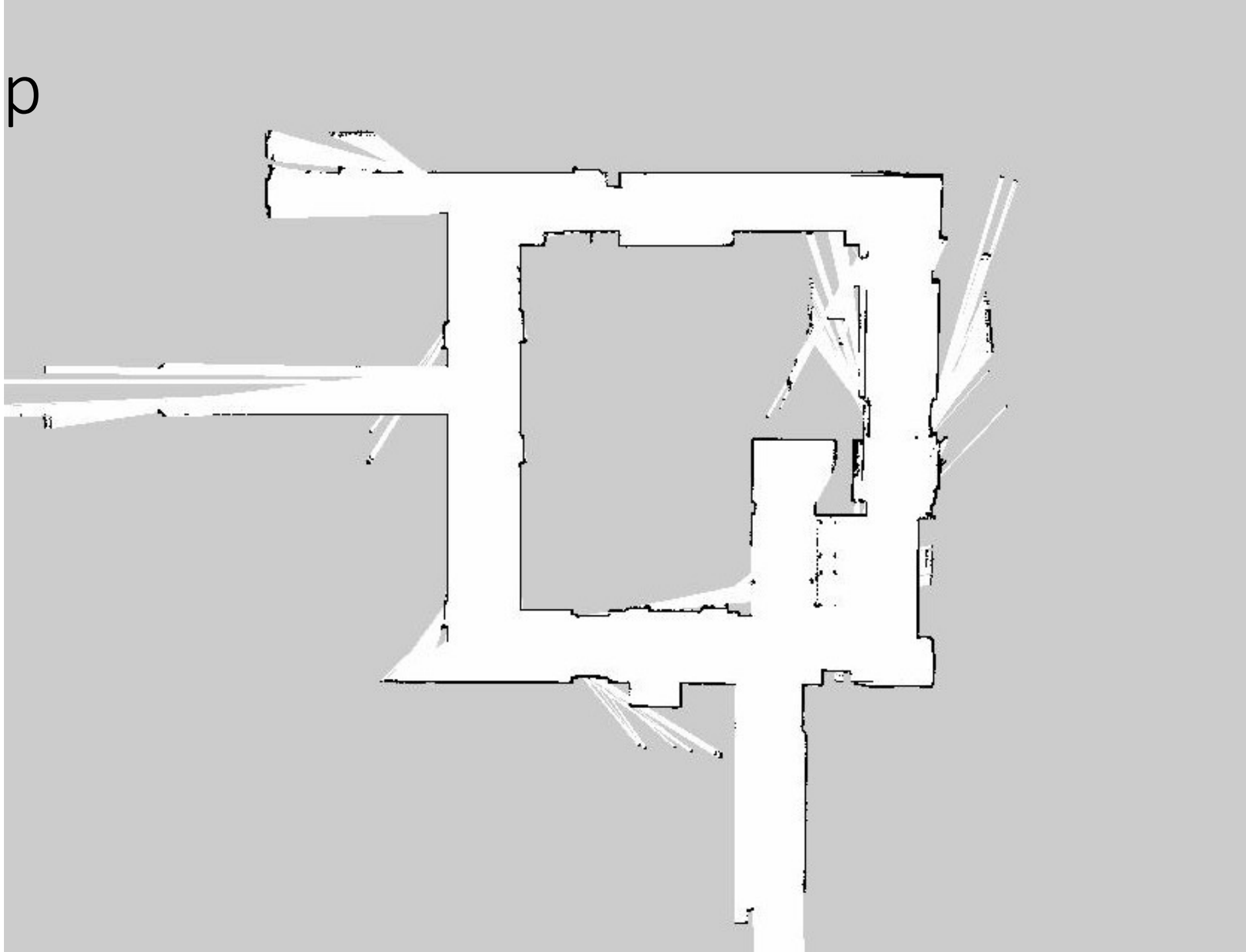


10 cm Grid Cell

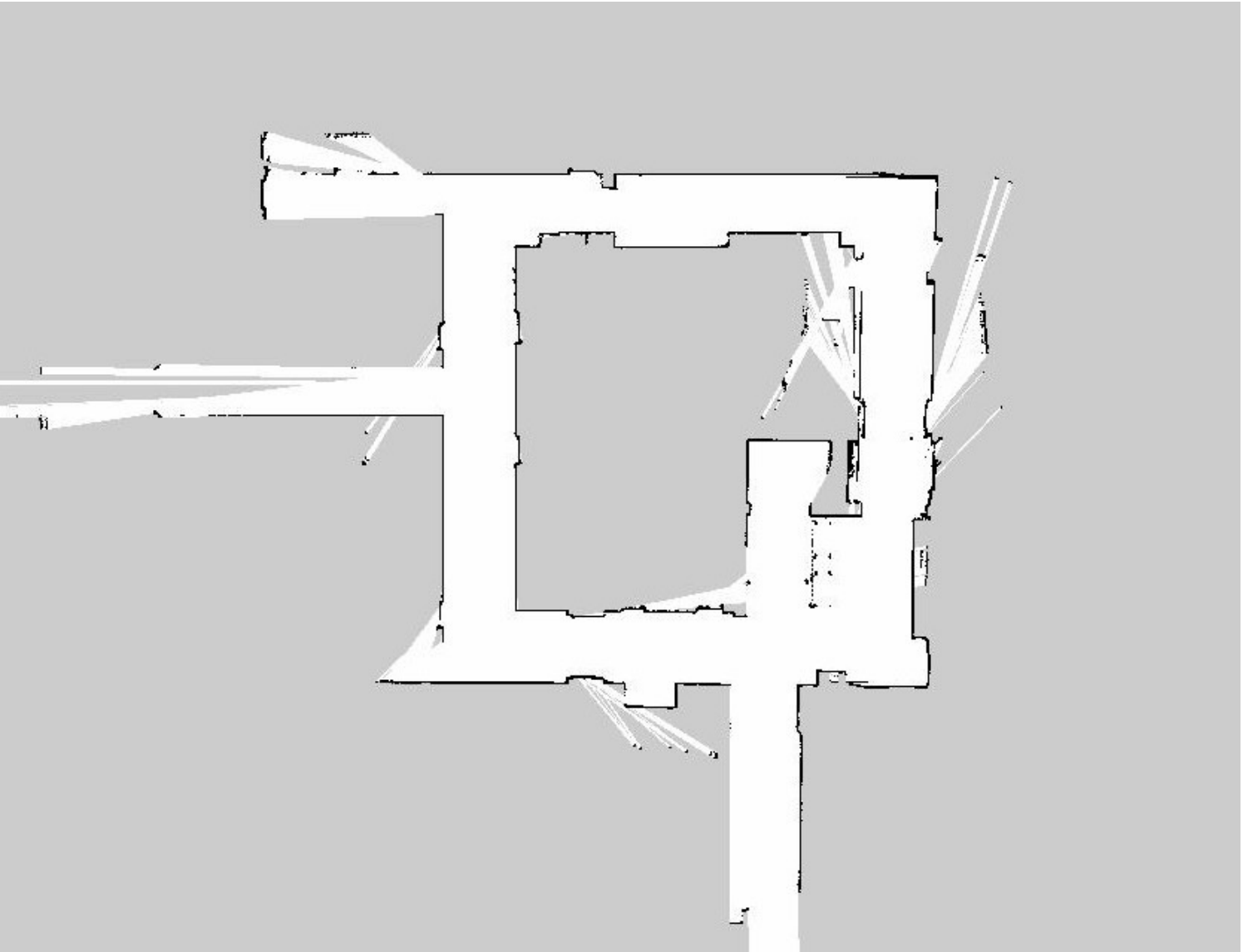


5 cm Grid Cell

Saving the map

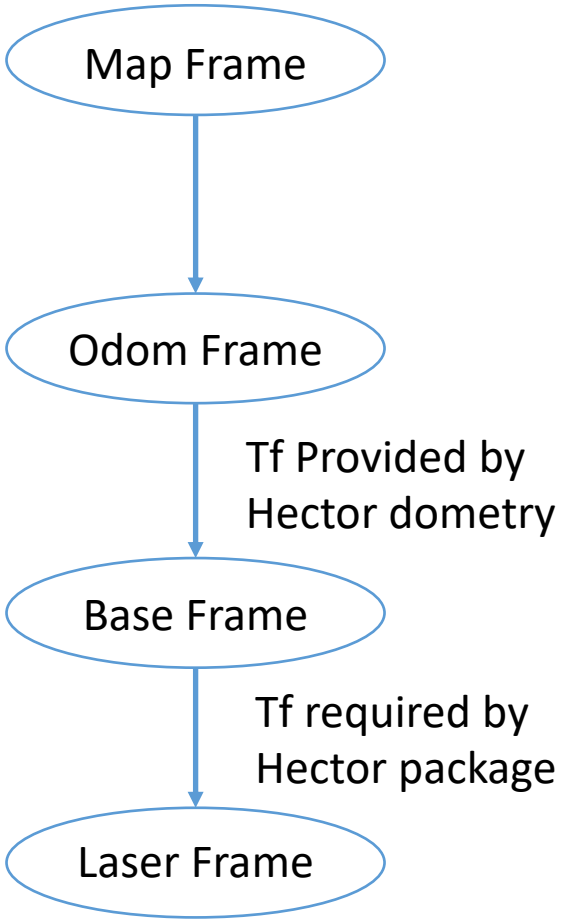


Saving the map



- ROS Package called
  MAP Server
- Allows saving a map currently being published over /map topic
- The saved map can be loaded for future tasks.

System Tf tree



Parameters for Hector SLAM : ROS

- map resolution
- map_update_distance_thresh
- map_update_angle_thresh
- laser_max_dist
- update_factor_free
- update_factor_occupied

Google Cartographer Overview

The Problem

Why did Google (not Waymo) make a SLAM package?

What's different about Cartographer

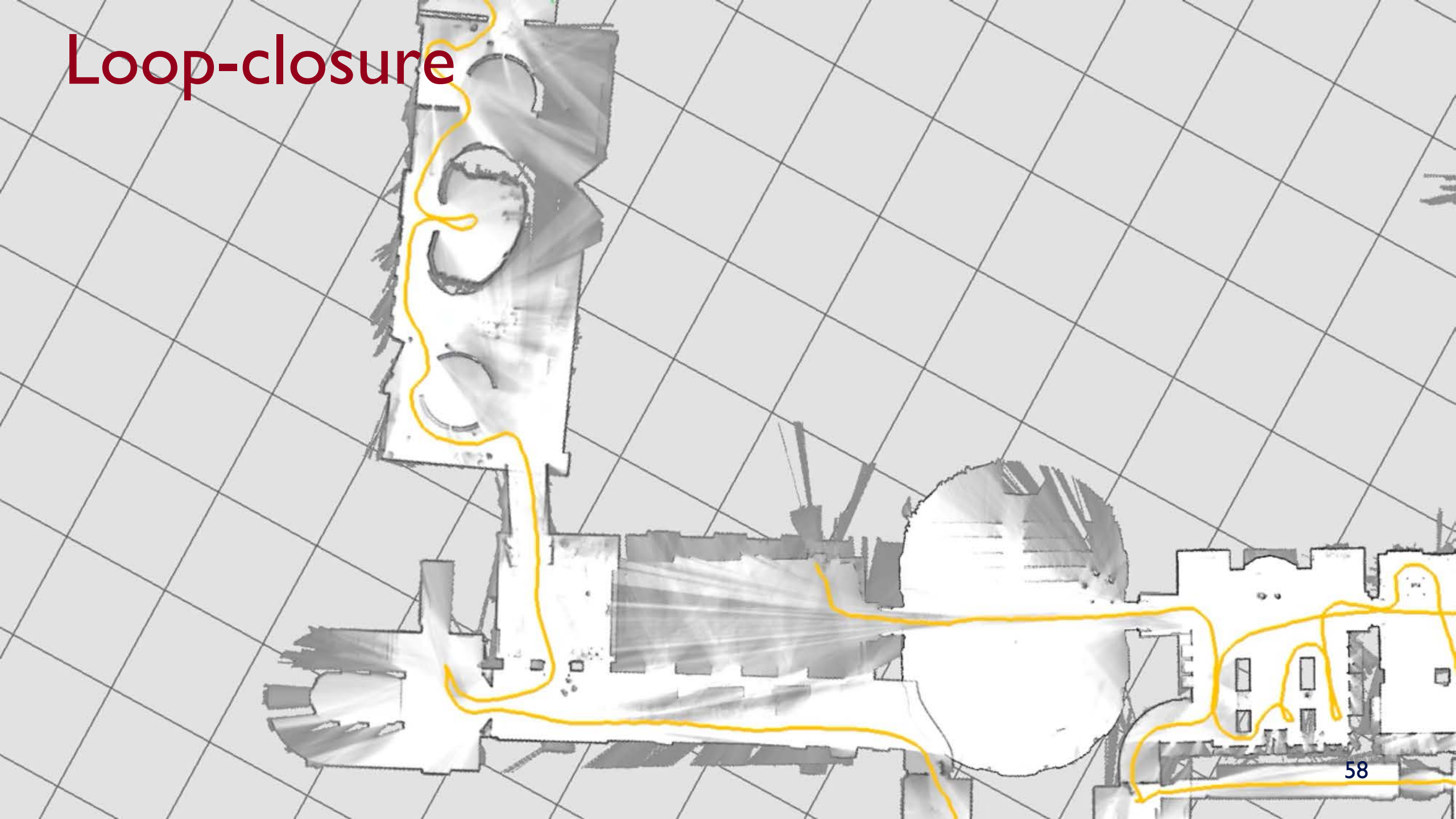
The contribution of this paper is a novel method for ***reducing the computational requirements of computing loop closure constraints*** from laser range data.

Loop-closure

Two regions in the map are found to be the same region in the world even though their position is incompatible given the uncertainty estimate in the map — the classic loop closure problem.

The system must then be able to calculate the transformation needed to align these two regions to ‘close the loop’.

Loop-closure

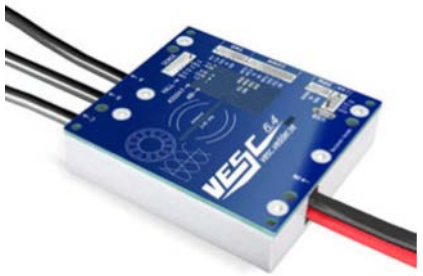


System Overview: Sensor Inputs

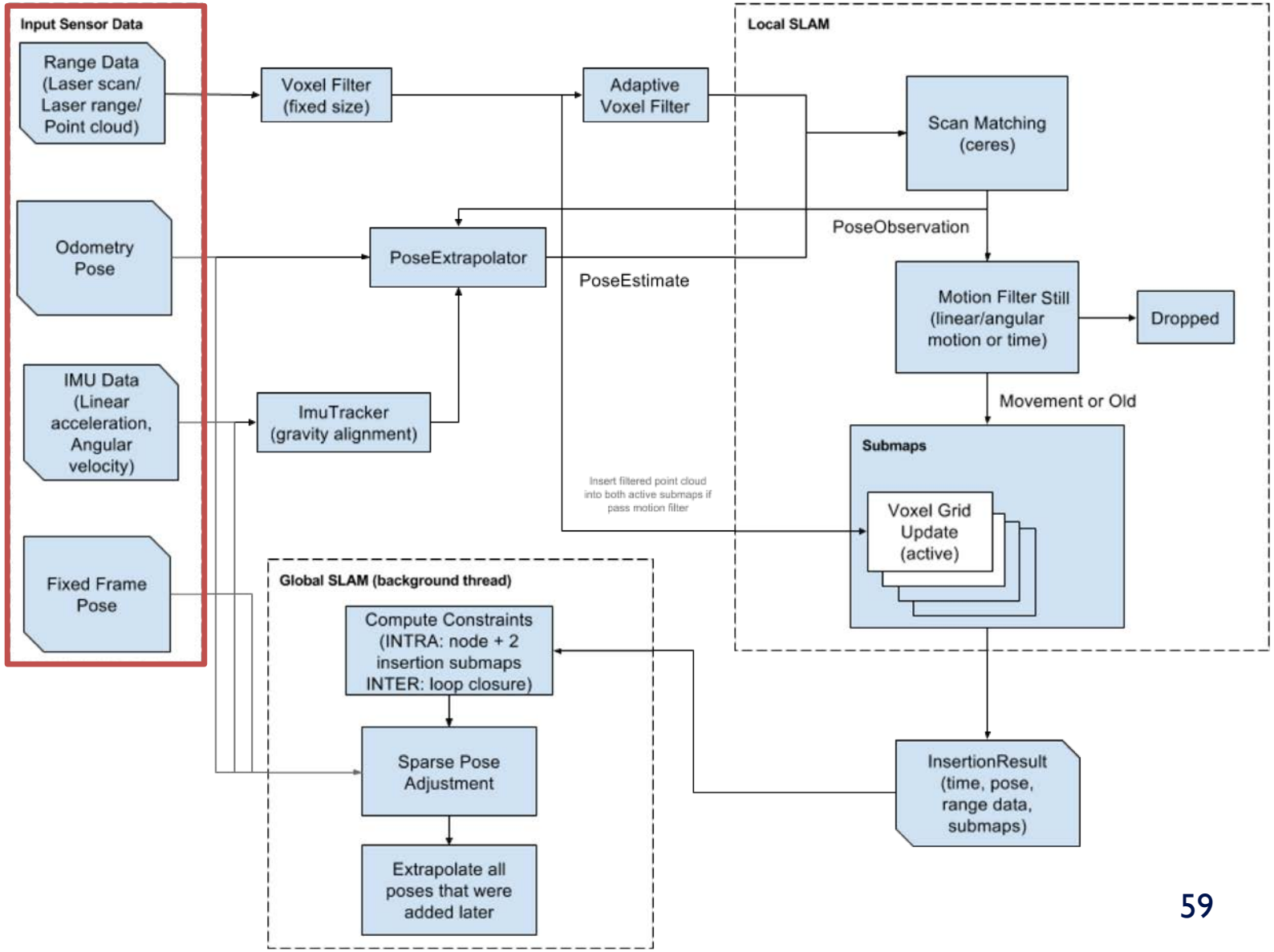
Up to four types of sensor measurements: (I) Range (II) Odometry (III) IMU (IV) Fixed Frame Pose

The F1/10 vehicle supplies two sensors:

2D LIDAR (Range)



VESC (Odometry)

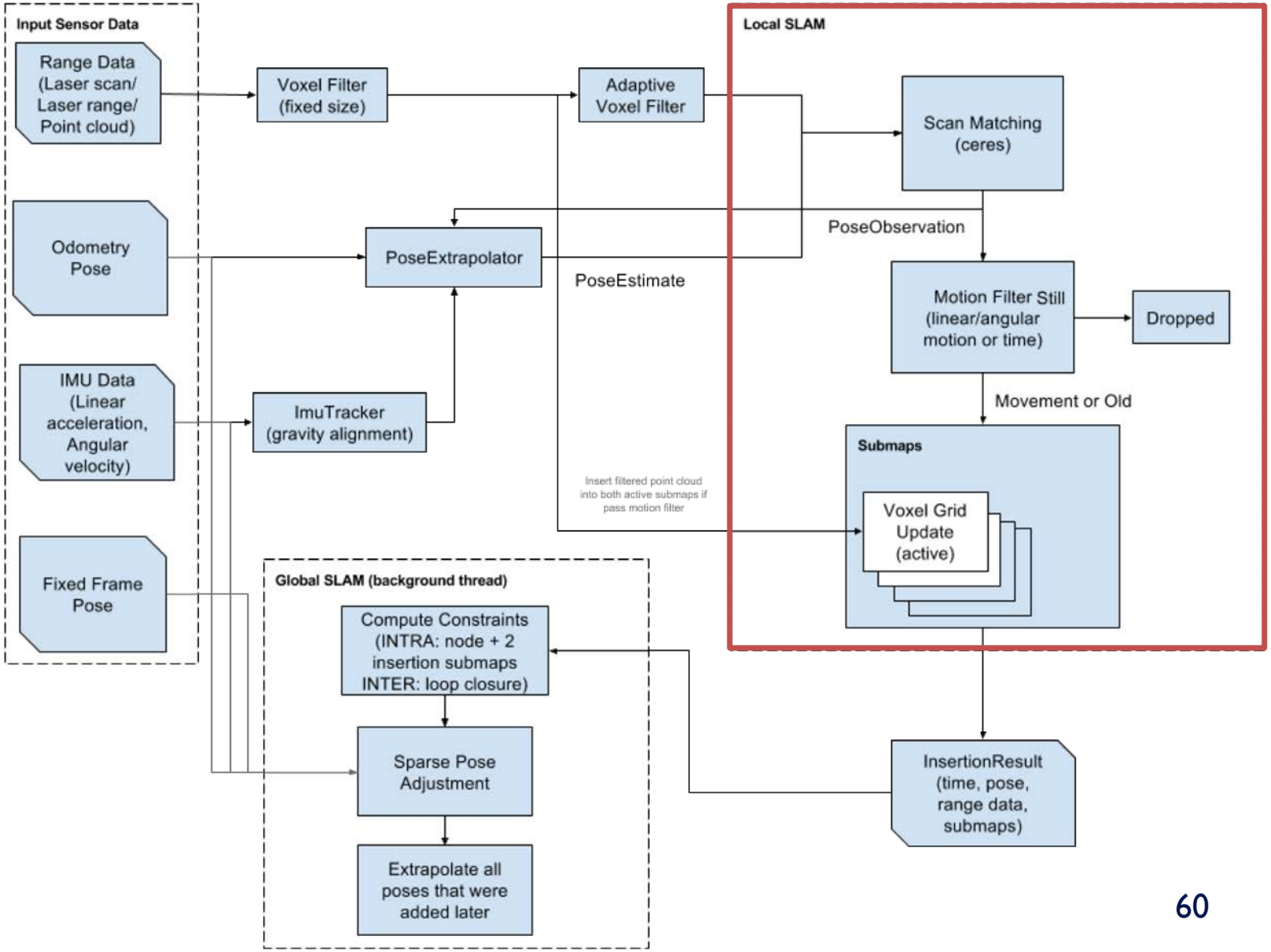


System Overview: Frontend

Cartographer consists of two separate subsystems: local SLAM (frontend or trajectory builder) and global SLAM (backend).

The job of local SLAM is to generate good submaps.

The job of global SLAM is to tie the submaps consistently together.

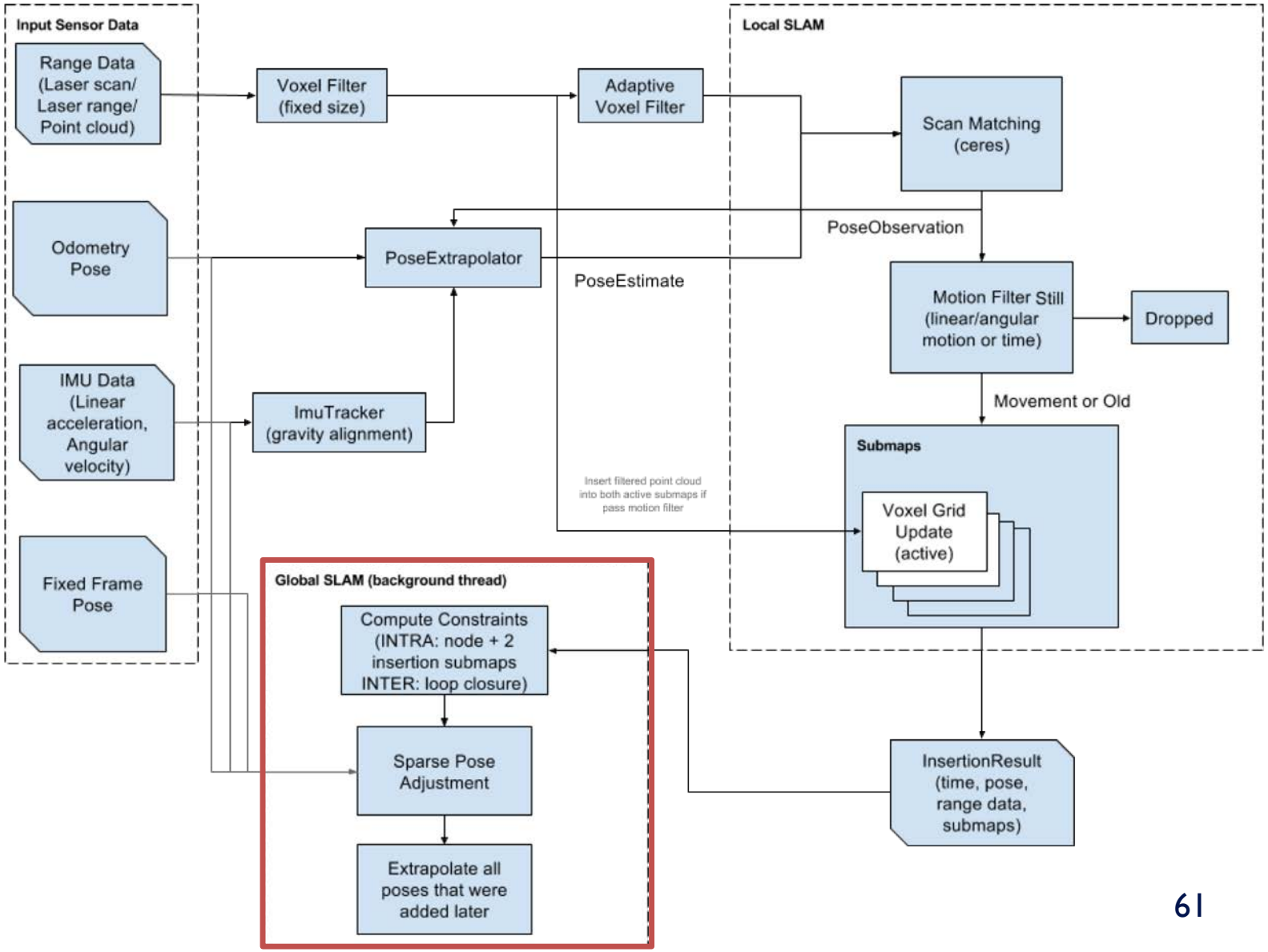


System Overview: Backend

Cartographer consists of two separate subsystems: local SLAM (frontend or trajectory builder) and global SLAM (backend).

The job of local SLAM is to generate good submaps.

The job of global SLAM is to tie the submaps consistently together.



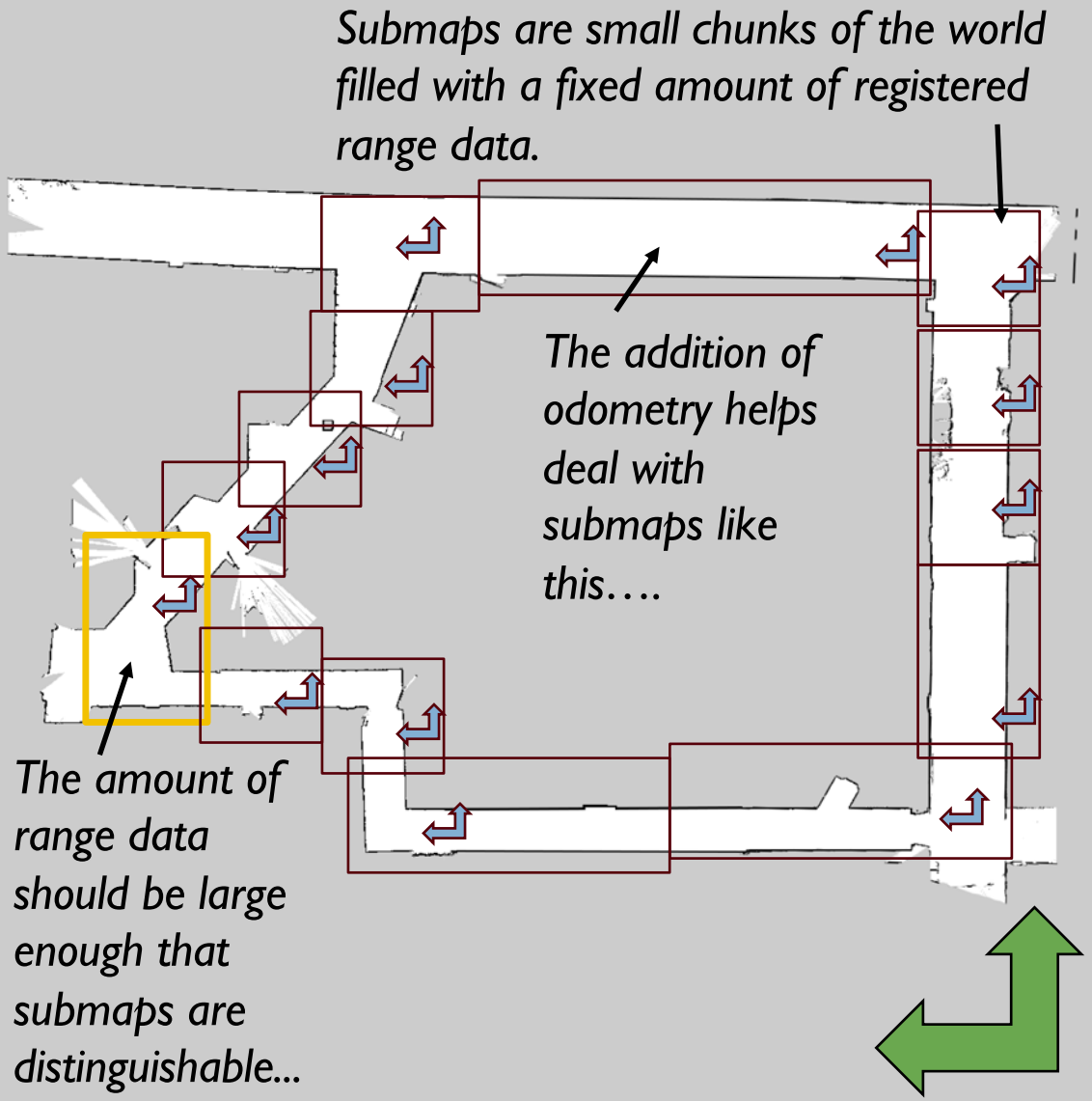
Frontend: Local SLAM

What is a submap?

A submap is considered as complete when the local SLAM has received a given amount of range data.

Submaps must be small enough so that the drift inside them is below the resolution of the occupancy grid, so that they are locally correct.

On the other hand, they should be large enough to be distinct for loop closure to work properly. More on this later

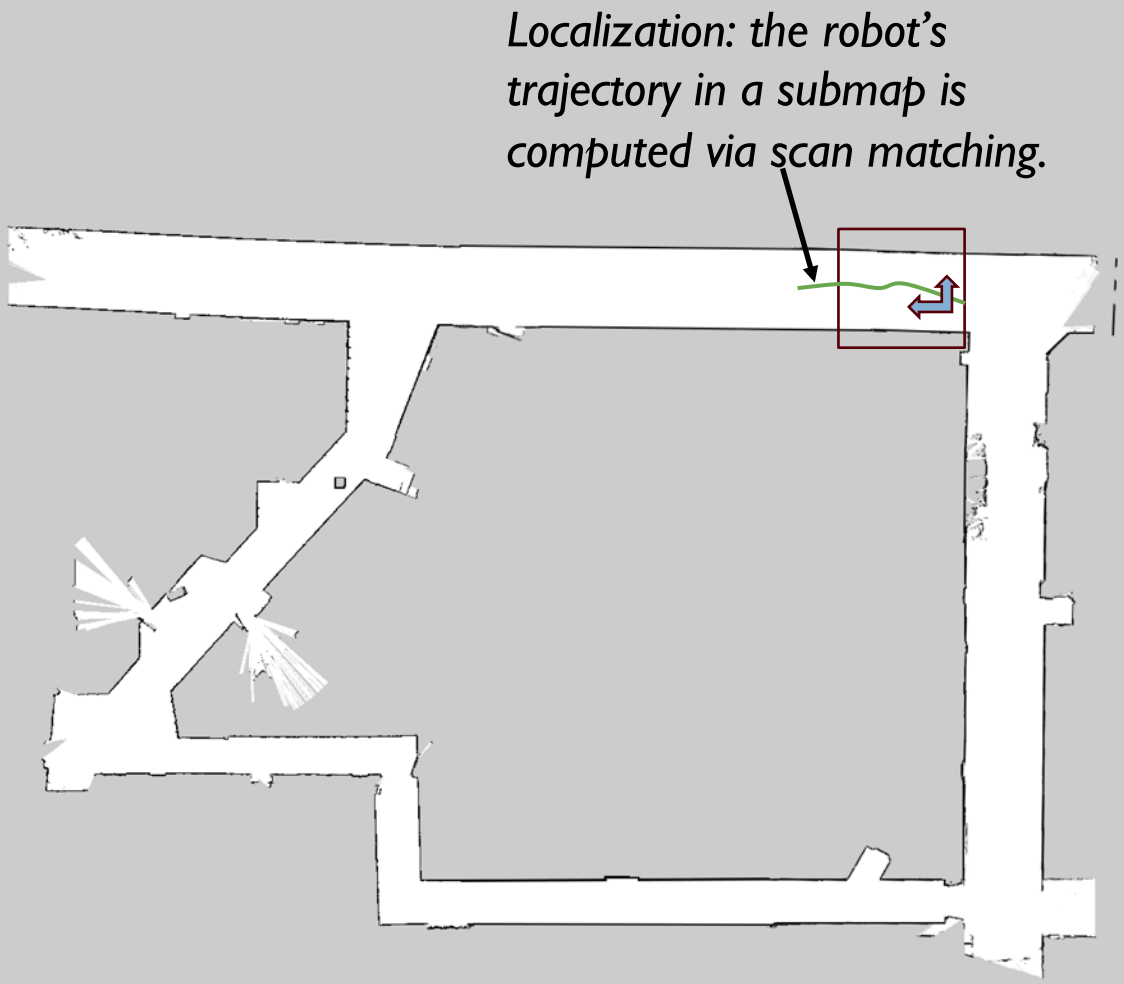


What is a submap?

Submaps each have their own static transform and contain a collection of registered range-measurements.

Consecutive measurements are connected by constraints which are ‘local’.

Here local means derived from odometry or recent scan overlaps and resultant scan matches.



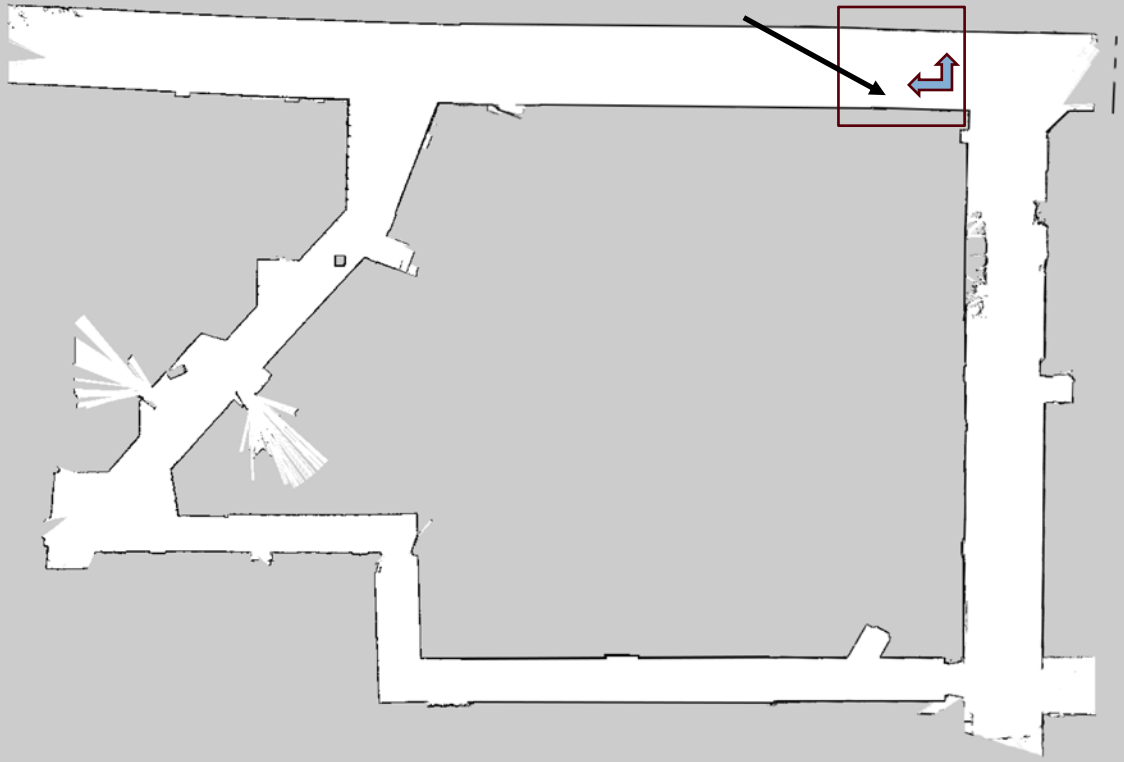
What is a submap?

Submaps each have their own static transform and contain a collection of registered range-measurements.

Consecutive measurements are connected by constraints which are ‘local’.

Here local means derived from odometry or recent scan overlaps and resultant scan matches.

Once a motion between two scans is found by the scan matcher, it goes through a motion filter. A scan is inserted into the current submap only if its motion is above a certain distance, angle or time threshold.

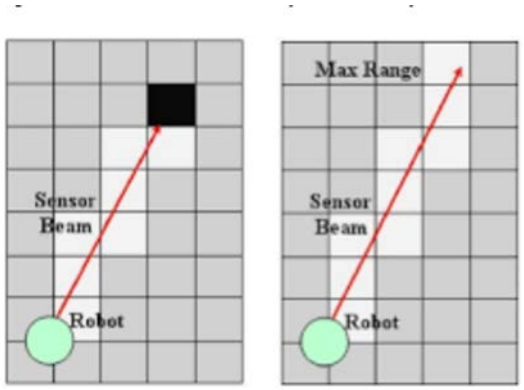


Submap Representation

Submaps can **store their range data in a probability grid**. For 2D a signed distance field representation is also possible.

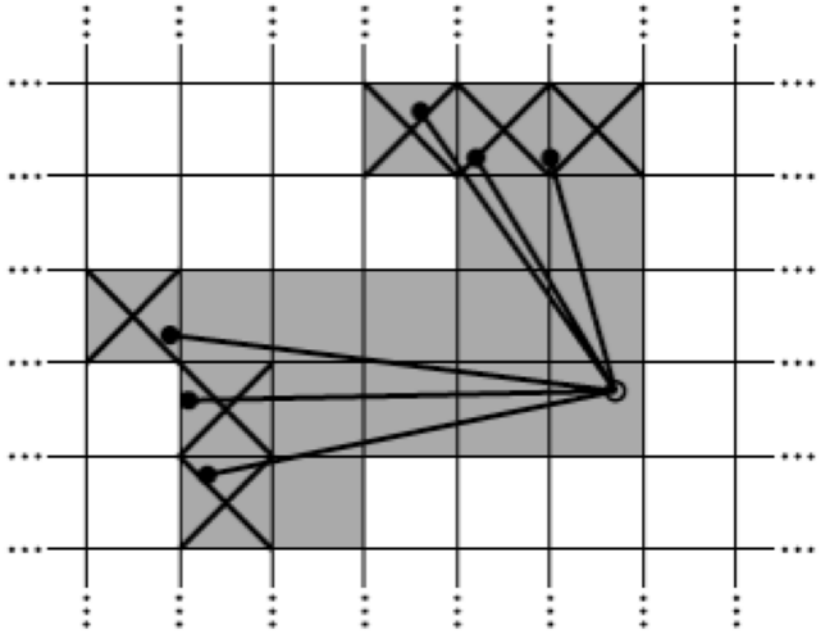
Probability grids are a 2D table where each **cell has a fixed size and contains the odds** of being obstructed.

Odds are updated according to **“hits”** (where the range data is measured) and **“misses”** (the free space between the sensor and the measured points).



Updating the submap

1. For every hit, we **insert the closest grid point** into the hit set.
2. For every miss, **we insert the grid point associated with each pixel that intersects one of the rays between the scan origin and each scan point**, excluding grid points which are already in the hit set.
3. If a grid point **has yet to be observed it is assigned a value p_{hit} or p_{miss}** depending on which set it is in.



Define the submap as:

$$M : r\mathbf{Z} \times r\mathbf{Z} \rightarrow [p_{min}, p_{max}]$$

Recall the definition of the odds function:

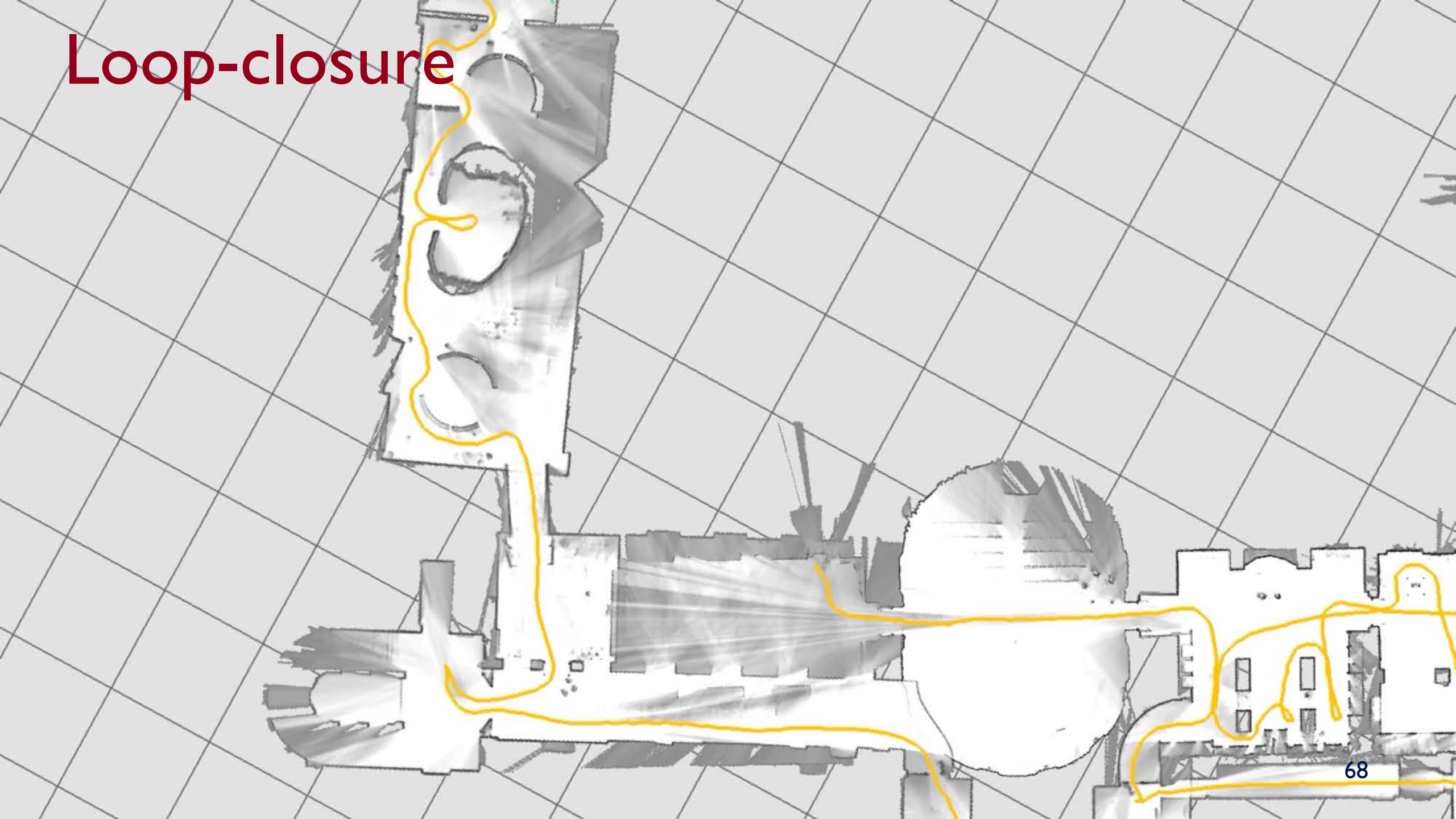
$$odds(p) = \frac{p}{1-p}$$

Then the map is updated according to the following operation:

Compute new p from odds...

$$M_{new}(x) = \mathbf{clamp}\left(odds^{-1}\left(odds(M_{old}(x))\, odds(p_{hit})\right)\right)$$

Loop-closure



Scan Matching:

How do we know that 'hits' and misses map to a particular cell in the probability grid?

- The collection of scan points relative to a moving reference frame attached to the robot are, $H = \{h_k\}$ for $k=1 \ldots K$ where h_k is a point in R^2.

- These points are placed in a submap at pose ξ in the global reference frame by applying a rigid body transform to each (rotation and translation).

- Submap construction is the iterative process of repeatedly aligning scan and submap coordinate frames. *Sound familiar?*

So we can understand why scan matching alone is insufficient for robust localization and explore the basics.

Does Cartographer use ICP?

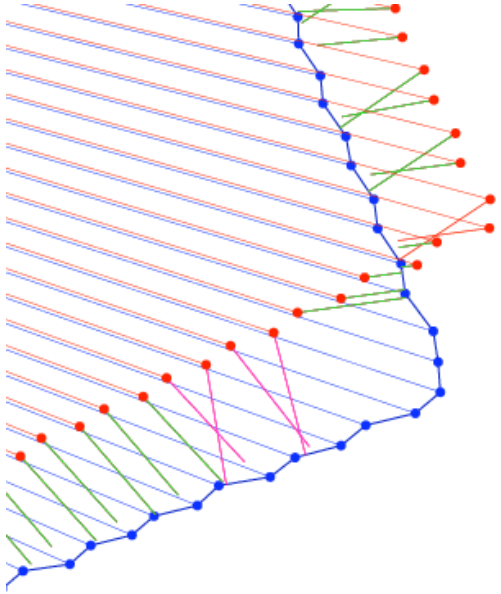
No. Cartographer needs to support 3D LIDARs and the correspondence problem is insidious.

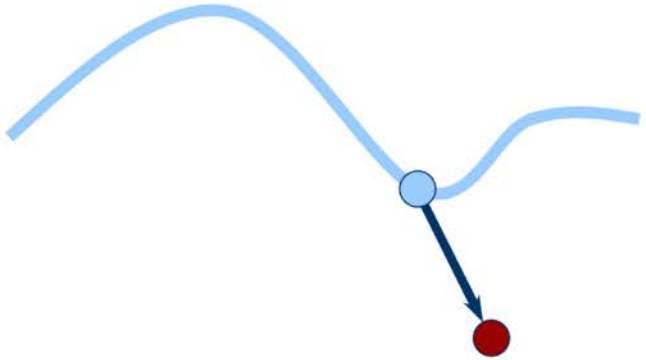
- Scan matching variants:
  - Iterative closest point (ICP)
  - Scan-to-scan
  - Scan-to-map
  - Map-to-map
  - Feature-based
  - RANSAC for outlier rejection
  - Correlative matching

Cartographer uses the Ceres-solver to formulate a nonlinear least-squares correlative scan matching problem.

Closest-Point Matching

- Find closest point in other the point set





Closest-point matching generally stable, but slow and requires preprocessing

Correlation-based Scan Matching

- In ICP, correspondences between two scans are explicitly computed, allowing a rigid-body transformation to be computed.
- Susceptible to local minima; poor initial estimates lead to incorrect data associations and divergence.
- **Correlation based methods search for a rigidbody transformation (without computing correspondences) that projects one set of LIDAR points (the query scan) on top of a reference map.**
- The reference map is generally implemented as a look-up table that assigns a cost for every projected query point.

Correlation-based Scan Matching

Sum over each scan point

Transform scan point from scan frame to local submap frame

Coordinate of scan return

$$\operatorname{argmin}_{\xi} \sum_{k=1}^{K} \left(1 - M_{smooth}(T_{\xi} h_k)\right)^2$$

Pose in local submap reference frame

Bicubic interpolation of probability grid

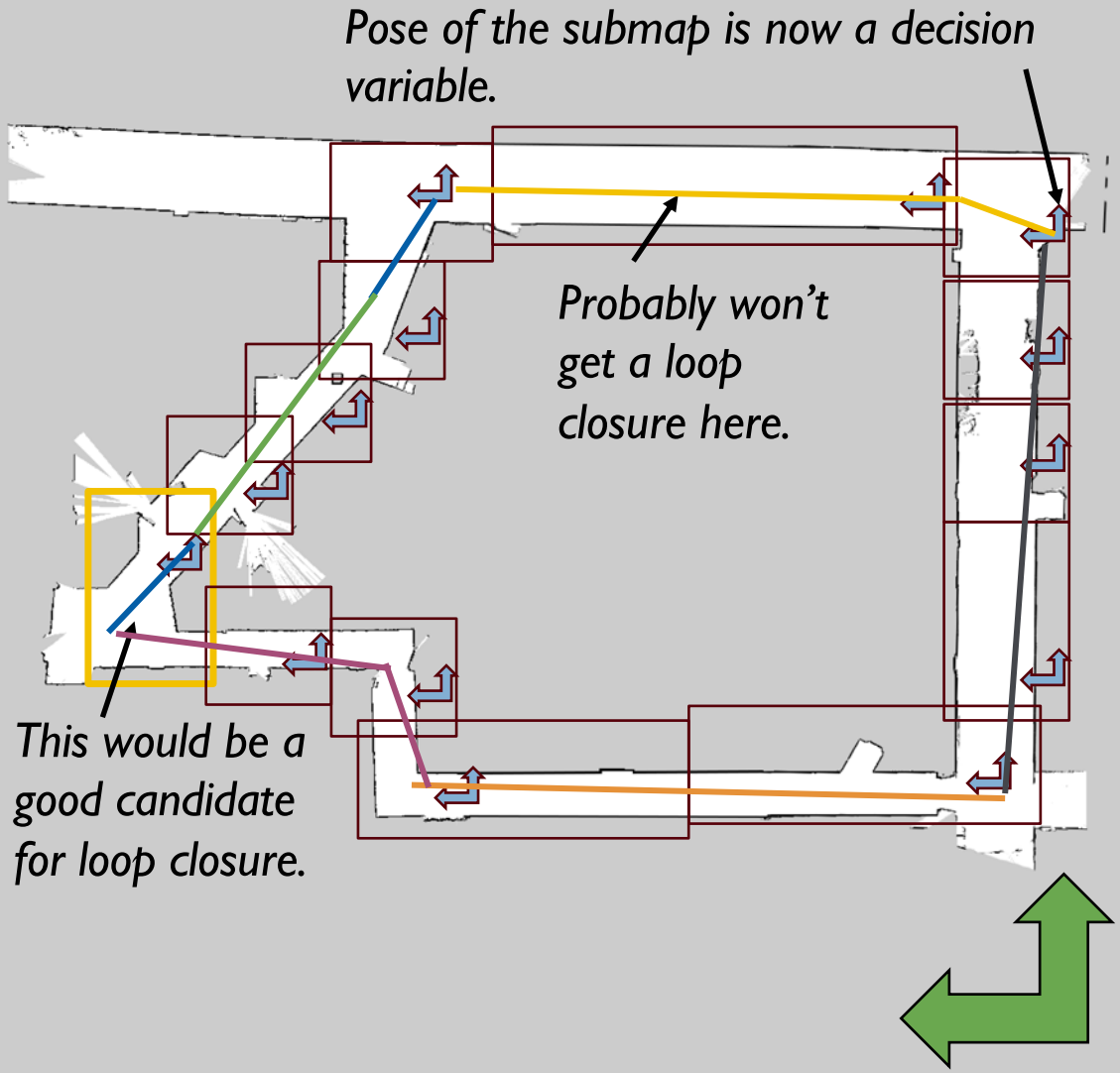
Backend: Global SLAM

Closing Loops

Goal: is our current scan in one of the submaps we have already seen?

Constraints take the form of relative poses ξ_{ij} and associated covariance matrices Σ_{ij} . **Relative poses now include both submap and scan poses.**

For a pair of submap i and scan j, the pose ξ_{ij} describes where in the submap coordinate frame the scan was matched.



Loop Closure

Sum over each scan, submap combination

Residual, see paper

Covariance matrix

$$\underset{\Xi^m,\Xi^s}{\operatorname{argmin}} \frac{1}{2} \sum_{i,j} \rho\left(E^2\left(\xi_i^m, \xi_j^s; \Sigma_{i,j}, \xi_{ij}\right)\right)$$

Submap poses

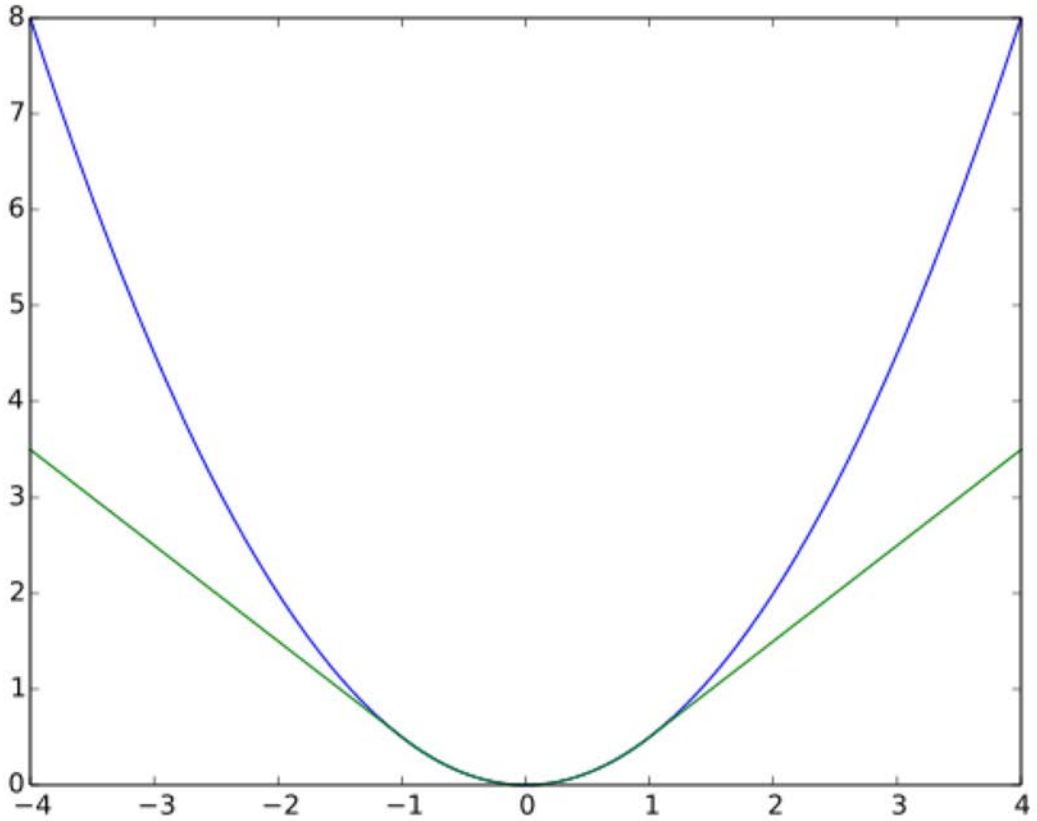
Scan poses

Huber loss, outlier rejection

Relative pose between submap and scan

Huber Loss

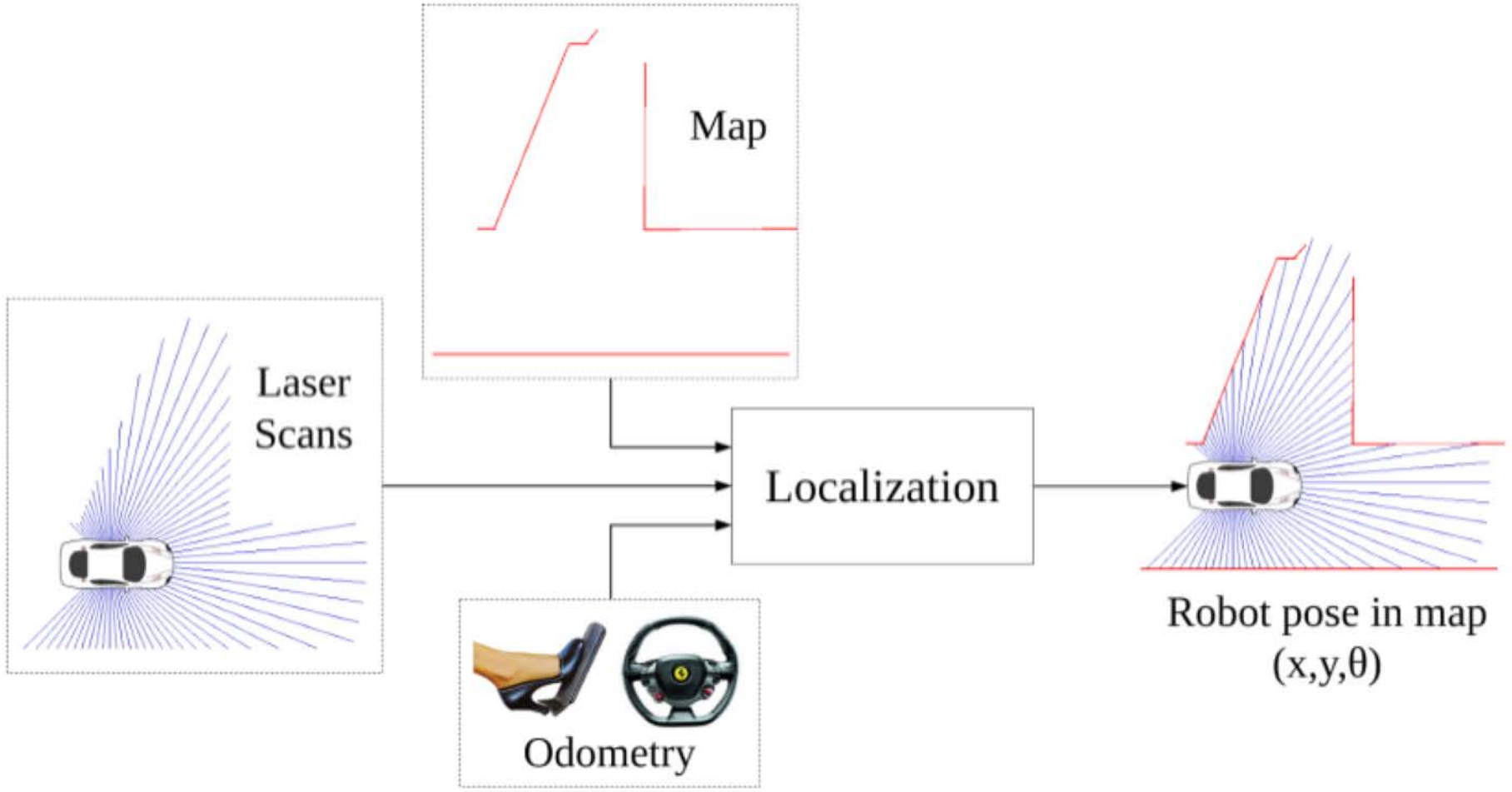
Uses a Huber loss to make the objective less susceptible to spurious constraints which are often added in symmetric environments… False positive is a disaster



$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta(|a| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$$

Hector SLAM

Problem Setting



Overview

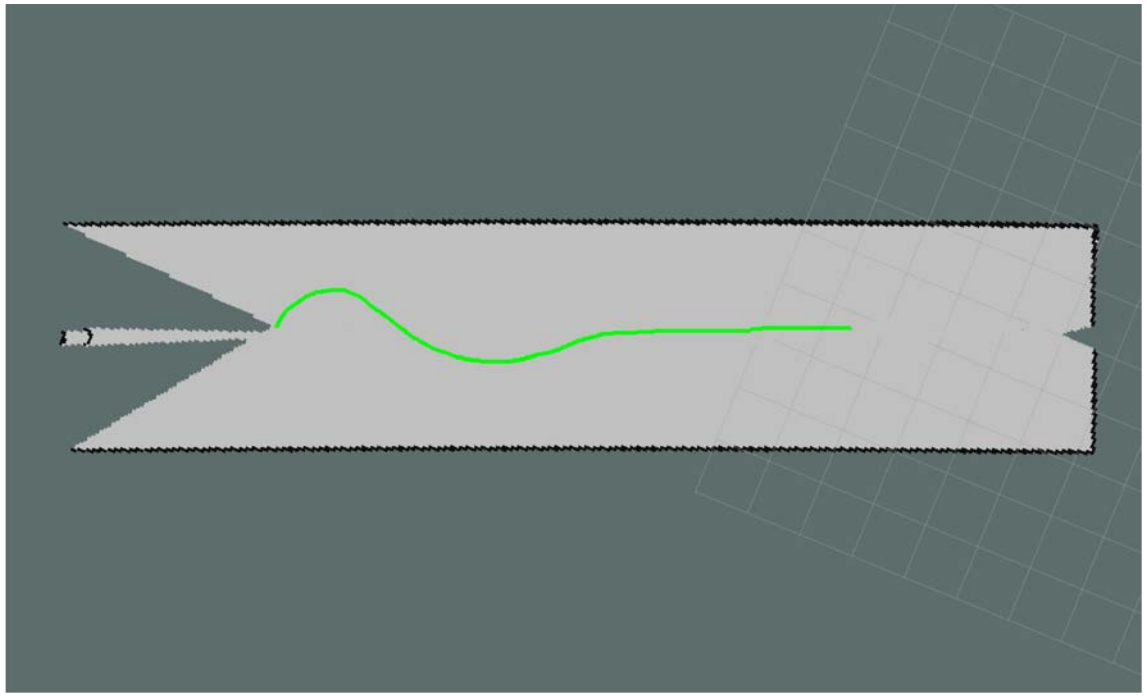
Input

1. Laser Scan

Overview

Input

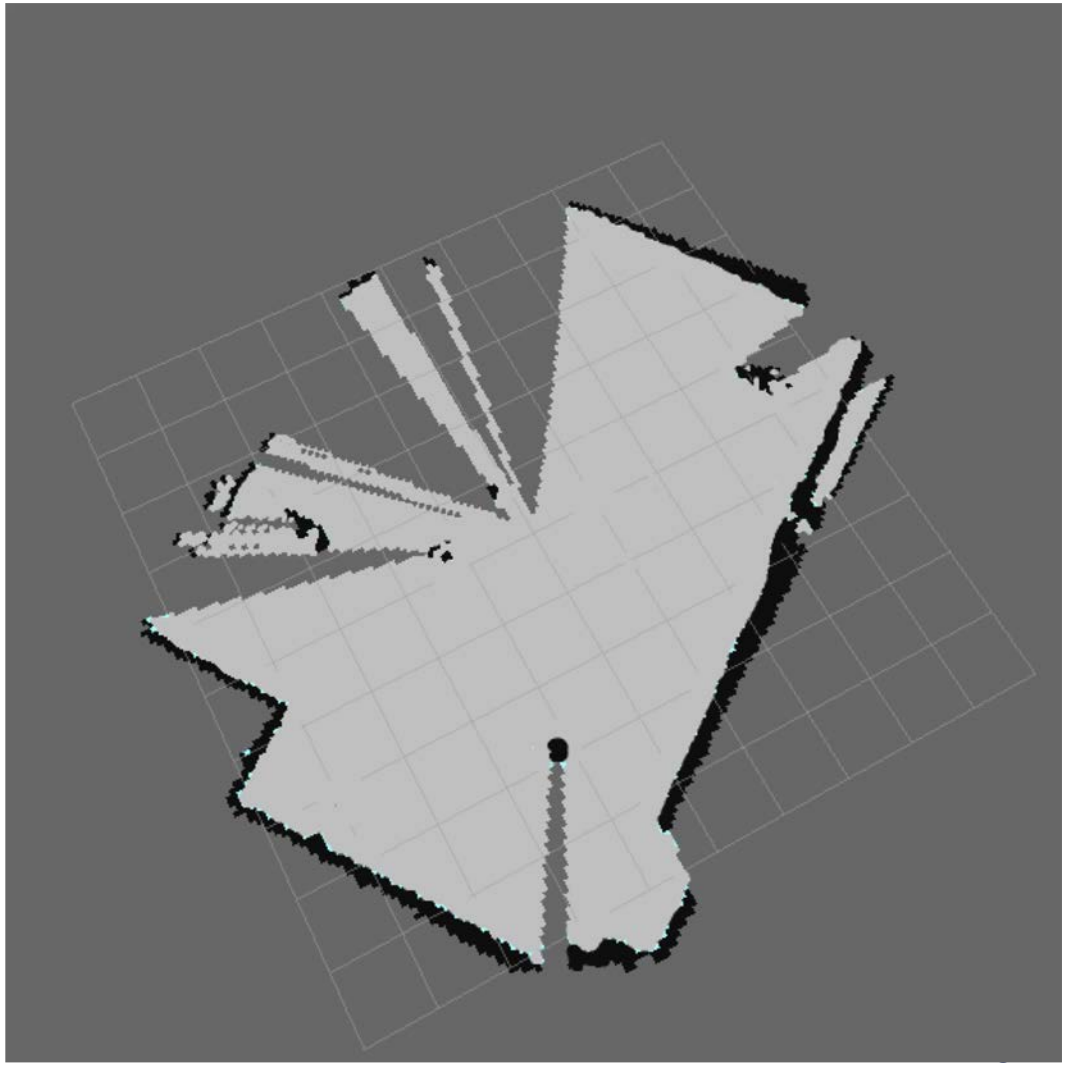
1. Laser Scan
2. Control Input



Overview

Input

1. Laser Scan
2. Control Input
3. Map



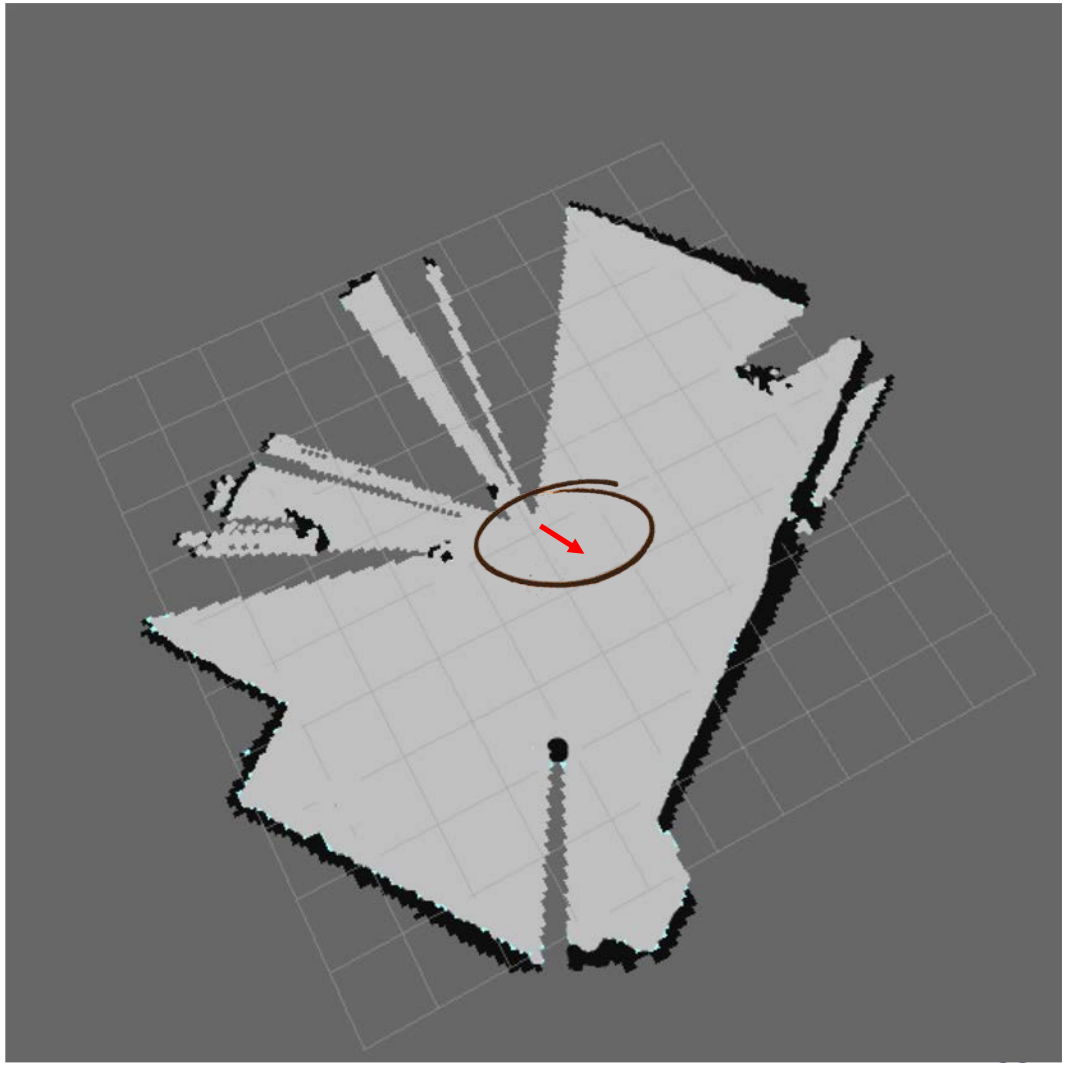
Overview

Input

1. Laser Scan
2. Control Input
3. Map

Output

1. Pose



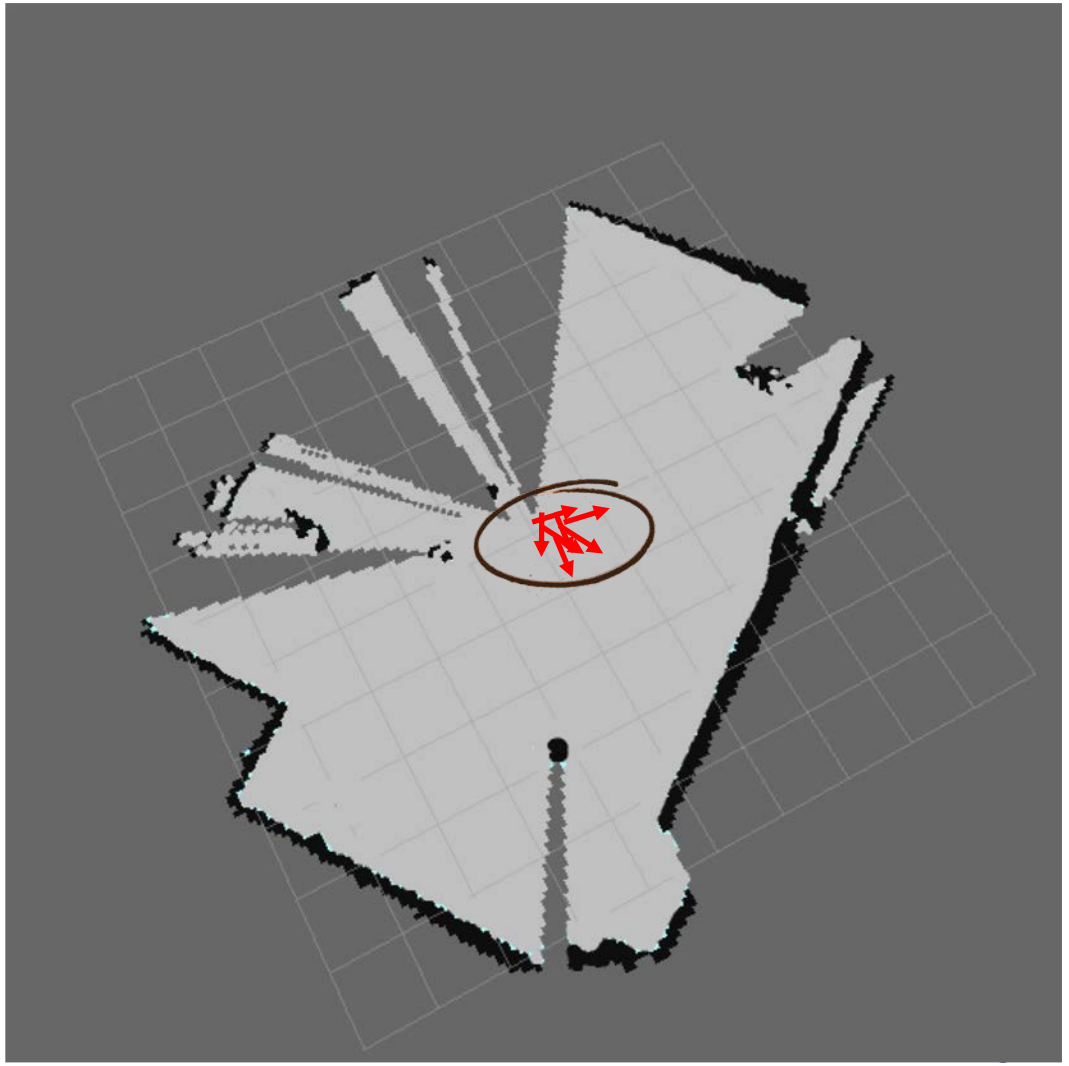
Overview

Input

1. Laser Scan
2. Control Input
3. Map

Output

1. Pose
2. Particles



Particle Filter Localization

Key Idea

The key idea of Bayes filtering is to estimate a probability density over the state-space conditioned on the data. This posterior is typically called the belief and is denoted:

$$Bel(x_t) = p(x_t \mid d_{0\ldots t})$$

Belief or posterior

Robot state

At time t

The data from time 0 to t.

Questions

1. How would you describe the robot state in the localization problem?
2. Is the belief a probability distribution, what kind, how do you write it down?
3. How would you describe the data from the F1tenth car?

$$Bel(x_t) = p(x_t \mid d_{0\ldots t})$$

Belief or posterior

Robot state

At time t

The data from time 0 to t.

Getting more specific…

For mobile robots, we distinguish two types of data: perceptual data such as laser range measurements, and odometry data, which carries information about robot motion. Denoting the former by o (for observation) and the latter by a (for action), we have:

$$Bel(x_t) = p(x_t \mid o_t, a_{t-1}, o_{t-1}, \ldots)$$

Belief or posterior

Robot state

Perceptual Observation

Odometry

History...

Recursion

After some simplification (see additional resources) we have:

Integrate out over previous beliefs. In practice finite approximation

$$Bel(x_t) = \eta \; p(o_t \mid x_t) \int p(x_t \mid x_{t-1}, a_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

Normalization Constant Make sure everything adds up to 1!

Sensor Model
Compute how likely your measurements were given updated particles.

Motion Model
Simulate noisy dynamics of particles based on control input.

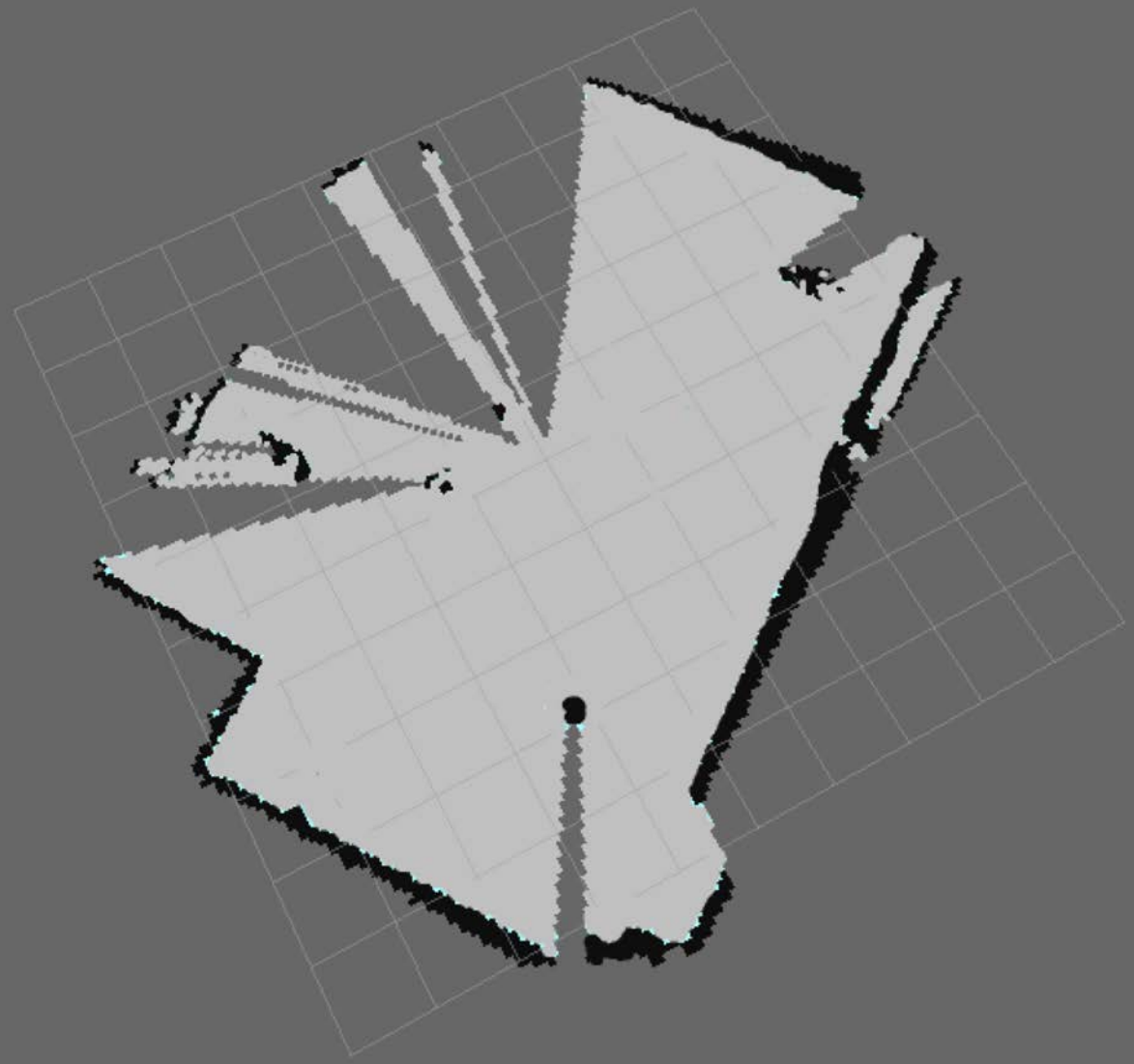
Previous Belief
Draw your particles with replacement according to importance weight.

Read left to right

In practice we represent the distributions non-parametrically using particles (a finite set of samples). More in the next slides.

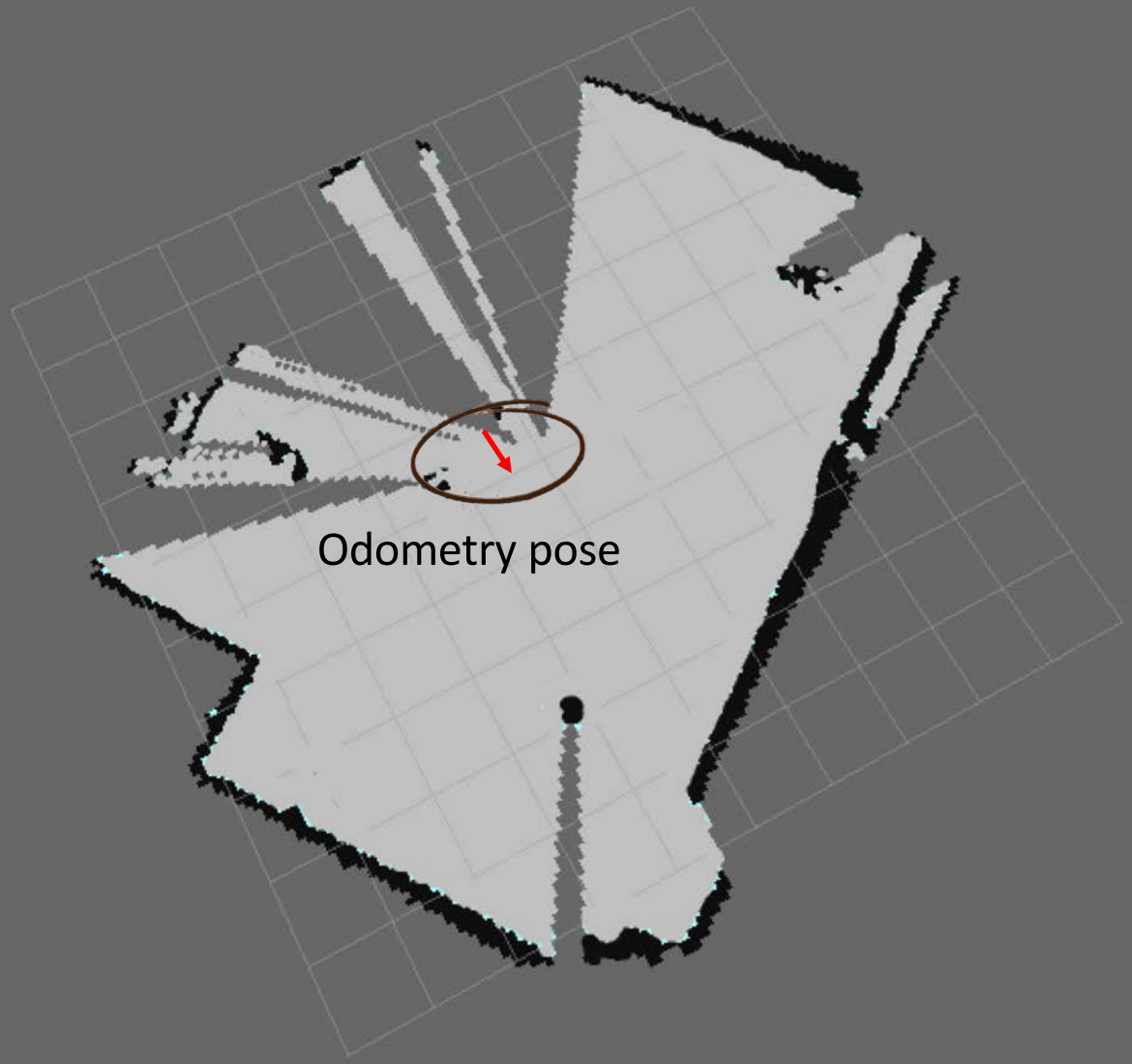
Initialization

Now let's look at particle filters in three dimensions and the process of using them in our car. Here we see a part of a map previously generated using Cartographer. The black pixels on the map denote the walls and the grey pixels denote free space.



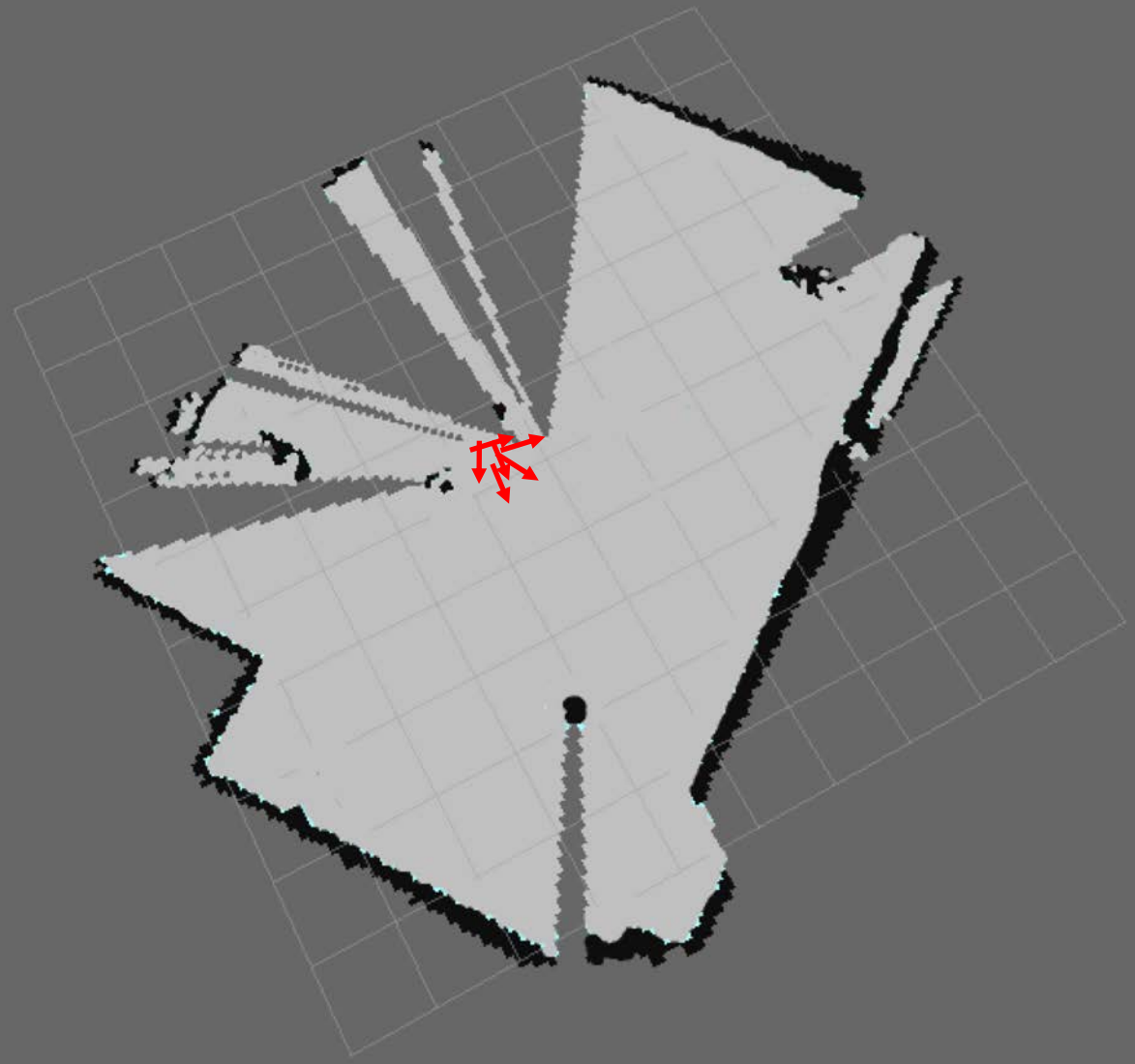
Initialization

The red arrow indicates our initial pose obtained from user input. Lets use particle filters to solve the pose tracking problem and localize more accurately in the map.



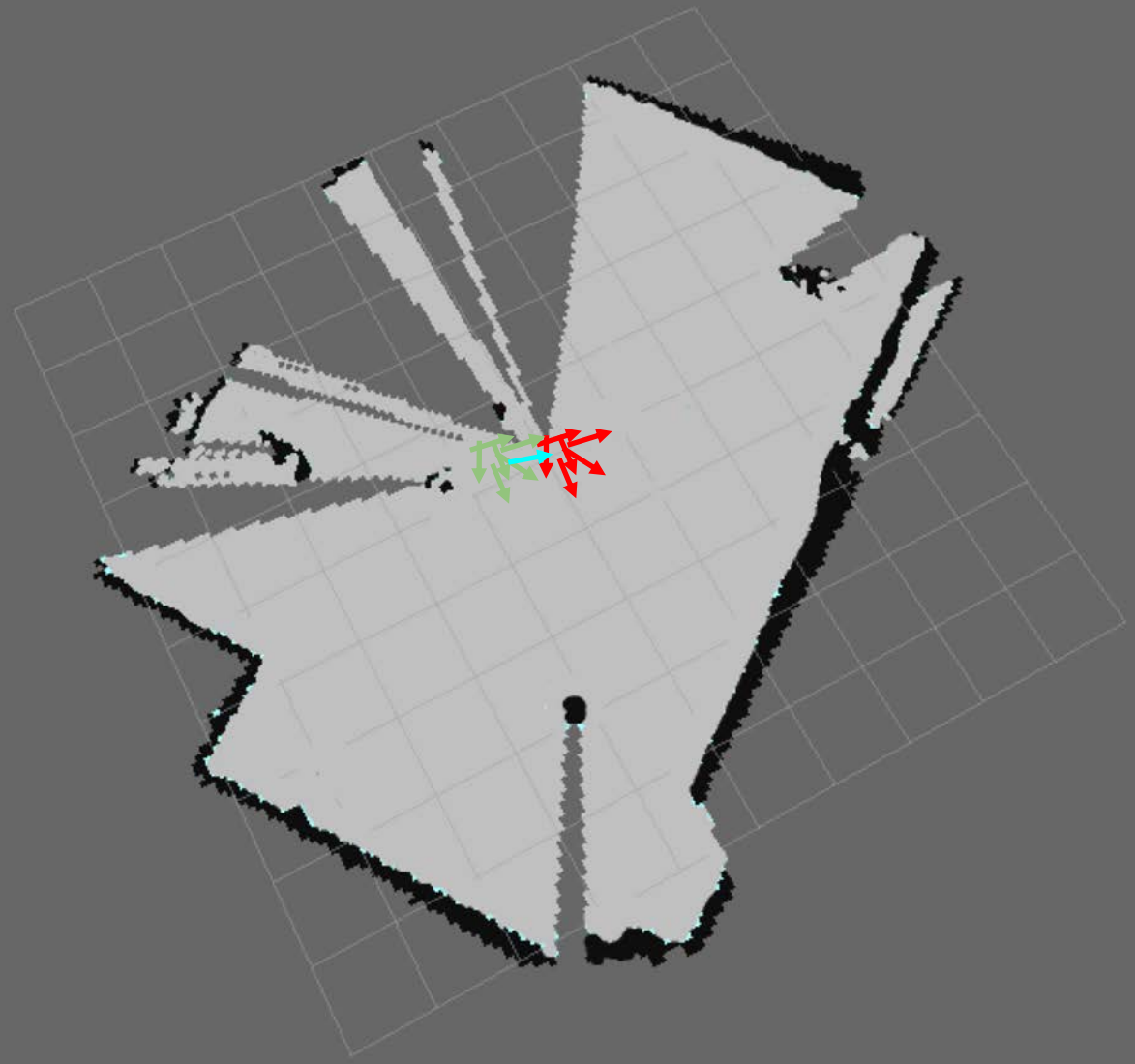
Initialization

First we need to generate a set of hypothesis for our first position. These the discrete particles drawn from a Gaussian distribution of mean being the initial guess and with a small covariance.



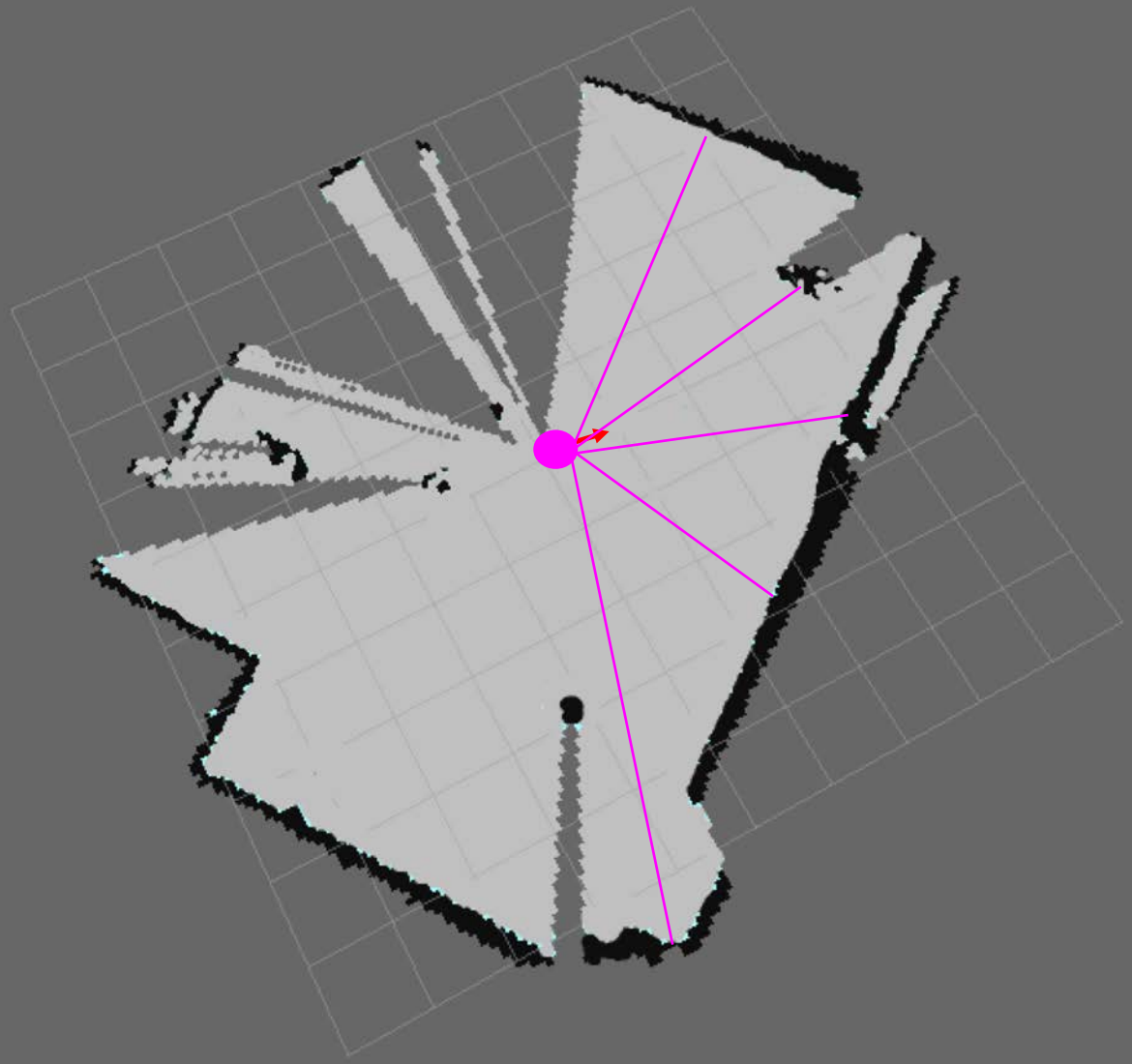
Motion Model

Applying the measured control input to the motion model (with noise), yields an updated set of possible poses.



Sensor Model

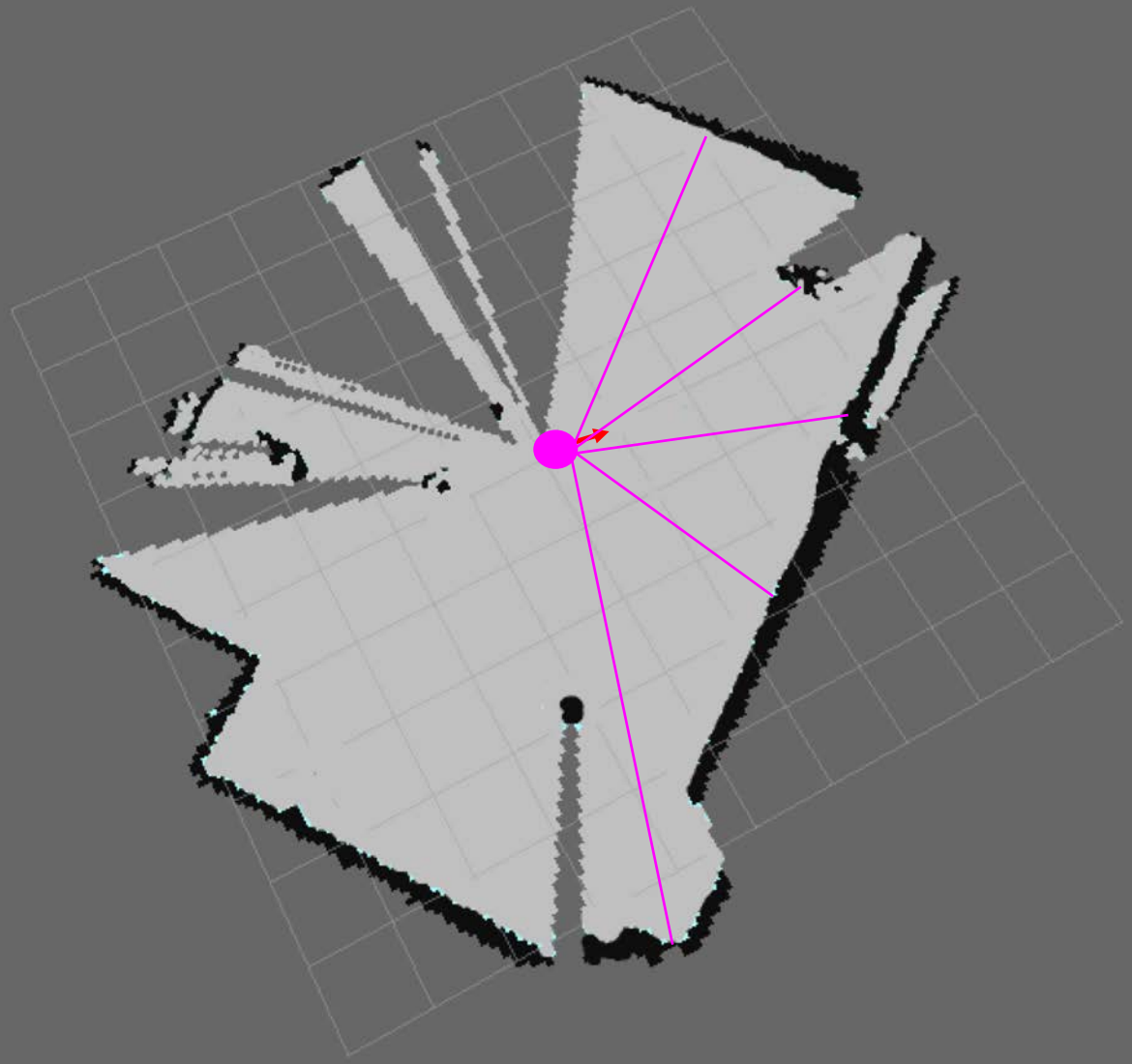
For each particle we can create a ‘fake’ laser scan by raycasting against the map at the particles pose.



Questions

How hard is creating the fake laser scan relative to computing the motion model update?

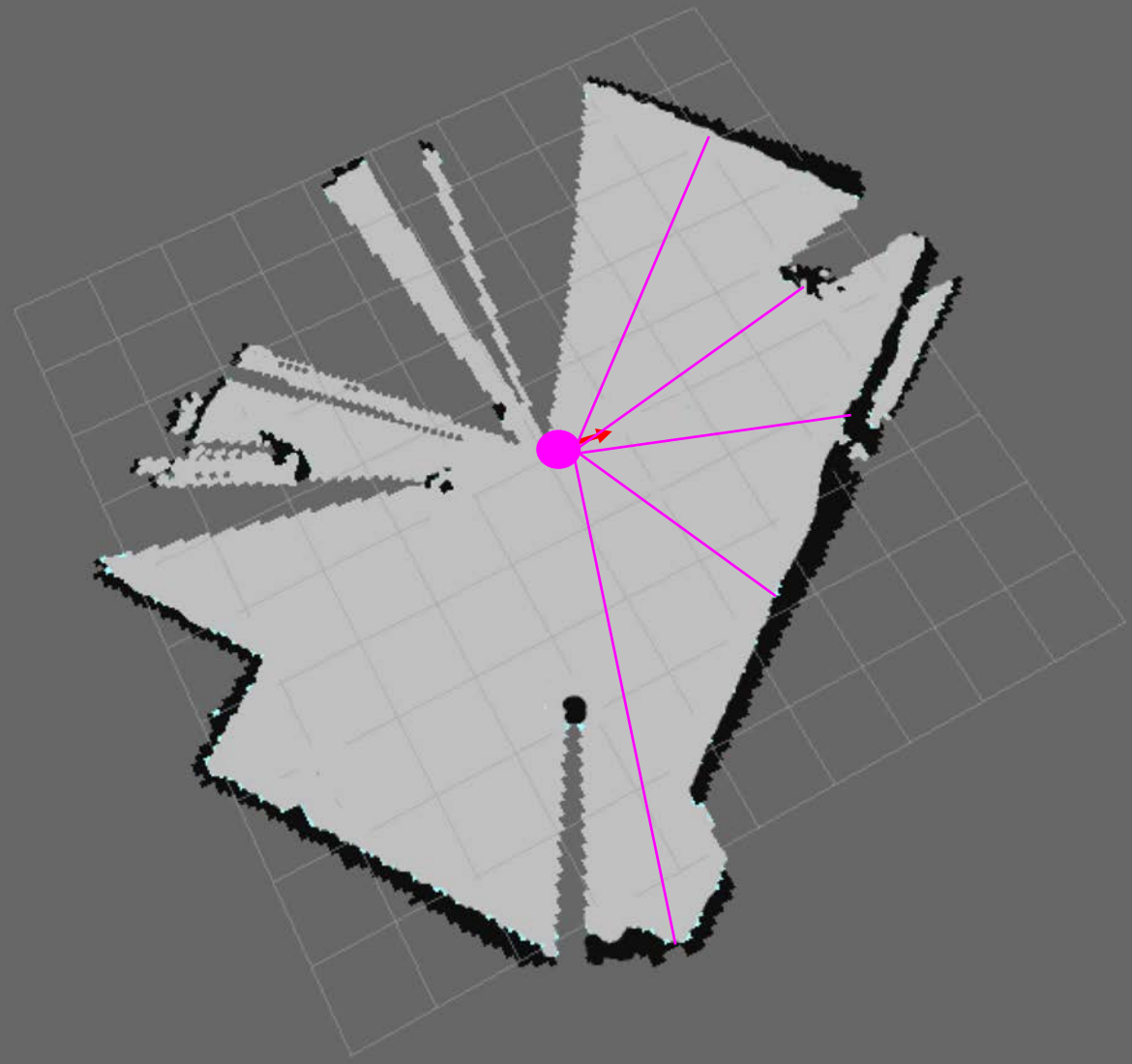
What is special about the sensor model?



Questions

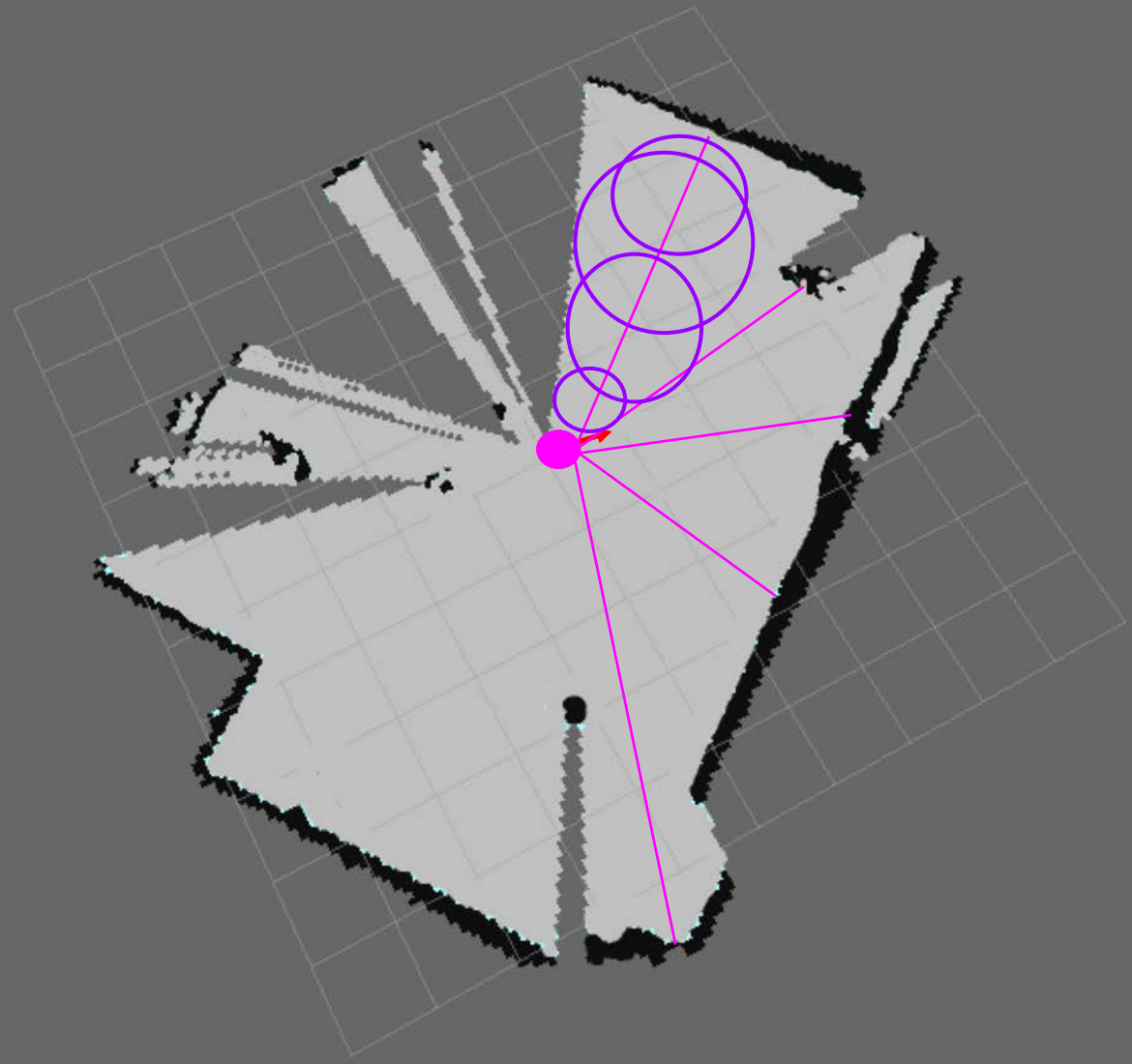
How hard is creating the fake laser scan relative to computing the motion model update?
Way harder.

What is special about the sensor model?
It's embarrassingly



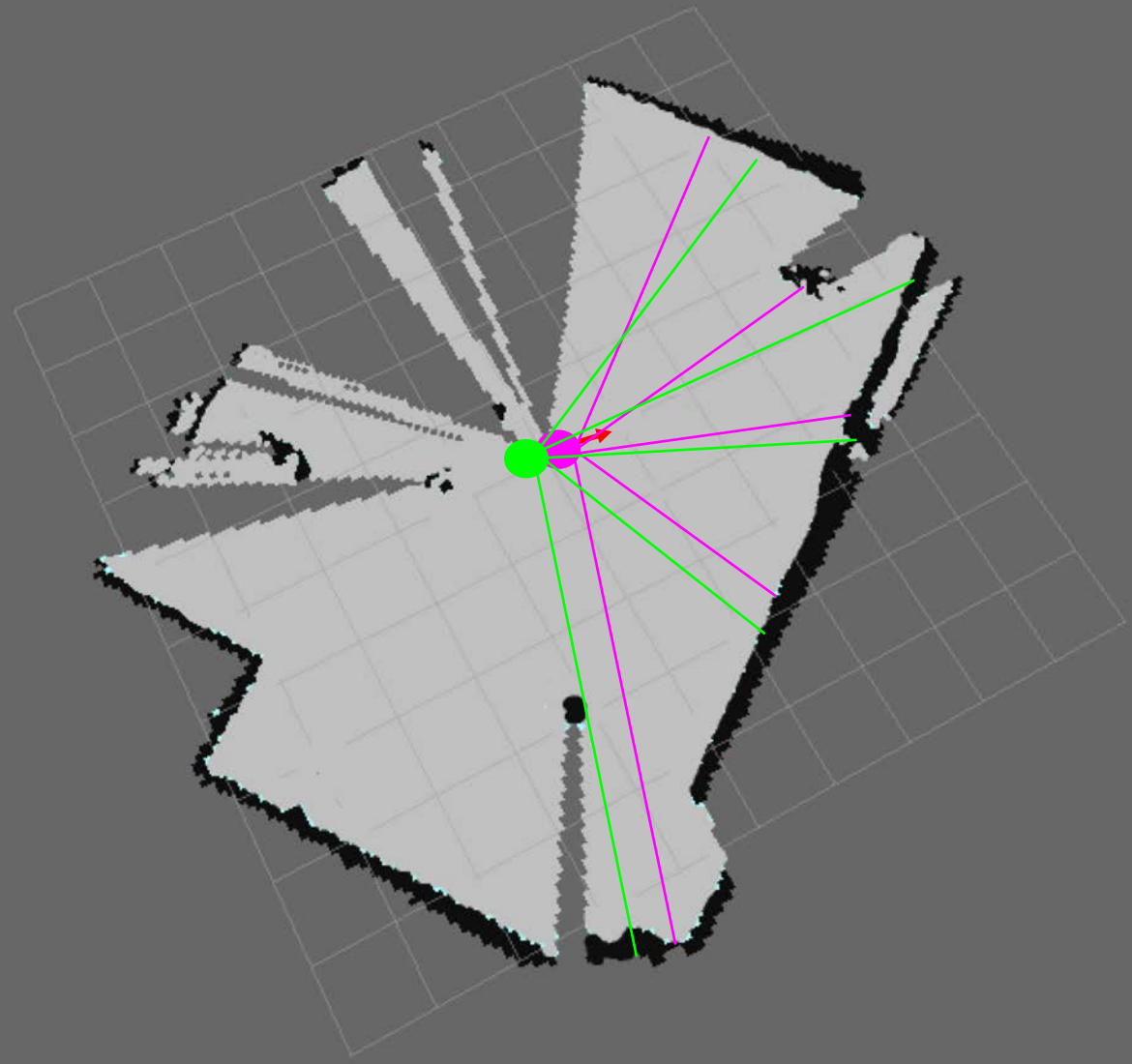
Raycasting

Efficient methods for raycasting such as Breshenham's Line method can be deployed. Alternately a signed distance field can be pre-computed.



Computing Particle Weights

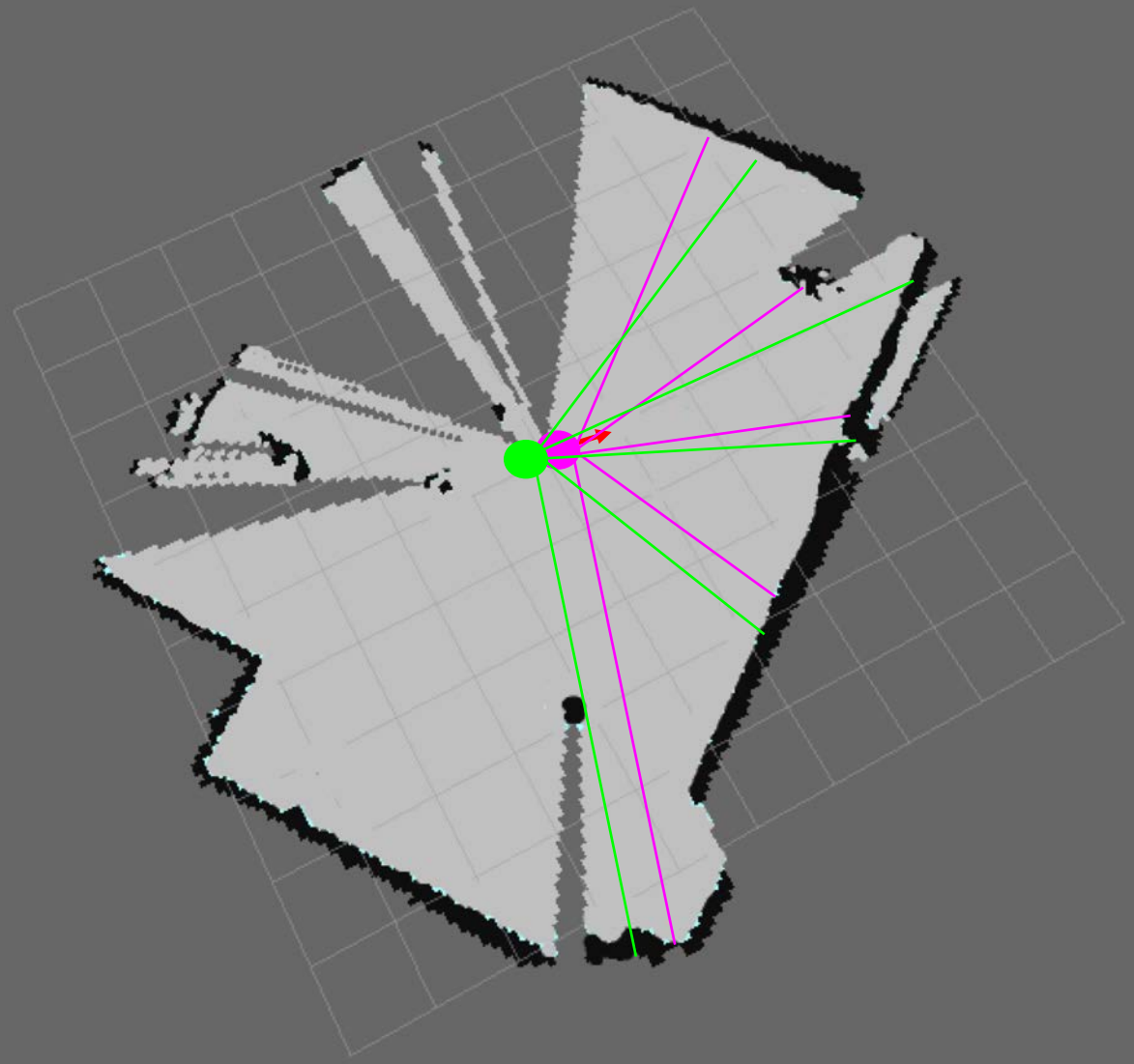
Recall, we have the ‘real’ laser scan which the car observed (shown in green). Note we don’t know where the green dot is but we do know the range measurements.



Computing Particle Weights

We can compute a score for the fake laser scan:

$$p(o_t \mid x_t^i) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(o_t - r^i)^2}{2\sigma^2}}$$

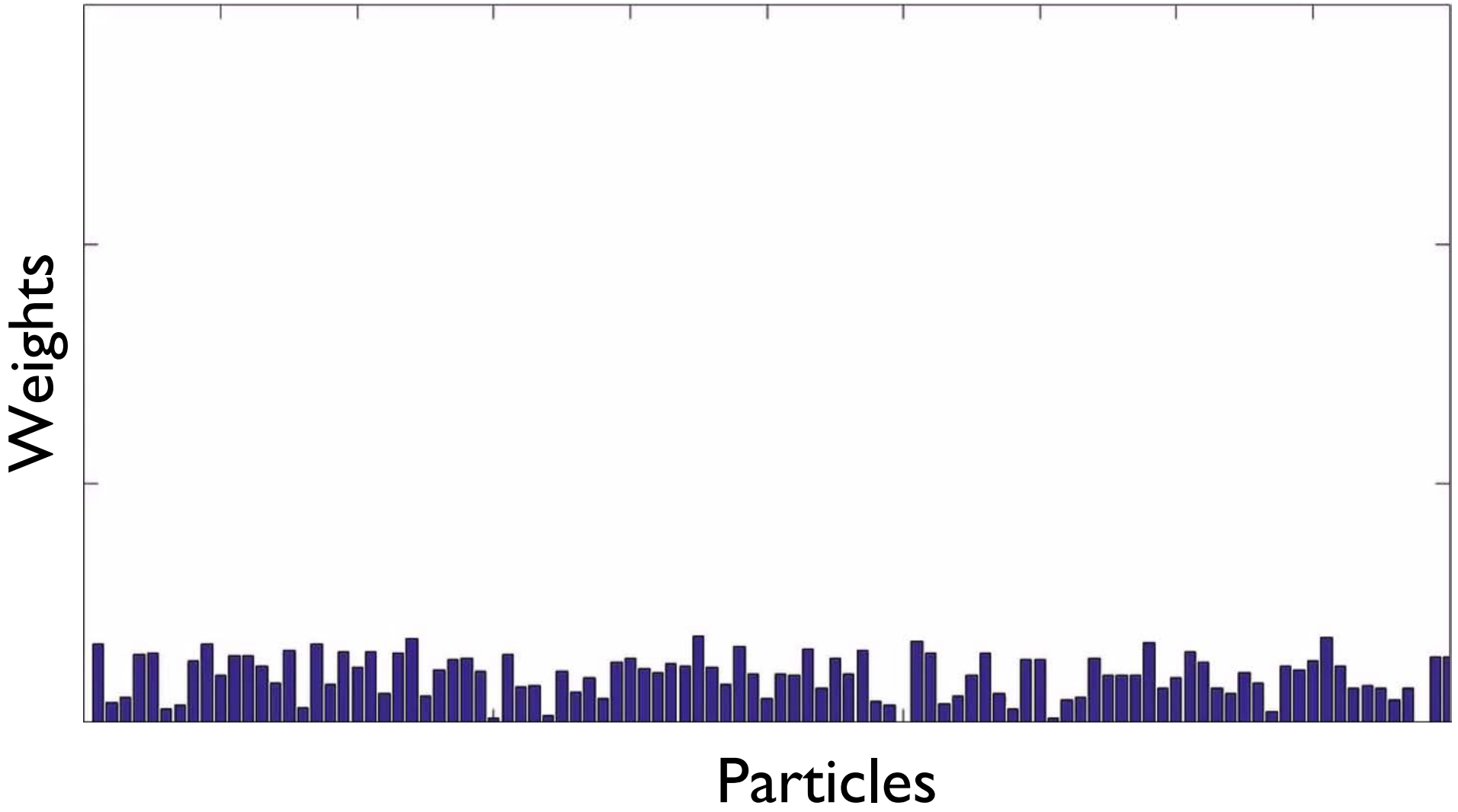


The Algorithm

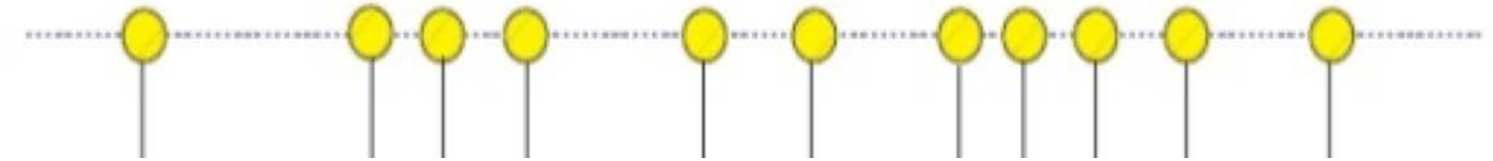
```
# a single recursive update step of the MCL algorithm
def MCL(X_{t-1}, a_{t-1}, o):
    X_t = {}
    for i in range(m):
        # sample a pose from the old particles according to old weights. Samples
        # implicitly represent the prior prob. dist. Bel(x_{t-1}) in eqn. (3)
        x^i_{t-1} ~ X_{t-1}
        # update the sampled pose according to the motion model
        x^i_t ~ p(x_t | x_{t-1}, a_{t-1})
        # weight the updated pose according to the sensor model
        w^i_t = p(o | x^i_t)
        # add the new pose and weight to the new distribution
        X_t = X_t ∪ { (x^i_t, w^i_t) }
    # normalize weights, should sum to 1
    X_t = normalize(X_t)
    return X_t

# iterative application of the MCL algorithm
def particle_filter():
    X = Bel(x_o) ← initial particles
    while true:
        a = get_last_odometry()
        o = get_last_sensor_readings()
        X = MCL(X,a,o)
        # inferred pose ← expected value over particle distribution
        pose = Ex[X]
```

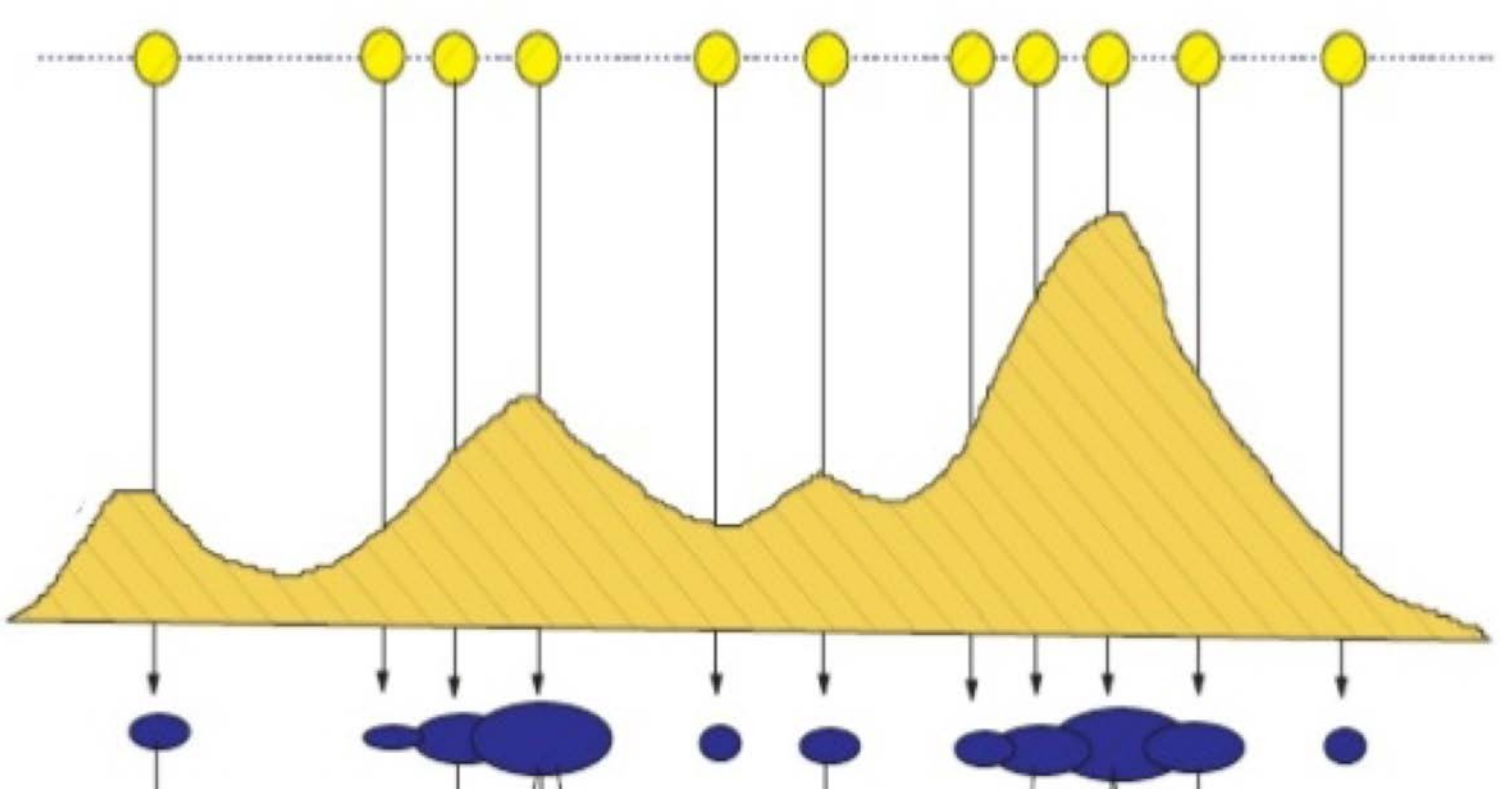
Particle Filter without Resampling



Resampling



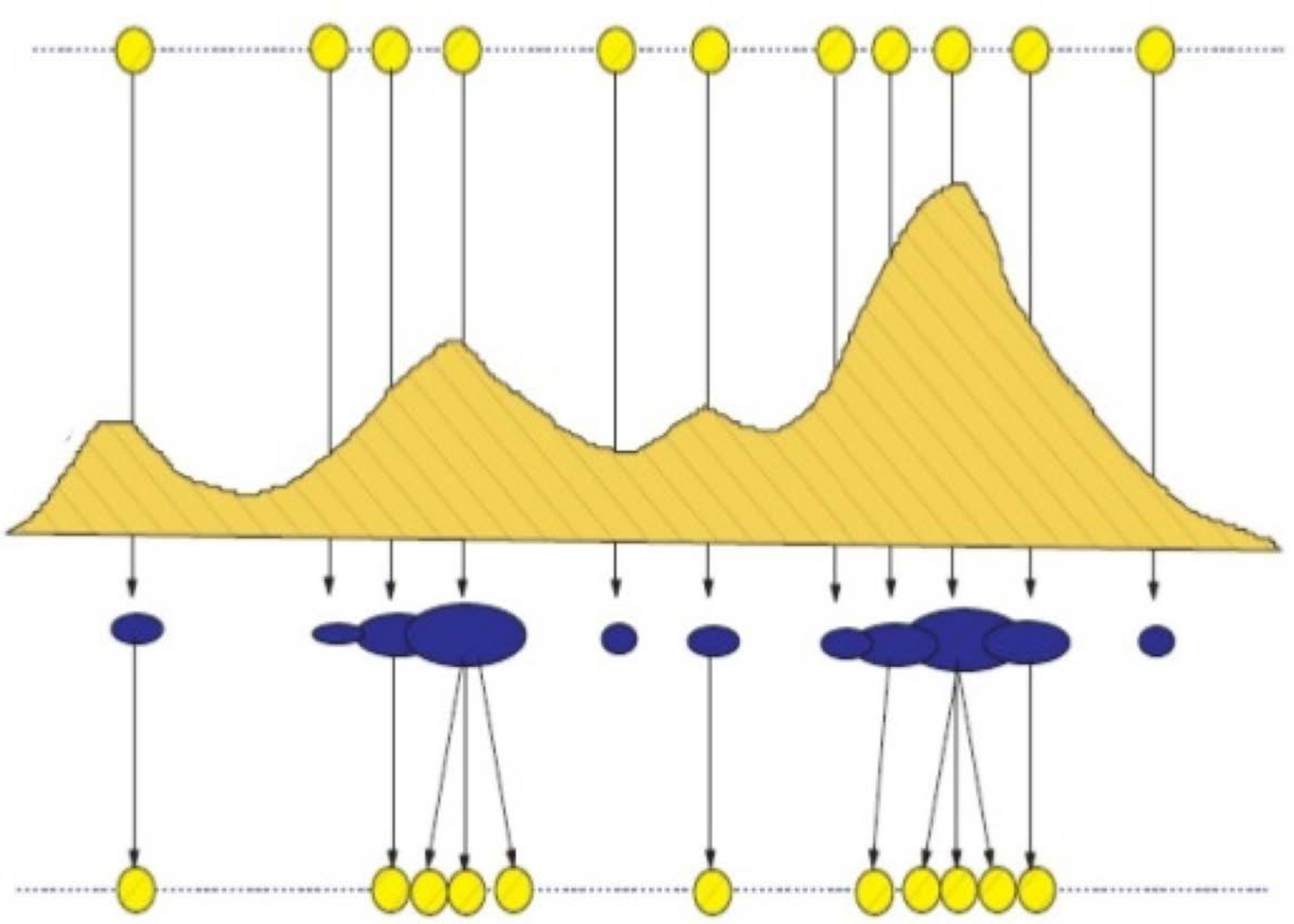
Resampling



Original Particles

After N iterations

Resampling

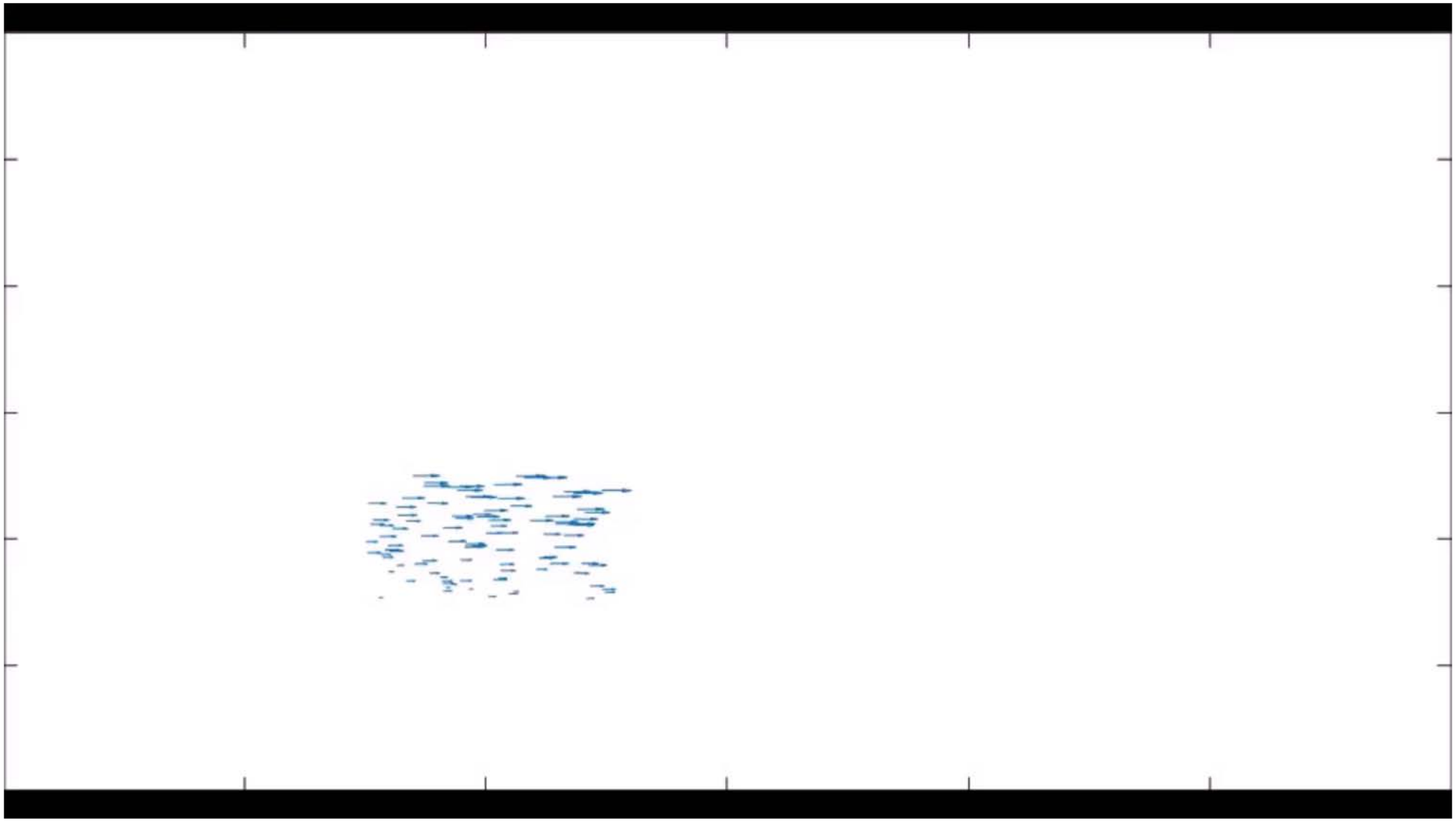


Original Particles

After N iterations

Resampling

Particles



Particle Filters in ROS

- Adaptive Monte Carlo Localization Package
- Localization for a robot moving in a 2D space
- Localizes against a pre-existing map

AMCL Parameters

`min_particles`

Default: 100

The minimum number of particles to be used for calculating correlation

`max_particles`

Default: 500

The maximum number of particles to be used for calculating correlation

AMCL Parameters

`update_min_d`

Default: 0.2m

The minimum translation movement required by the vehicle before an pose update is published

`update_min_a`

Default: $\pi/_6$ radians

The minimum angular movement required by the vehicle before an pose update is published

AMCL Parameters

`initial_pose_x`	Default: 0
`initial_pose_y`	Default: 0
`initial_pose_a`	Default: 0

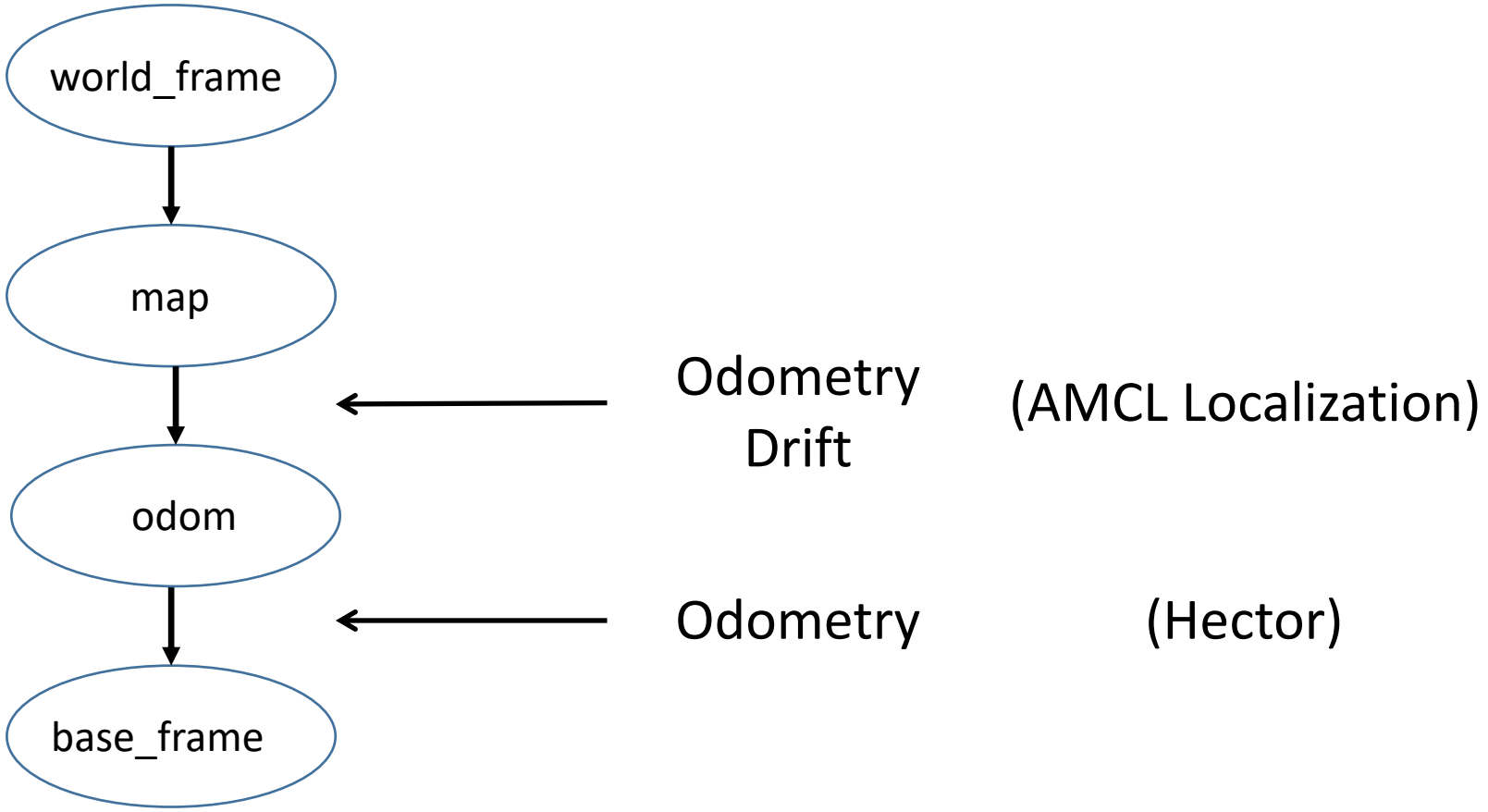
The initial mean position of the particles to initialize the particle filter

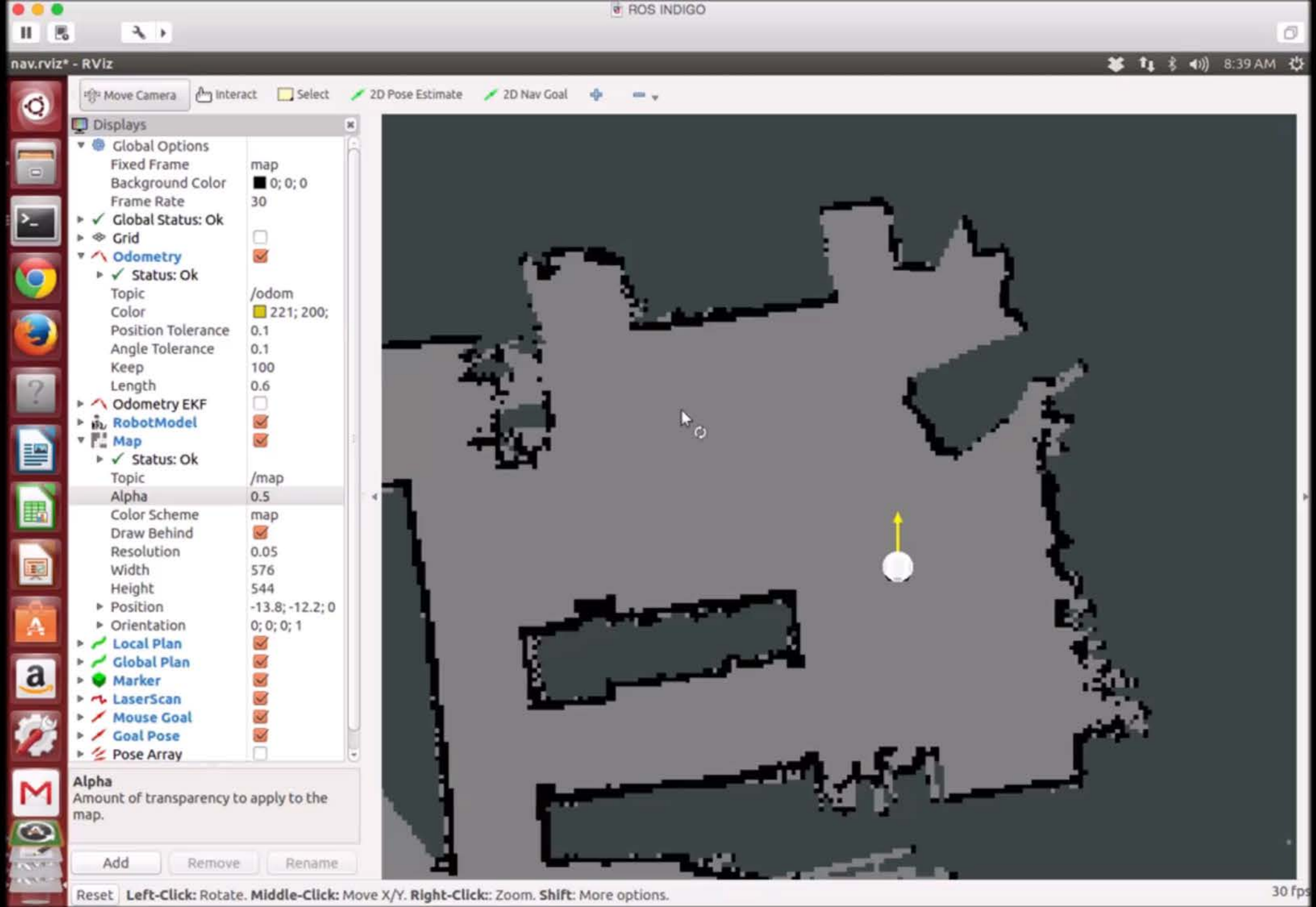
AMCL Parameters

`initial_cov_xx`	Default: 0
`initial_cov_yy`	Default: 0
`initial_cov_aa`	Default: 0

The covariance of particles distributed around the mean

Tf tree – Where does AMCL fit in




ROS INDIGO
nav.rviz* - RViz
8:39 AM
Move Camera
Interact
Select
2D Pose Estimate
2D Nav Goal
Displays
Global Options
Fixed Frame map
Background Color 0; 0; 0
Frame Rate 30
Global Status: Ok
Grid
Odometry
Status: Ok
Topic /odom
Color 221; 200;
Position Tolerance 0.1
Angle Tolerance 0.1
Keep 100
Length 0.6
Odometry EKF
RobotModel
Map
Status: Ok
Topic /map
Alpha 0.5
Color Scheme map
Draw Behind
Resolution 0.05
Width 576
Height 544
Position -13.8; -12.2; 0
Orientation 0; 0; 0; 1
Local Plan
Global Plan
Marker
LaserScan
Mouse Goal
Goal Pose
Pose Array
Alpha
Amount of transparency to apply to the map.
Add
Remove
Rename
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.
30 fps

amcl_config.rviz* – RViz

2D Pose Estimate | 2D Nav Goal | Publish Point

- Grid
- Odometry
  - Status: Warn
  - Topic: /amclOdom
  - Color: 15; 31; 255
  - Position Tolerance: 0.1
  - Angle Tolerance: 0.1
  - Keep: 1
  - Length: 1
- Map
- Map
- Pose
- PoseArray
- Odometry

Keep
Number of arrows to keep before removing the oldest. 0 means keep all of them.

Add | Remove | Rename

Time

ROS Time: 1458504970.57 | ROS Elapsed: 7.28 | Wall Time: 1460587807.26 | Wall Elapsed: 153.93 | Experimental

Reset | **Left-Click:** Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:** Zoom. **Shift:** More options.

30 fps

References

S. Thrun, W. Burgard. "Probabilistic Robotics." Chapter 4 and Chapter 8.
http://www.probabilistic-robotics.org/

S. Thrun. "Artificial Intelligence for Robotics, Lesson 3." Udacity.
https://www.udacity.com/course/artificial-intelligence-for-robotics--cs373

S. Thrun, D. Fox, W. Burgard and F. Dellaert. "Robust Monte Carlo Localization for Mobile Robots." Artificial Intelligence Journal. 2001.
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.71.6016&rep=rep1&type=pdf

D. Fox, W. Burgard, and S. Thrun. "Markov localization for mobile robots in dynamic environments," Journal of Artificial Intelligence Research , vol. 11, pp. 391427, 1999.
http://www.jair.org/media/616/live-616-1819-jair.pdf

D. Fox. "KLD-sampling: Adaptive particle filters," Advances in Neural Information Processing Systems 14 (NIPS), Cambridge, MA, 2002. MIT Press.
https://papers.nips.cc/paper/1998-kld-sampling-adaptive-particle-filters.pdf

D. Bagnell "Particle Filters: The Good, The Bad, The Ugly"
http://www.cs.cmu.edu/~16831-f12/notes/F14/16831_lecture05_gseyfarth_zbatts.pdf

C. Walsh, S. Karaman. "CDDT: Fast Approximate 2D Ray Casting for Accelerated Localization." Arxiv, 2017.
http://arxiv.org/abs/1705.01167