SLAM-Particle Filter

Project 4

2016-03-03

Contents

1. Mapping

   :Occupancy grid mapping

2. Localization

   : Map Registration using Range Sensor

3. PF-based SLAM

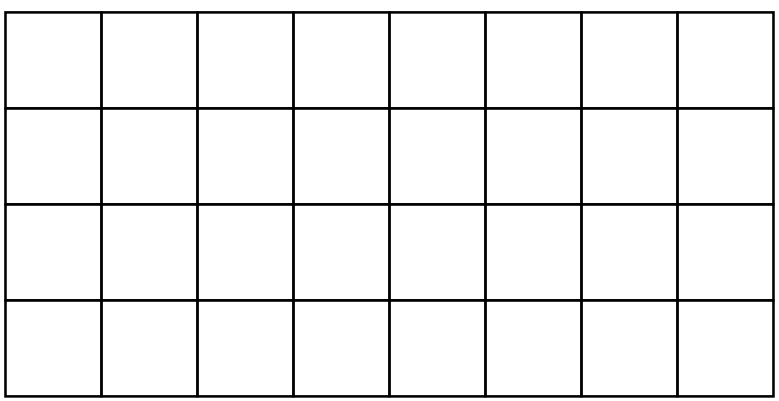
4. Project 4

1. Mapping

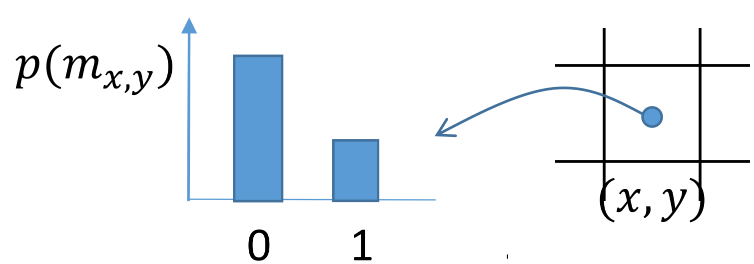
[Reference] Sebastian Thrun, “Robotic Mapping: A Survey”, 2003.

Occupancy Grid Mapping

- Occupancy: binary R.V. $m_{x,y}: \{free, occupied\} \rightarrow \{\,0\,,1\,\}$
- Occupancy grid map
  : fine-grained grid with occupancy variable associated with cell
- Bayesian filtering
- Usually based on a range sensor



For each cell,
we update $p(m_{x.y}|z)$



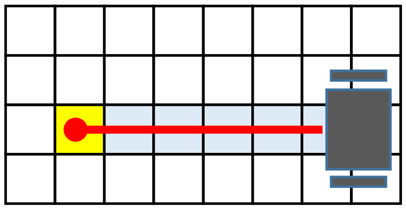
Occupancy Grid Mapping

- Measurement

Free

Occupied

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



- Measurement model

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

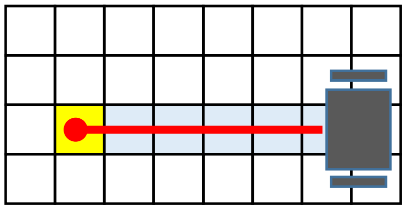
Occupancy Grid Mapping

- Measurement

$$z \sim \{ \underbrace{-1}_{\text{Free}}, \underbrace{1}_{\text{Occupied}} \}$$

- Measurement model

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



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

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

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

$$p(z = -1|m_{x,y} = 0) =$$

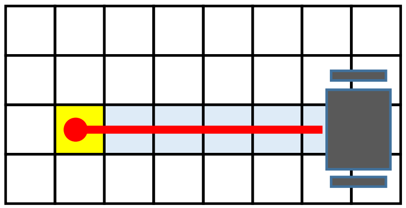
Occupancy Grid Mapping

- Measurement

Free

Occupied

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



- Measurement model

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

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

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

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

$$p(z = -1|m_{x,y} = 0) = 1 - p(z = 1|m_{x,y} = 0)$$

Occupancy Grid Mapping

- Odd

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

Occupancy Grid Mapping

- Log-odd

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

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

(Log odd) (Log LH ratio) (Log prior ratio)

(log odd) ← (log odd) + (log meas model ratio)

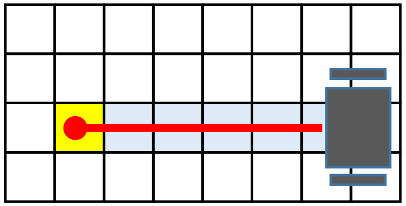
Occupancy Grid Mapping

- Example

Measurement Model

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

$$\log odd\_occ := \log \frac{p(z=1|m_{x,y}=1)}{p(z=1|m_{x,y}=0)} = \log \frac{0.7}{0.2}$$



$$\log odd\_free := \log \frac{p(z=-1|m_{x,y}=0)}{p(z=-1|m_{x,y}=1)} = \log \frac{0.8}{0.3}$$

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

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

Occupancy Grid Mapping

- Example (continued)

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

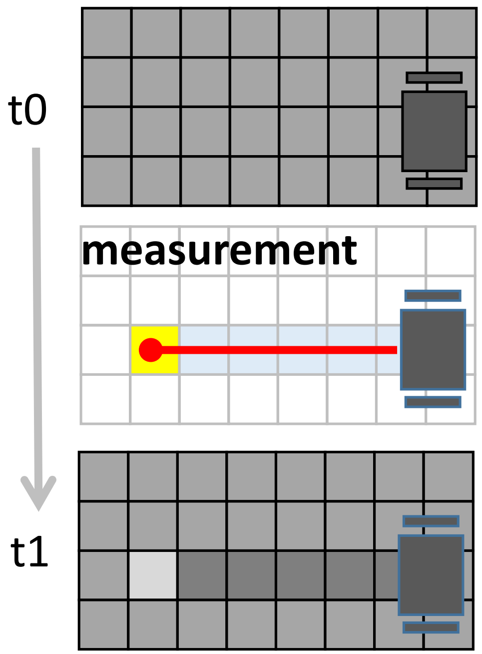
<u>Case I : cells with z=1</u>

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

$$\log odd \leftarrow 0 + \log(\frac{0.7}{0.2})$$

<u>Case II : cells with z=-1</u>

<u>Case III : cells with no z</u>



Occupancy Grid Mapping

- Example (continued)

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

<u>Case I : cells with z=1</u>

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

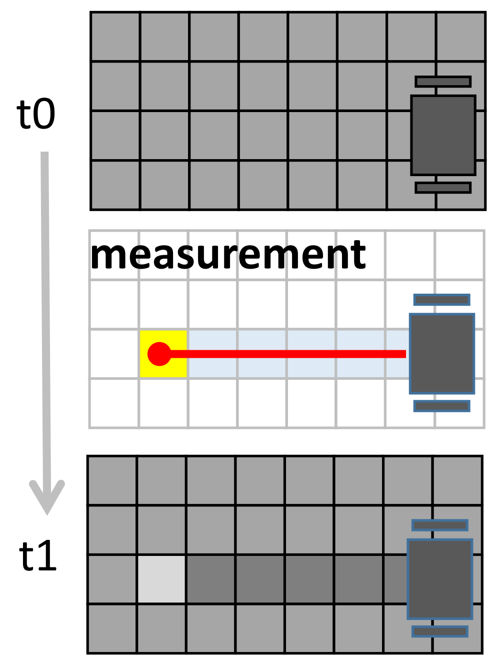
$$\log odd \leftarrow 0 + \log(\frac{0.7}{0.2})$$

<u>Case II : cells with z=-1</u>

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

$$\log odd \leftarrow 0 - \log(\frac{0.8}{0.3})$$

<u>Case III : cells with no z</u>



Occupancy Grid Mapping

- Example (continued)

<u>Case I : cells with z=1</u>

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

$$\log odd \leftarrow 0 + \log(\frac{0.7}{0.2})$$

<u>Case II : cells with z=-1</u>

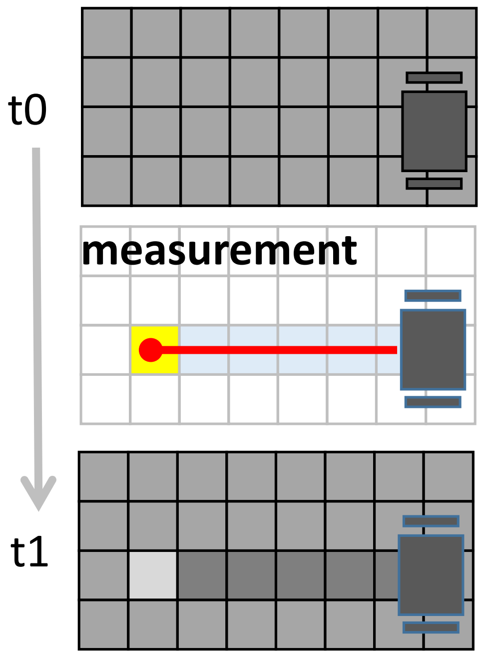
$$\log odd \leftarrow \log odd - \log odd\_free$$

$$\log odd \leftarrow 0 - \log(\frac{0.8}{0.3})$$

<u>Case III : cells with no z</u>

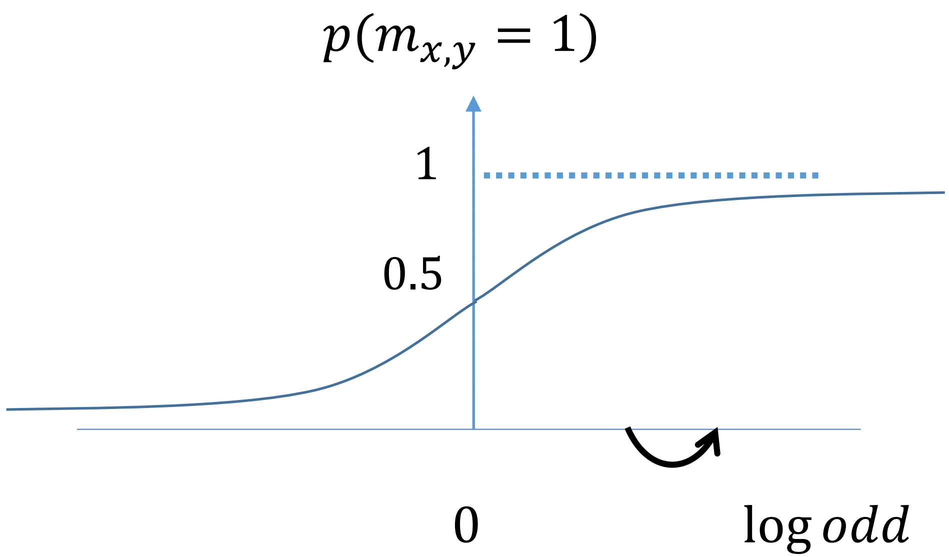
$$\log odd = 0$$

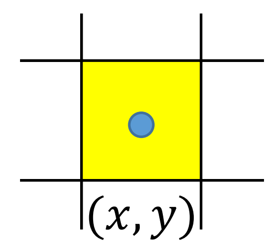
$$\log odd \mathrel{+}= \log \frac{p(z|m_{x,y}=1)}{p(z|m_{x,y}=0)}$$



Occupancy Grid Mapping

- In summary… for cells with z = 1

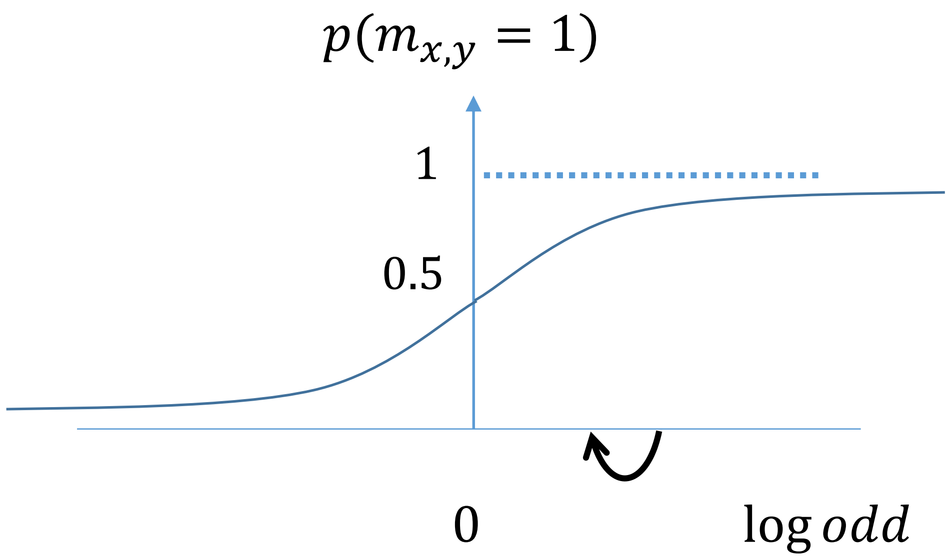


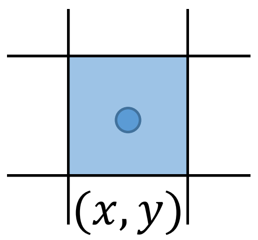


$$\log odd \mathrel{+}= \log odd\_occ$$

Occupancy Grid Mapping

- In summary… for cells with z = -1

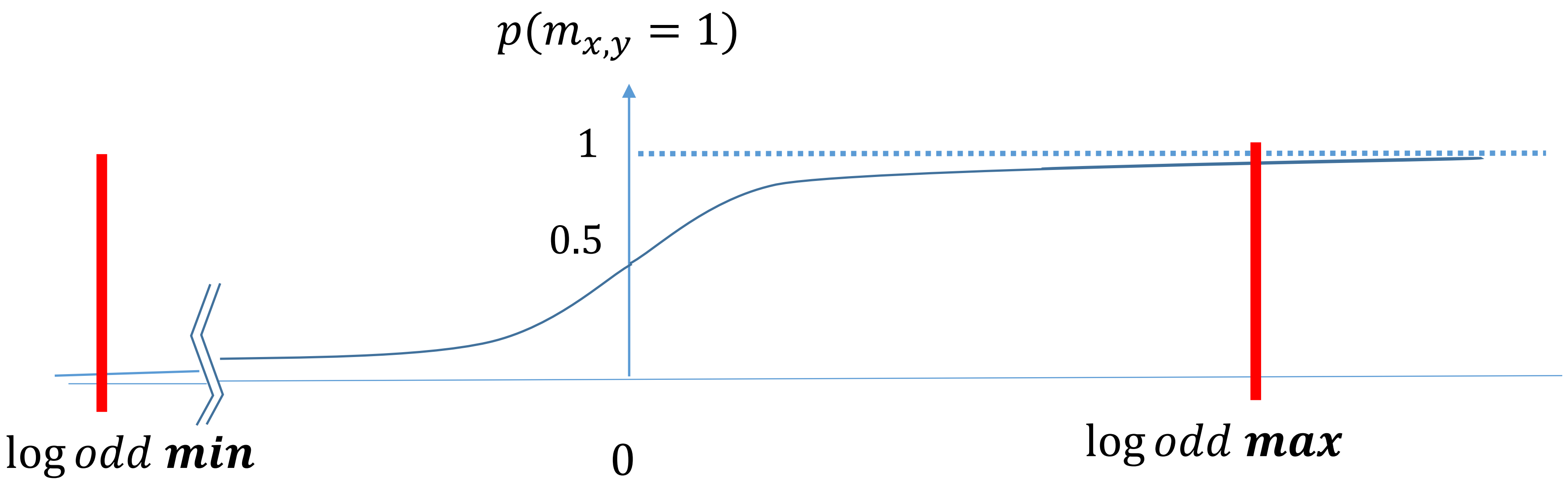




$$\log odd -= \log odd\_free$$

Occupancy Grid Mapping

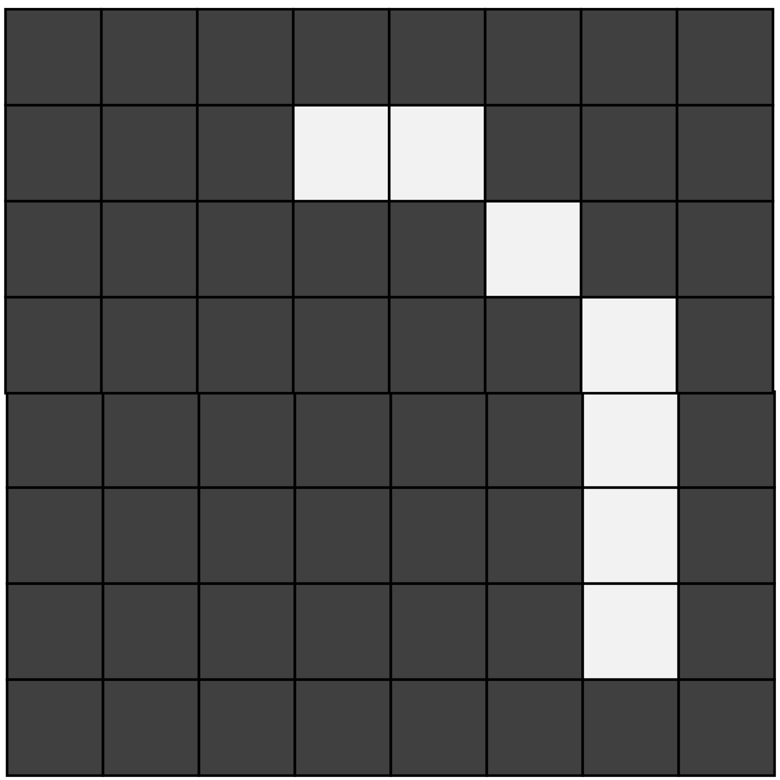
- Tips : Never make anything certain! Saturate log-odd.



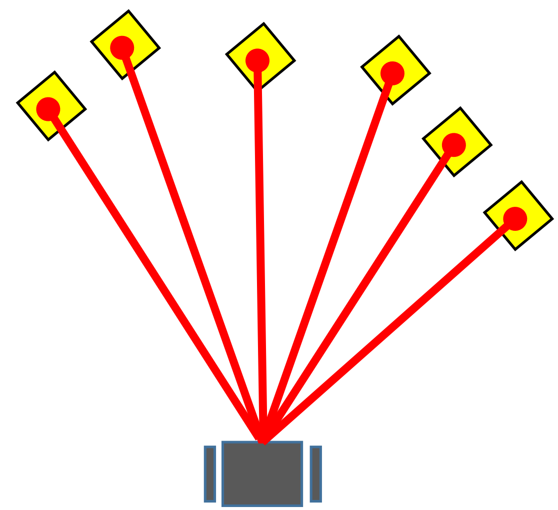
2. Localization

Map Registration

Map



Range measurement



Robot

Map Registration

- Correlation-based Matching

    1) General hypotheses

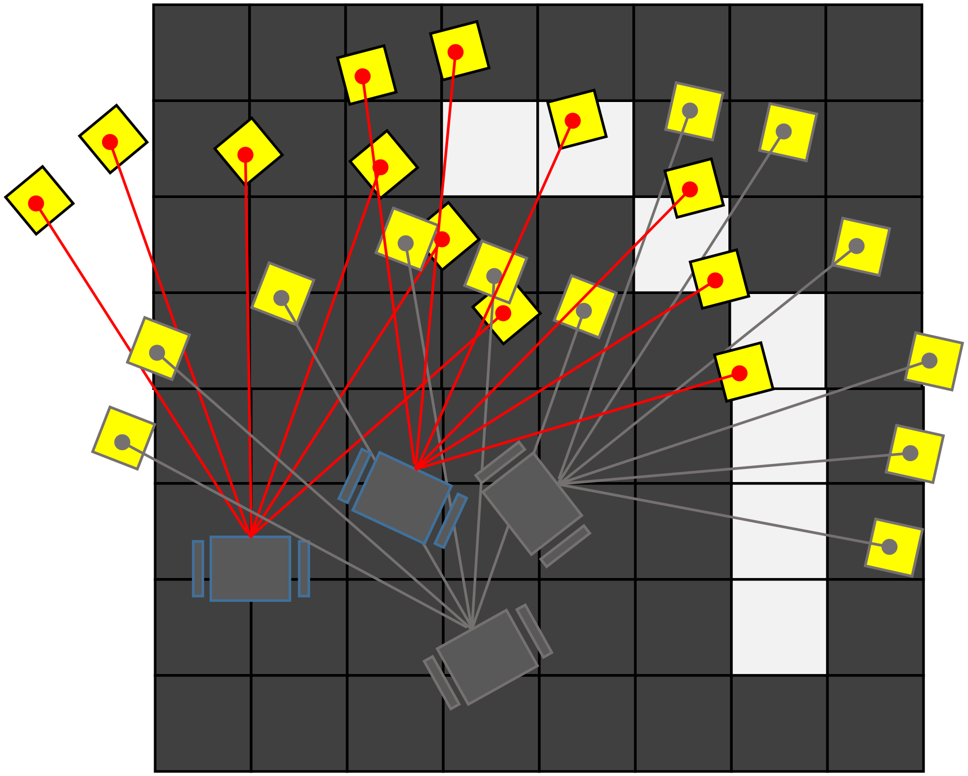
    2) Evaluate hypotheses

    3) Pick the best

Map Registration

- Correlation-based Matching

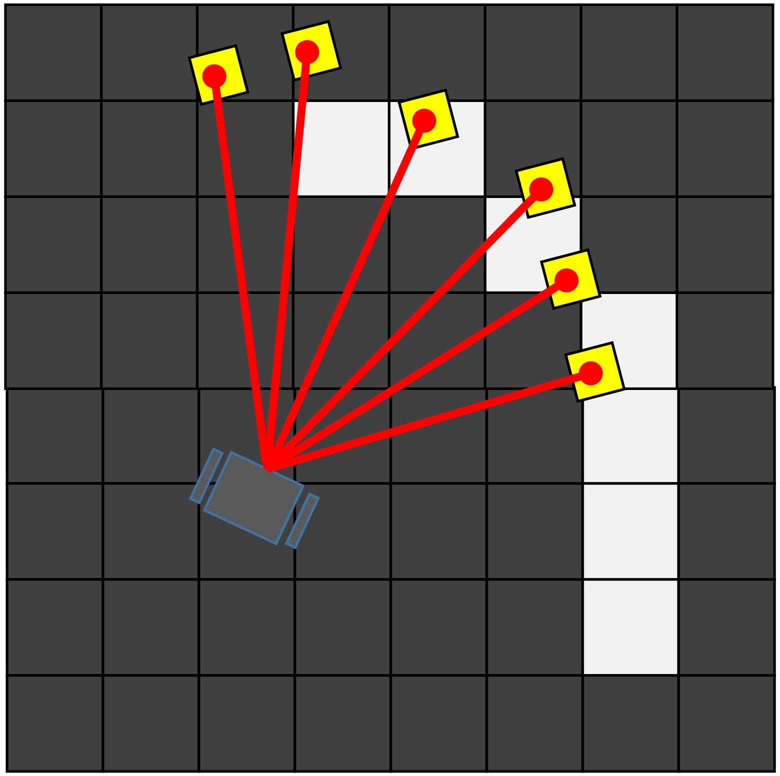
1) Generate hypotheses



Map Registration

- Correlation-based Matching

1) Generate hypotheses

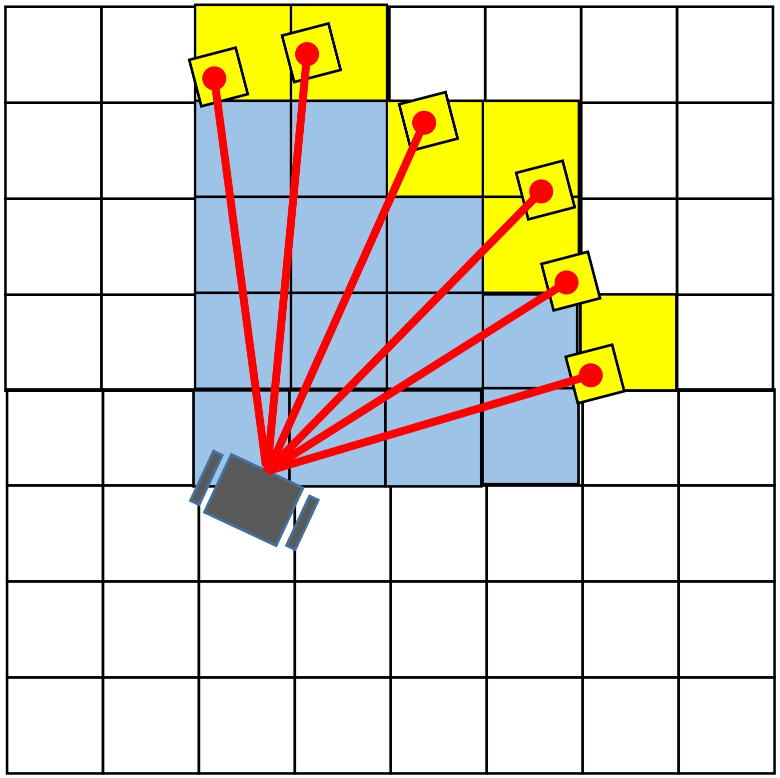


(For each hypothesis…)

Map Registration

- Correlation-based Matching

1) Generate hypotheses

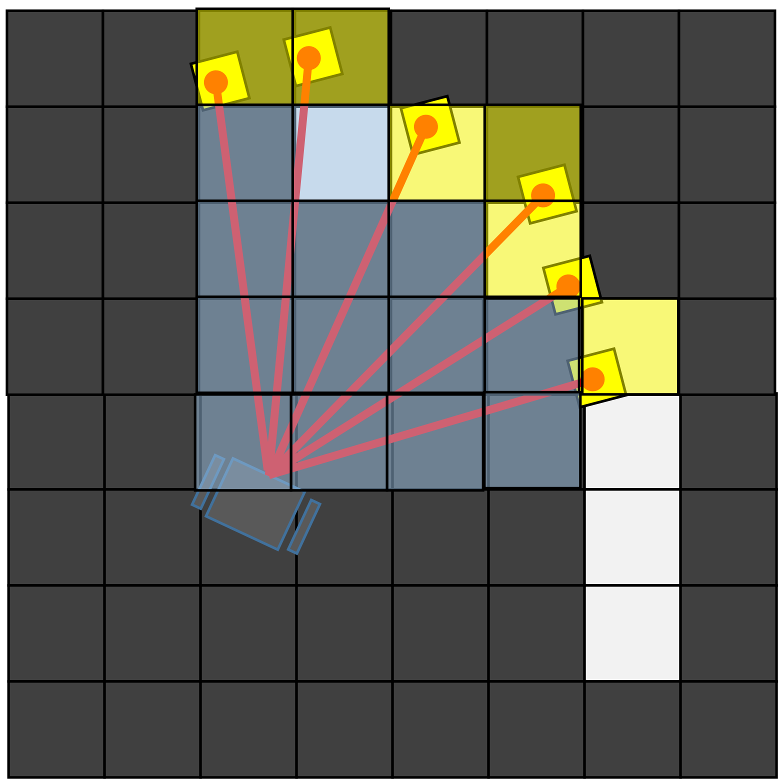


(Build a local map from the measurement in a form that can be compared with the global map)

Map Registration

- Correlation-based Matching

2) Evaluate hypotheses



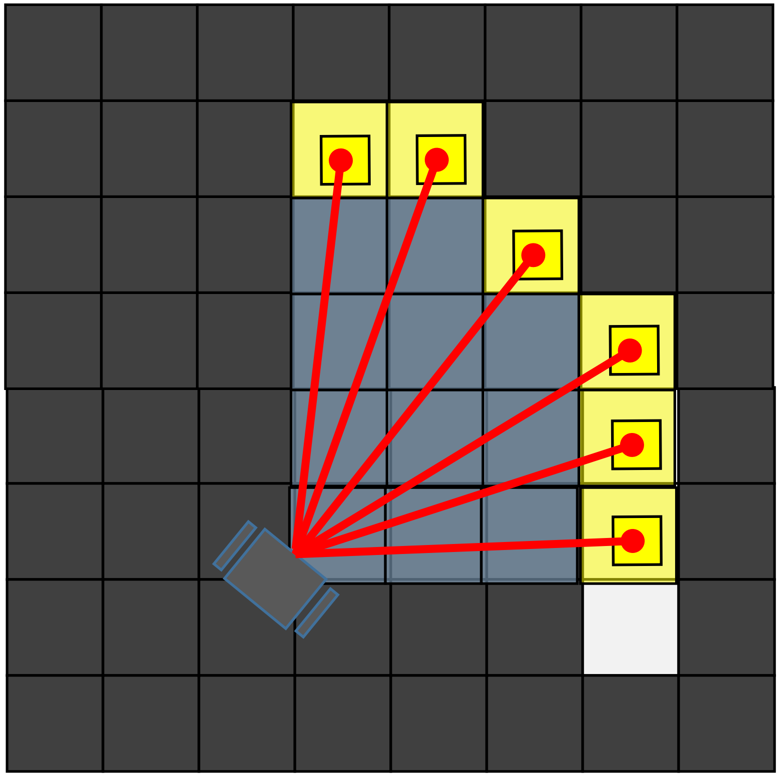
(Then score the hypothesis.)

$$\sum_{x,y} r_{x,y} LOR(m_{x,y})$$

Map Registration

- Correlation-based Matching

3) Find the best*



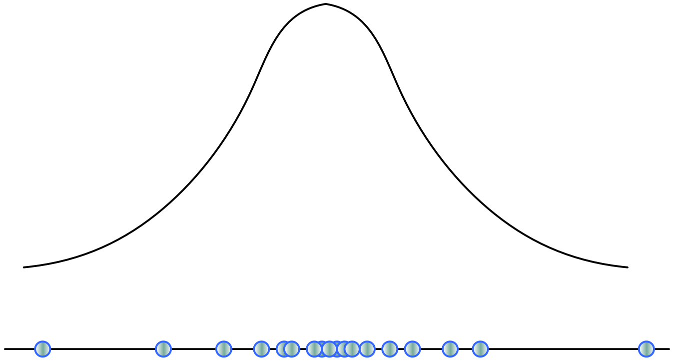
(Among all the hypotheses, choose the one that has the largest score in order to represent your current location.)

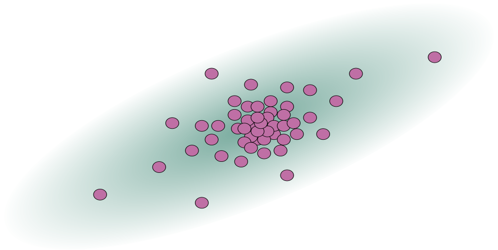
$$\max_{H} \sum_{x,y} r_{x,y} LOR(m_{x,y})$$

3. PF-based SLAM

Particle Filter

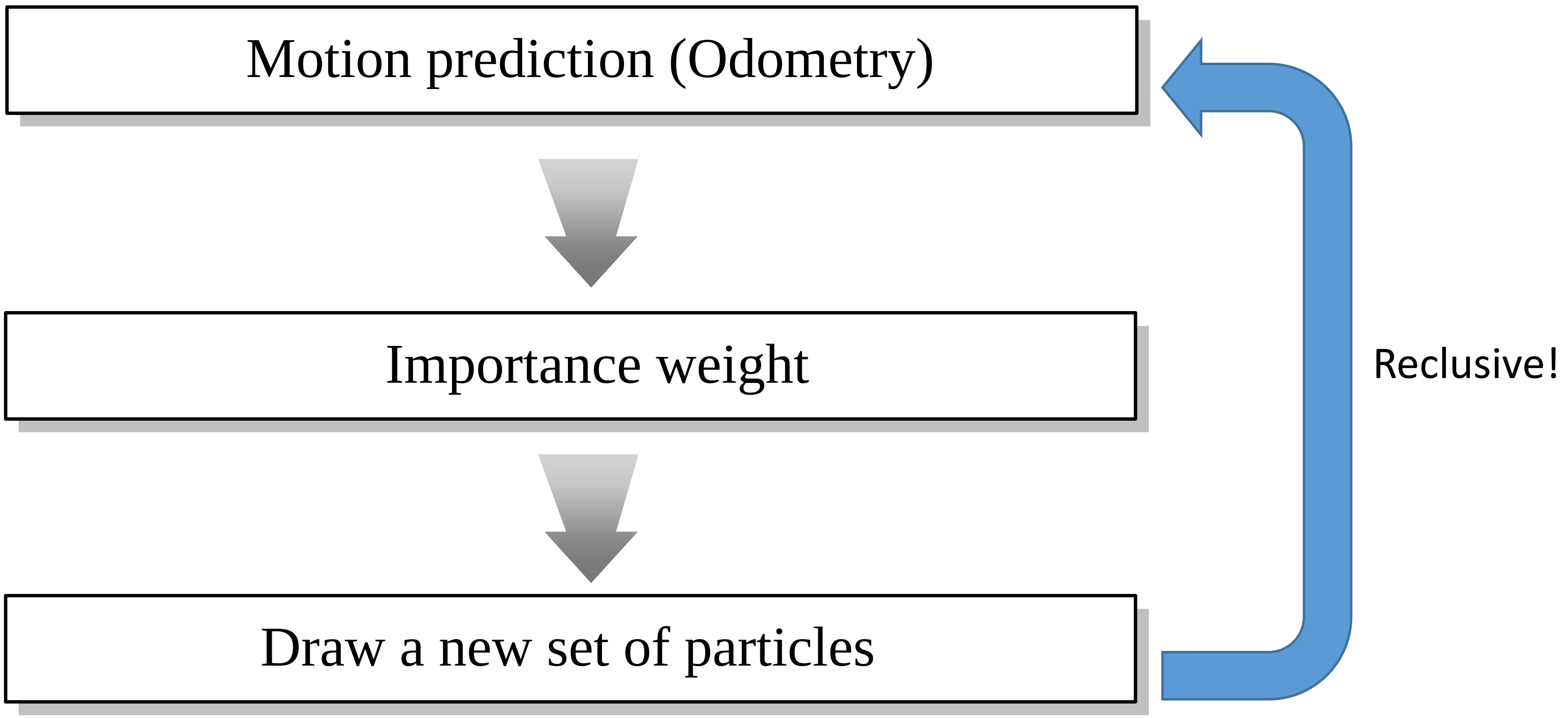
- Non-parametric model (multimodal)
  - Mixtures of Gaussians, multi-hypothesis Kalman Filter
- Uses particles instead of probability distribution
- Fast and efficient





Particle Filter

- Flowchart

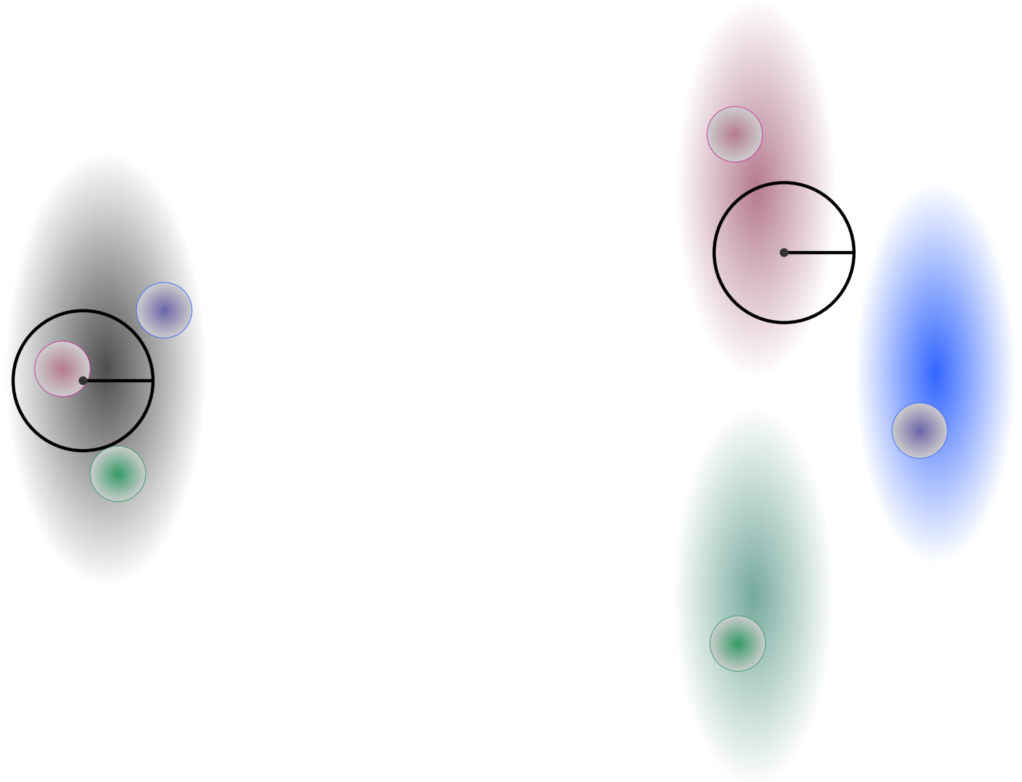


1st step: New pose from motion

New pose given new control

1^{st} step: New pose from motion

New pose given new control

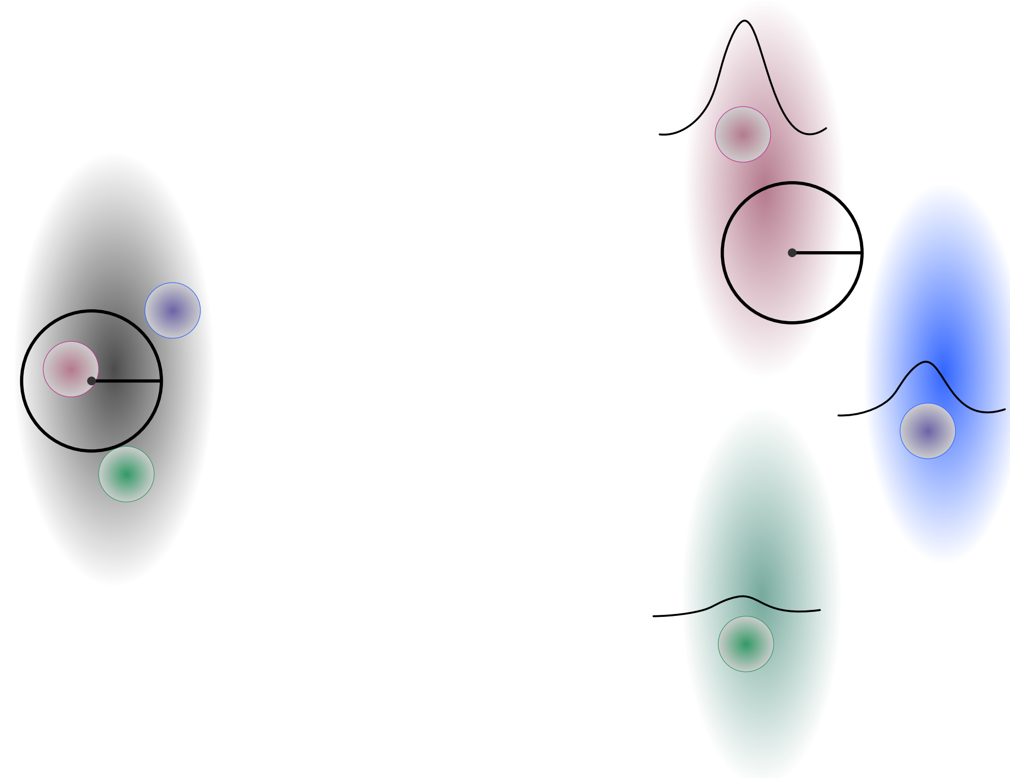


2nd step: Importance weight

New pose given new control

↓

Importance weight



3rd step: Resampling

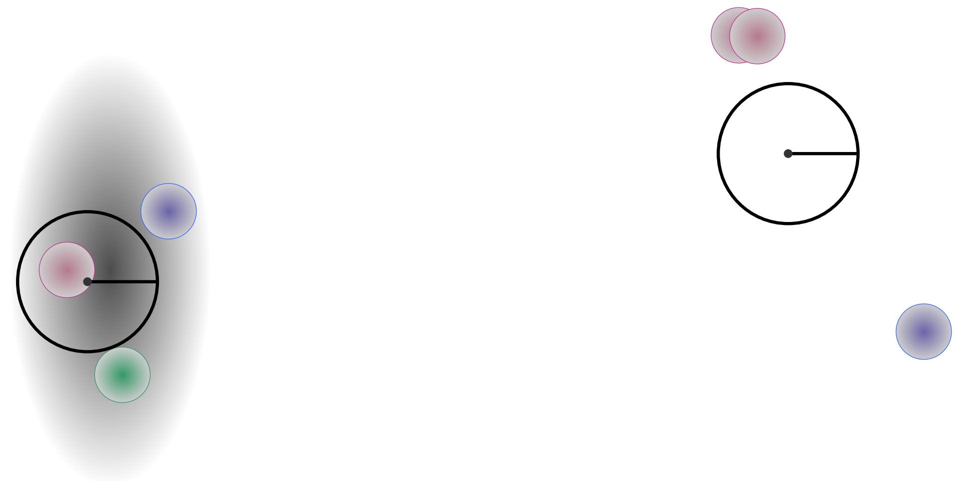
New pose given new control

↓

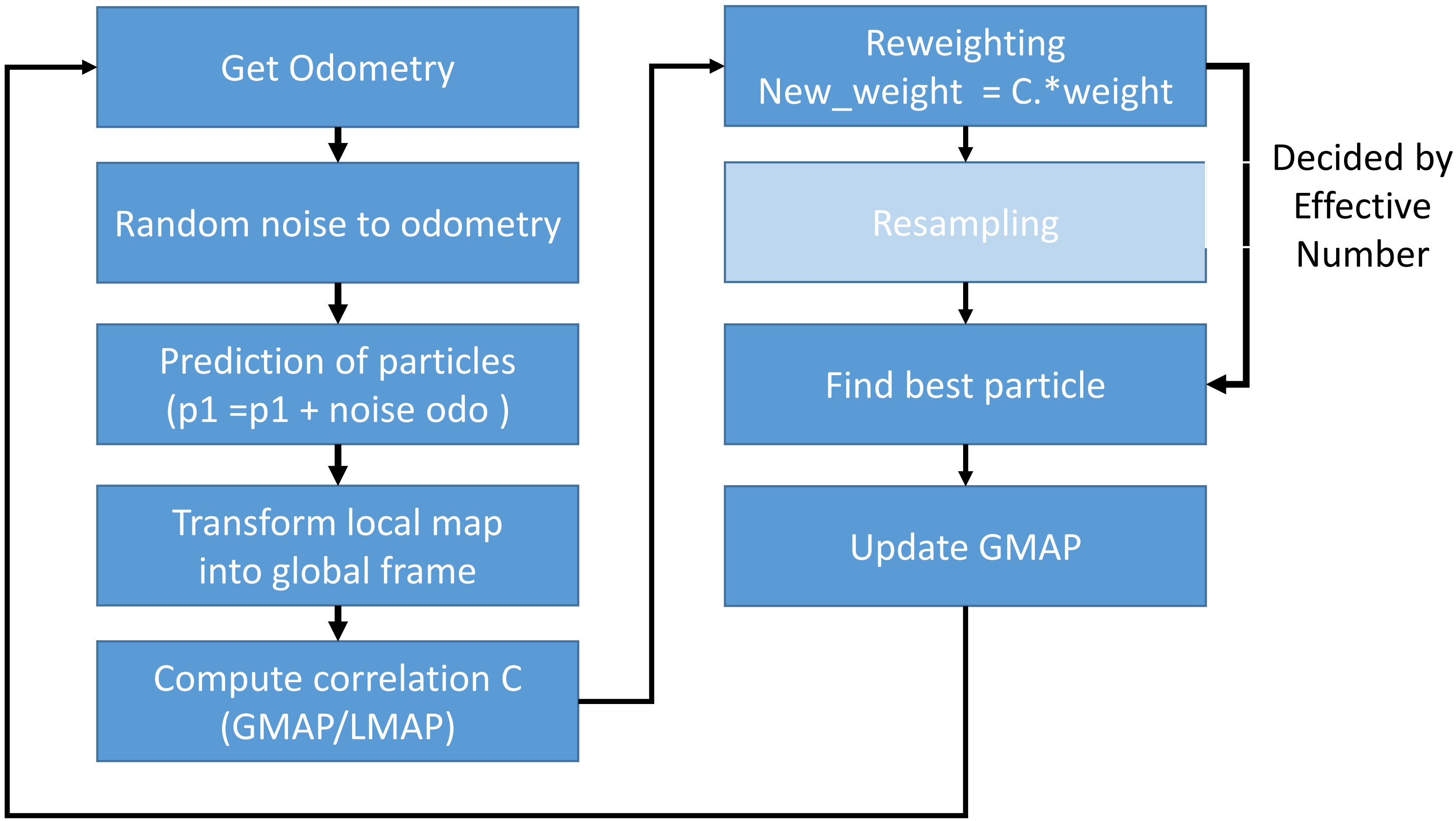
Importance weight

↓

Draw a new set of particles



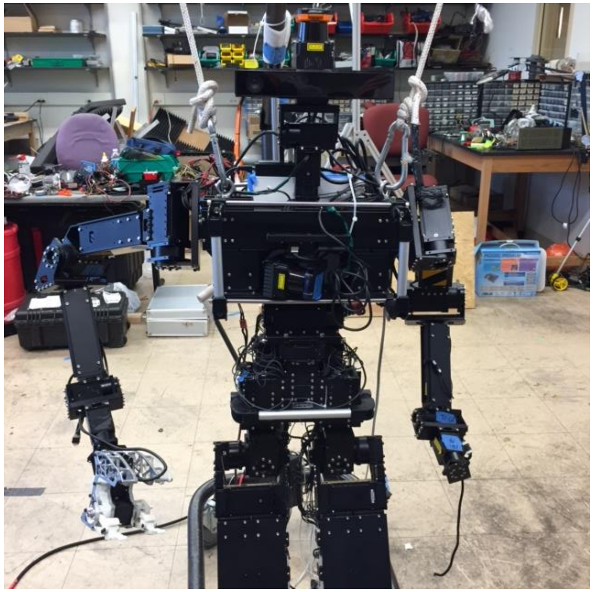
Flow diagram (PF)



Project #4

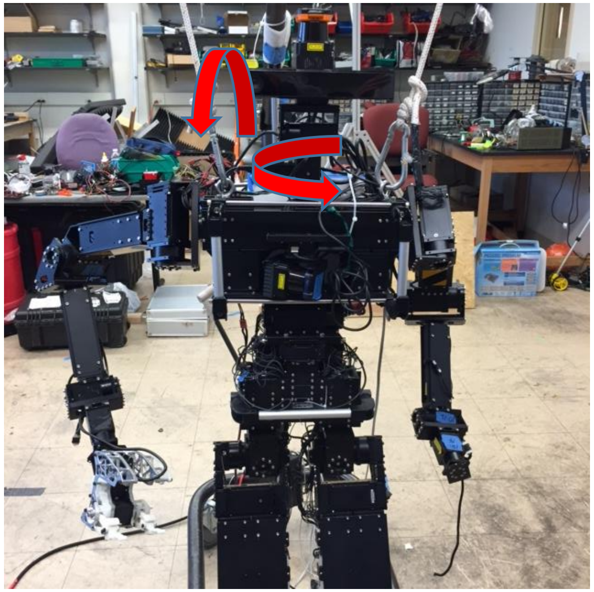
Project #4 SLAM-PF

- Humanoid SLAM
    - 2D SLAM based on several sensors
    - Odometry, Lidar (Head), IMU, Vision (Kinect)
- Ground Detection
    - Visualization of the detected ground
- Difficulties
    - 3D jerky motion of the THOR
    - Roll & Pitch motion
    - Head motion
    - Moving obstacles



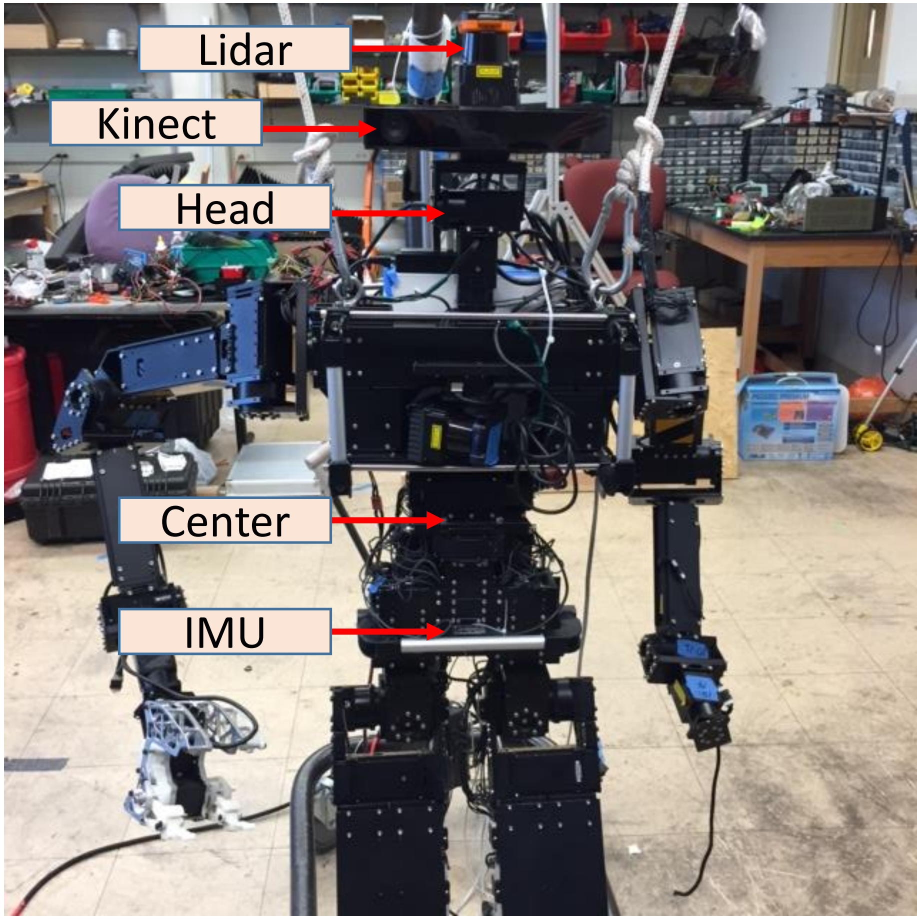
Project #4 SLAM-PF

- 3D jerky motion of the THOR
  - Roll and Pitch motion
  - Poor odometry
- Head motion while moving
  - Compensate roll and yaw motion
  - Remove the lidar scan of the ground
- Relative coordinate
  - IMU, Lidar, Kinect coordinates
- Etc.
  - Moving obstacles



Pre-processing

- Relative position
- Center of Mass kept at 0.93 meters
- Lidar: 41cm above Center of Mass
- IMU: 16.5cm below Center of Mass
- Head: 39.5cm above Center of Mass
- (roll and yaw angles are given)
- Kinect: 8.5cm above Head



Project #4 SLAM-PF

- Training Data set(#0 ~ #3)
  - Train_lidar.mat
  - Train_joint.mat
  - RGB.mat
  - Depth.mat

- cpp files
  - Map_correlation.cpp
  - GetMapCellsFromRay.cpp

Project #4 SLAM-PF

- Joint.mat
  - Joint angles
  - pos: Matrix of positions (Maybe you don’t need)
  - ts: Array of timestamps (relative time)
  - gyro: Matrix of gyro readings: figure(1);plot(ts(:), gyro(:,3))
  - iNeck = get_joint_index('Neck') % head yaw
  - iHead = get_joint_index('Head') % head pitch
  - Head_angles = [pos(idx,iNeck), pos(idx,iHead)];

Project #4 SLAM-PF

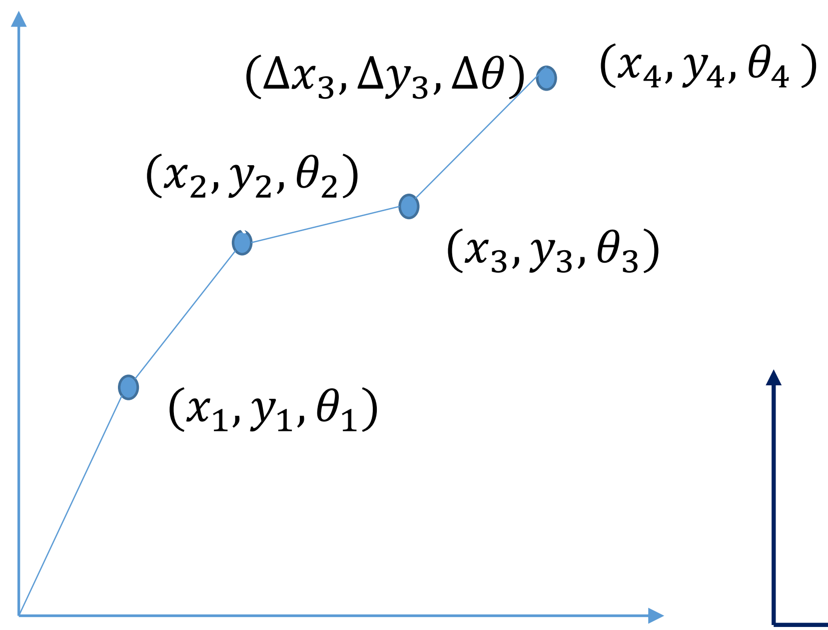
- lidar.mat
- t: 1.4268e+09(absolute time)
- ~~rsz: 4324~~ (You don’t need it)
- pose: [0 0 0] (global odometry)
- res: 0.0044 (radian, resolution)
- rpy: [-0.0120 -0.0164 -0.1107] (IMU roll pitch yaw)
- scan: [1x1081 single] (Scan data, range -135deg to 135 deg)

Project #4 SLAM-PF

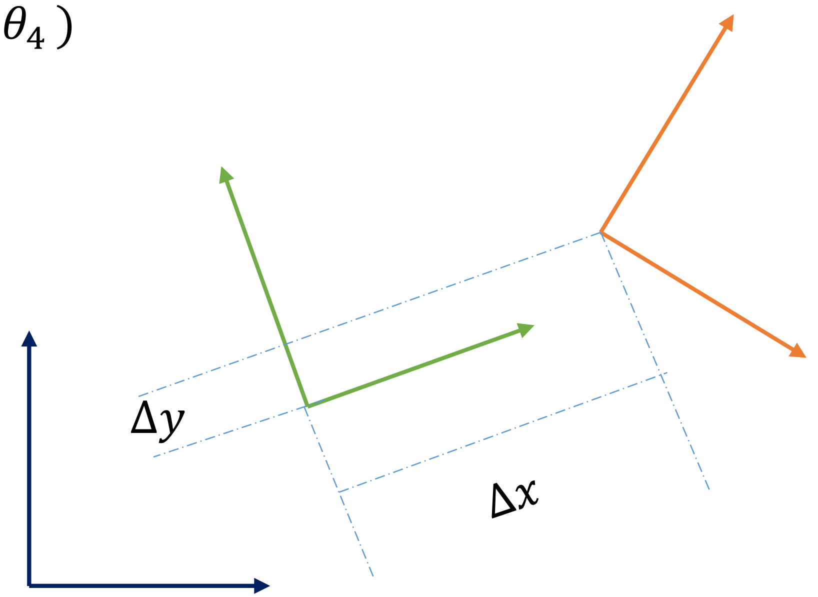
- Odometry
  - lidar{i}.pose: [x, y, theta]
  - +x: forward from robot
  - +y: left from robot
  - +z: up from robot
  - theta: rotation around +z

Implementation

- Relative pose based on the odometry
  - Given global odometry
  - Find delta x, delta y and delta theta

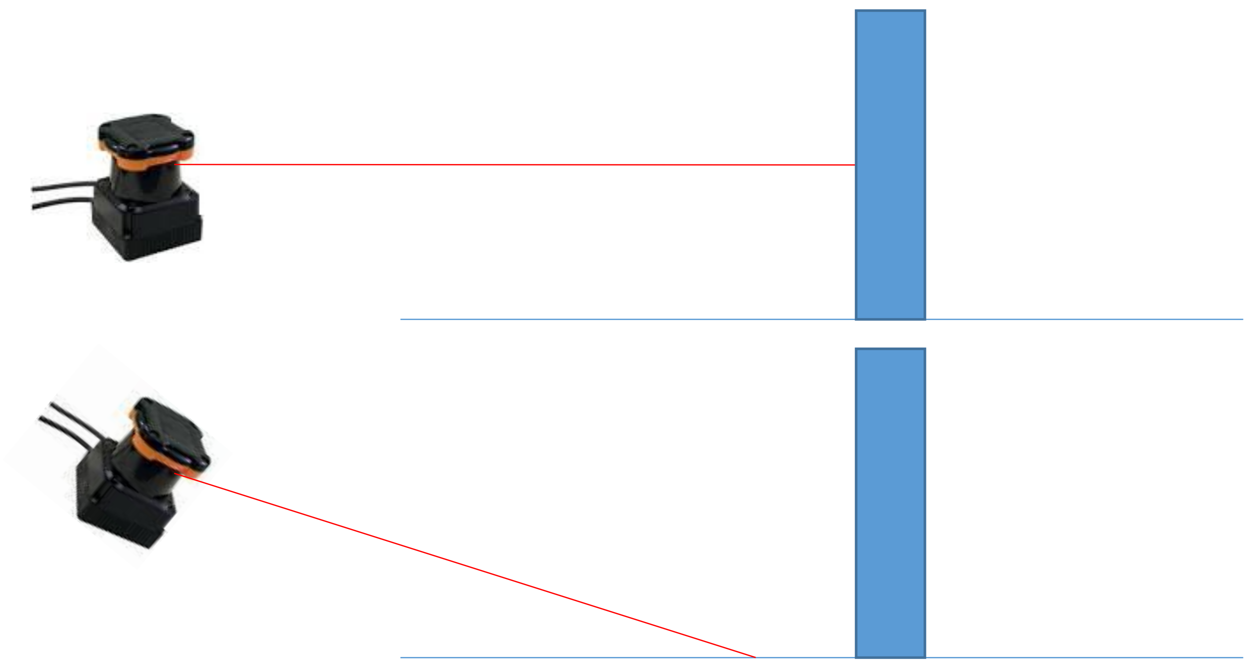


Transform to relative coordinate frame!!!!



Implementation

- Lidar Data
  - Remove the hits on the ground
  - Compensate the roll and yaw of head pose
  - Possible to compensate the motion of the robot



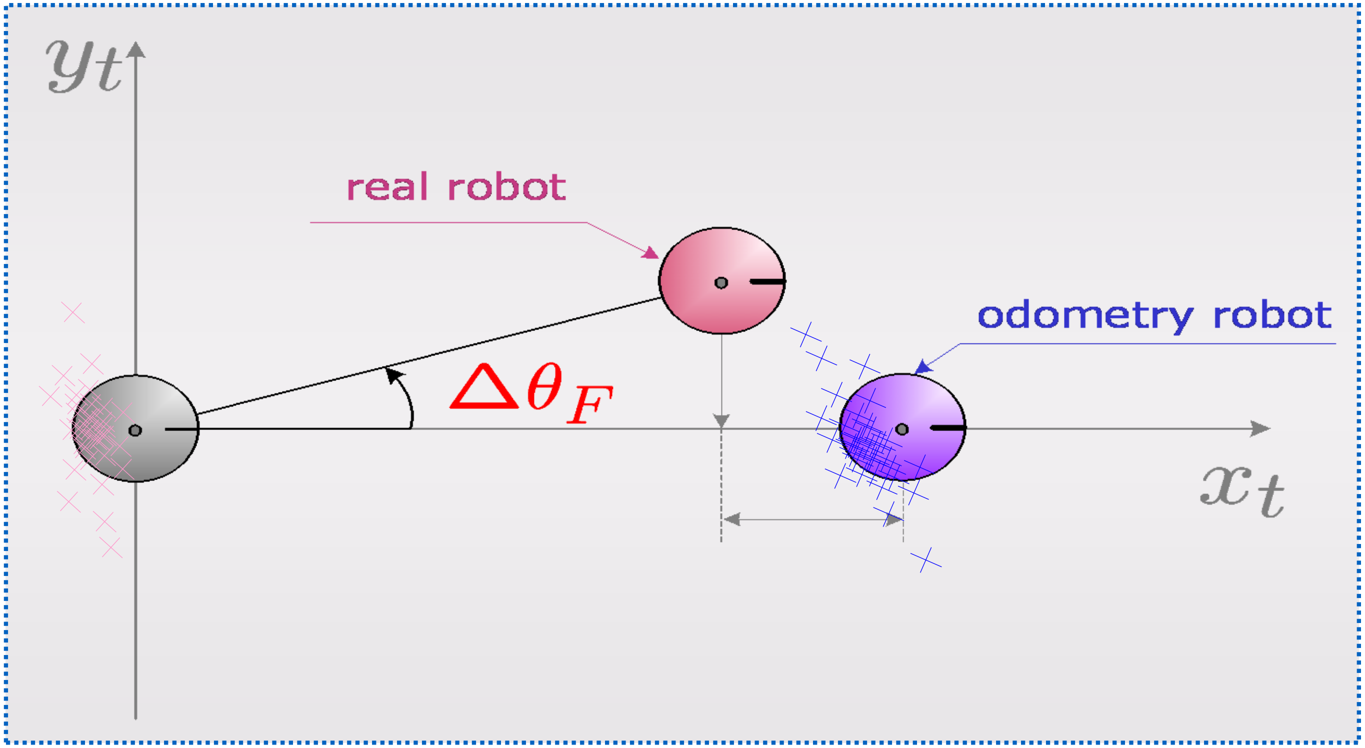
Implementation

- Prediction
  - particle = particle + delta (Odometry)
  - Particle numbers
  - Random noises (mu = 0, sigma = σ)

- Update
  - find particle that best matches
  - map_correlation.cpp
  - Particle weights

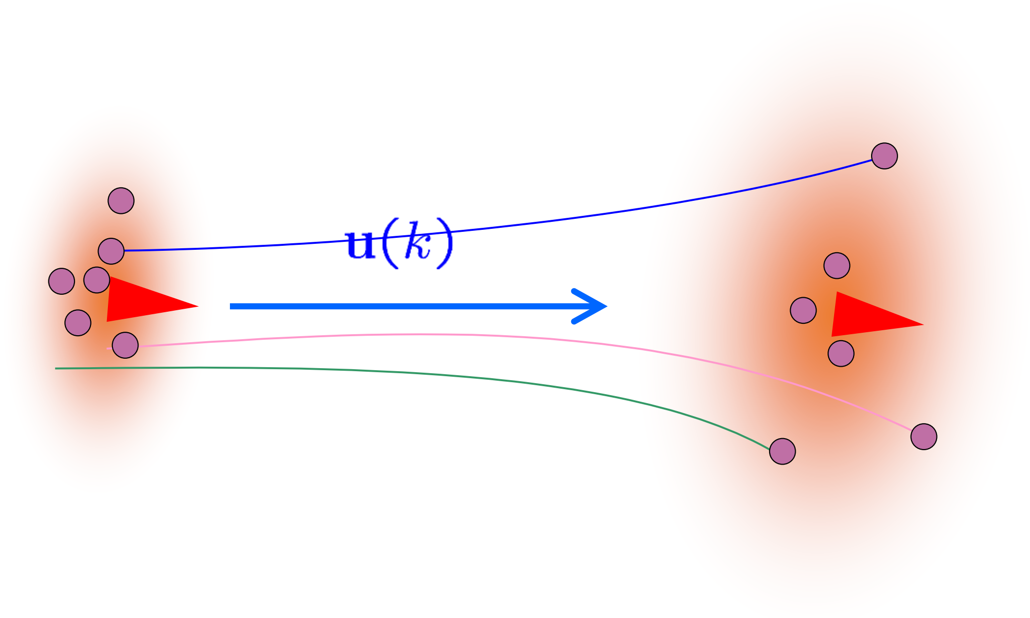
Implementation

- Motion Model
  - Error accumulates as robot moves



Implementation

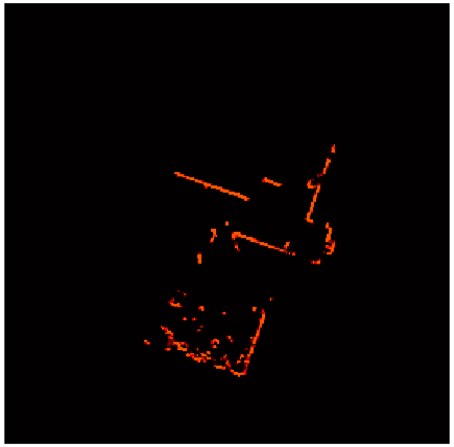
- Motion noise
  - Gaussian Random Noise (mu = 0, sigma = σ)



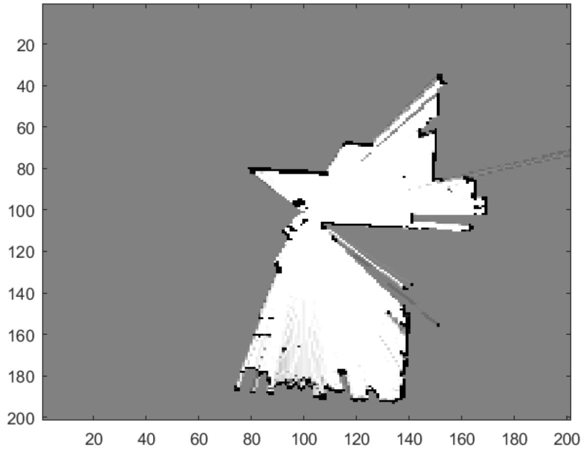
Mapping

- Global map/local map
  - Uniform grid map/Quad-tree map
  - Accurate is limited by grid size

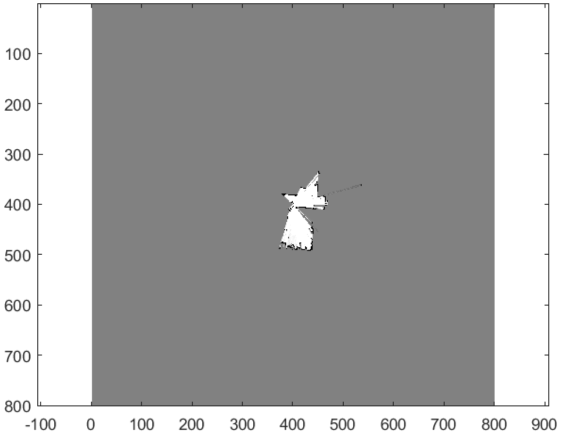
Result



Lidar



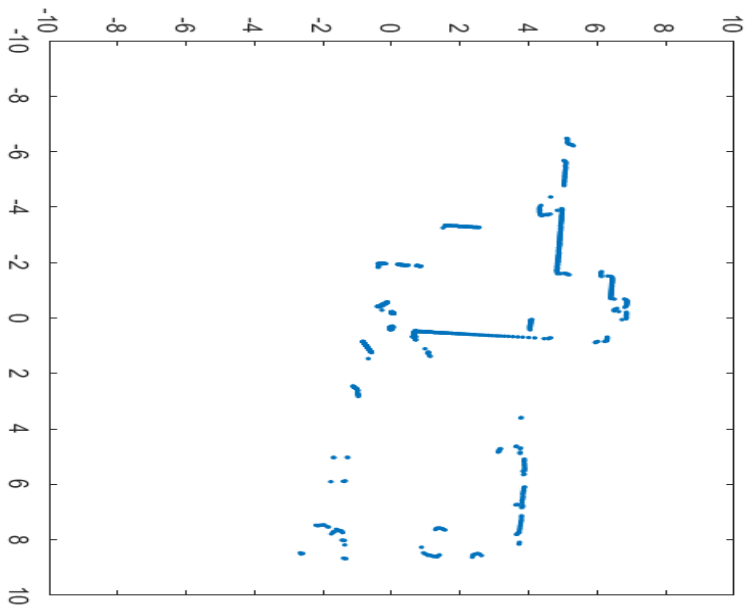
Local Map

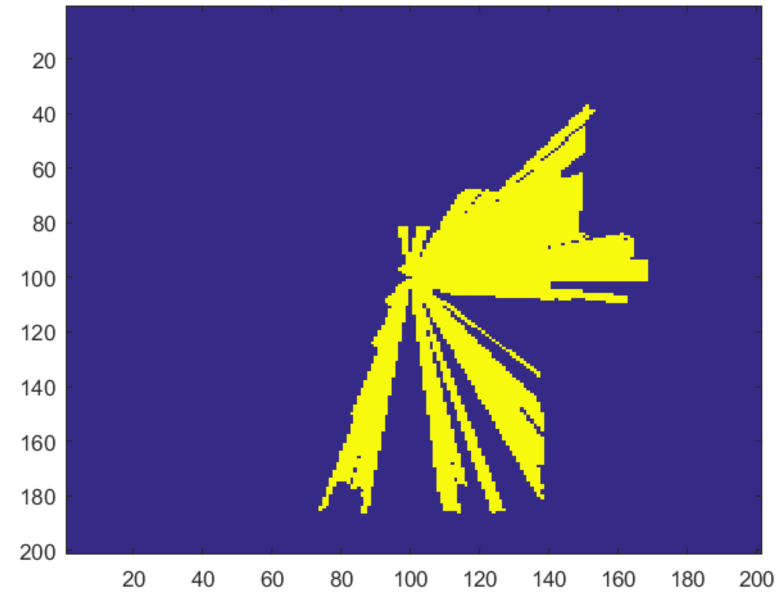


Global Map

Occupancy grid map

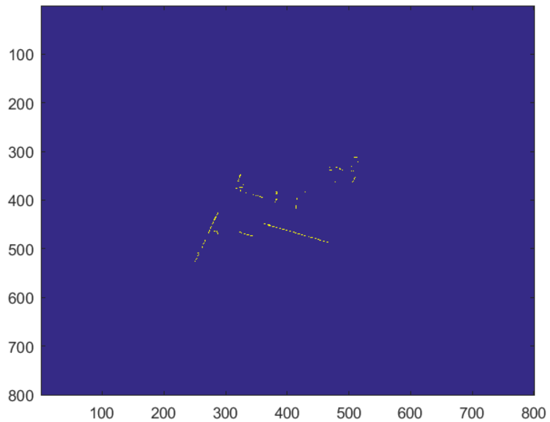
- getMapCellsFromRay.cpp
- [x_between, y_between] = getMapCellsFromRay(xori, yori, xis(i), yis(i));
- Get empty cells from this function

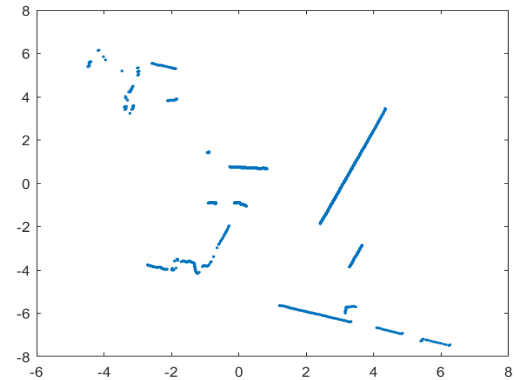


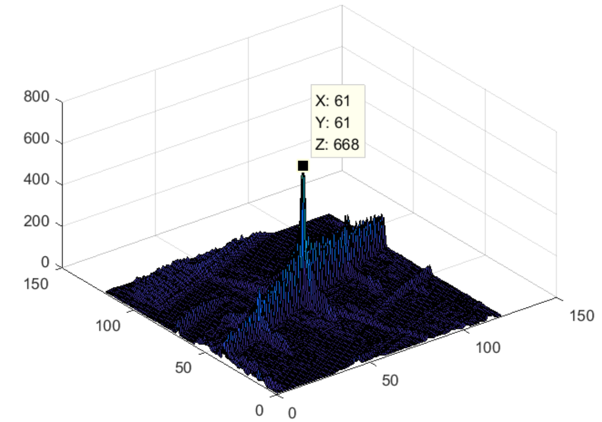


Map correlation

- map_correlation.cpp
- c = map_correlation(MAP.map,x_im,y_im,Y(1:3,:),x_range,y_range);
- MAP.map : Global Map
- x_im,y_im : physical x,y positions of the grid map cells
- im,Y(1:3,:) : occupied x,y positions from range sensor
- x_range,y_range : physical x,y,positions you want to evaluate "correlation"

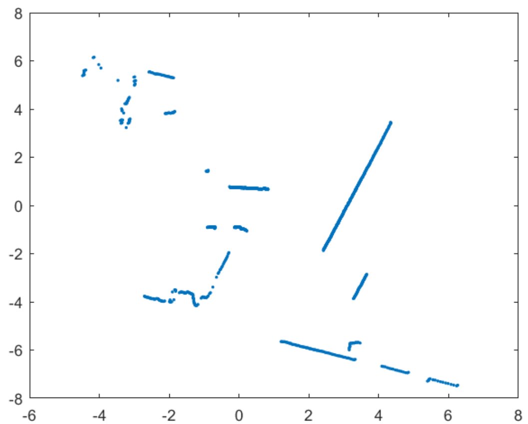




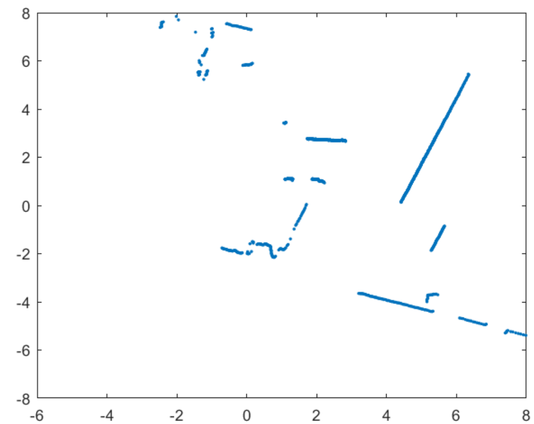


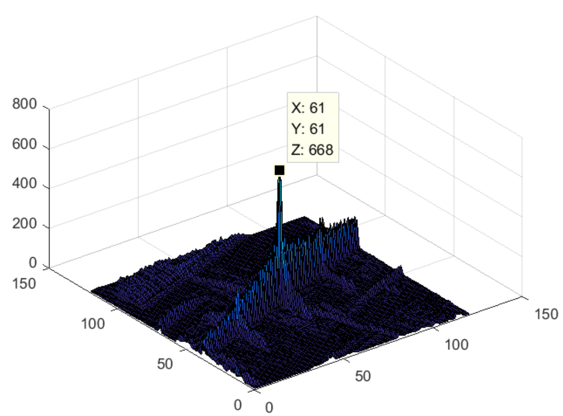
Map correlation

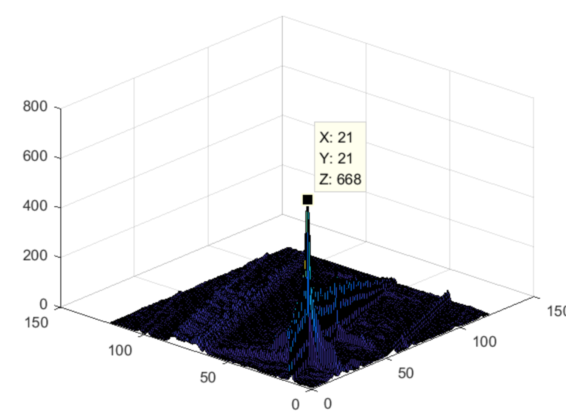
- map_correlation.cpp



X: +2m
Y: +2m

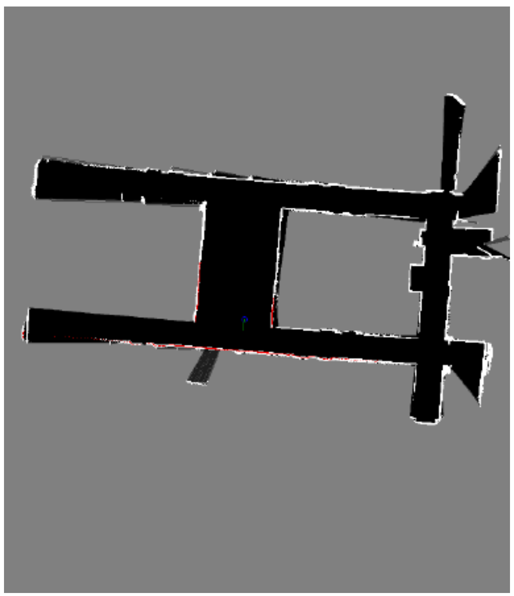


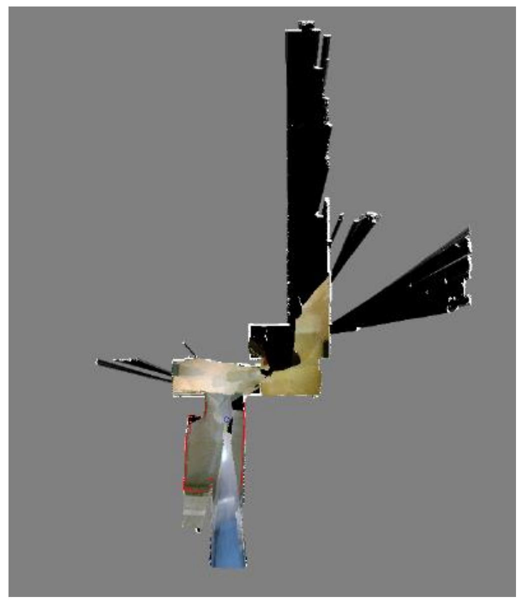


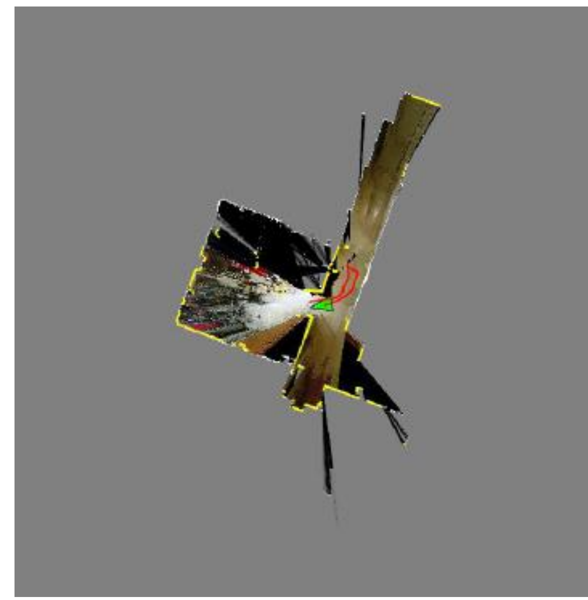


Project #4 SLAM-PF

- Previous good examples

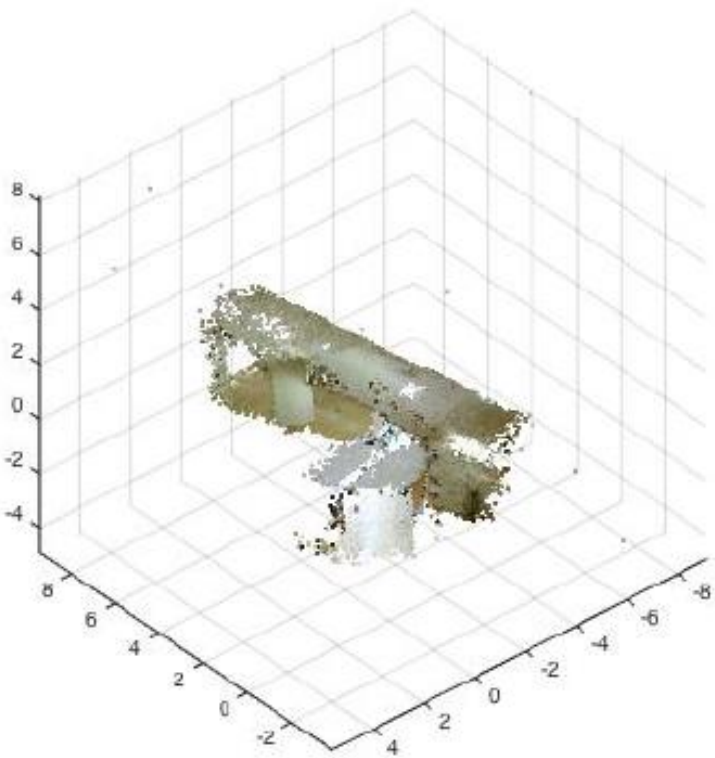


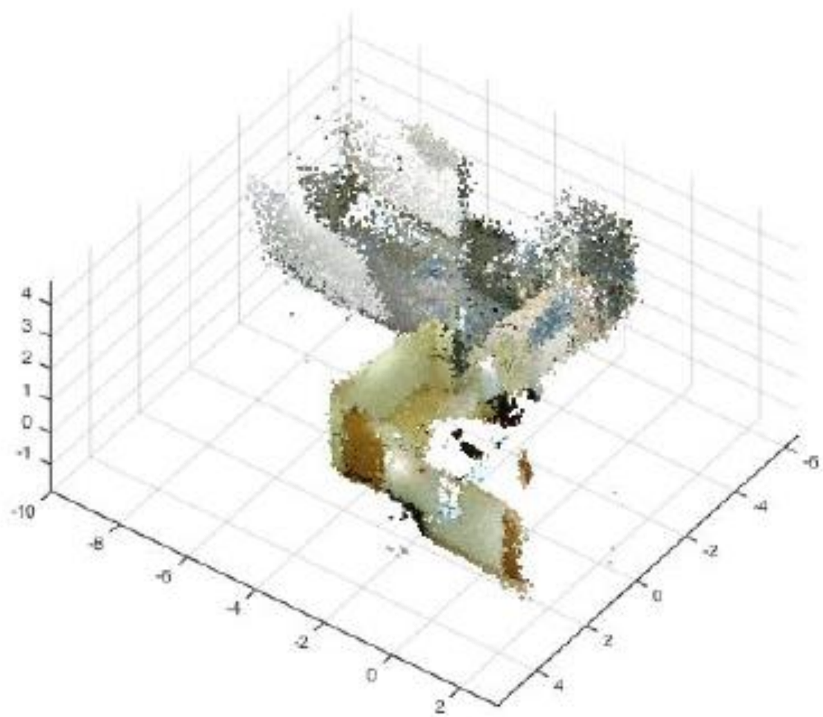




Project #4 SLAM-PF

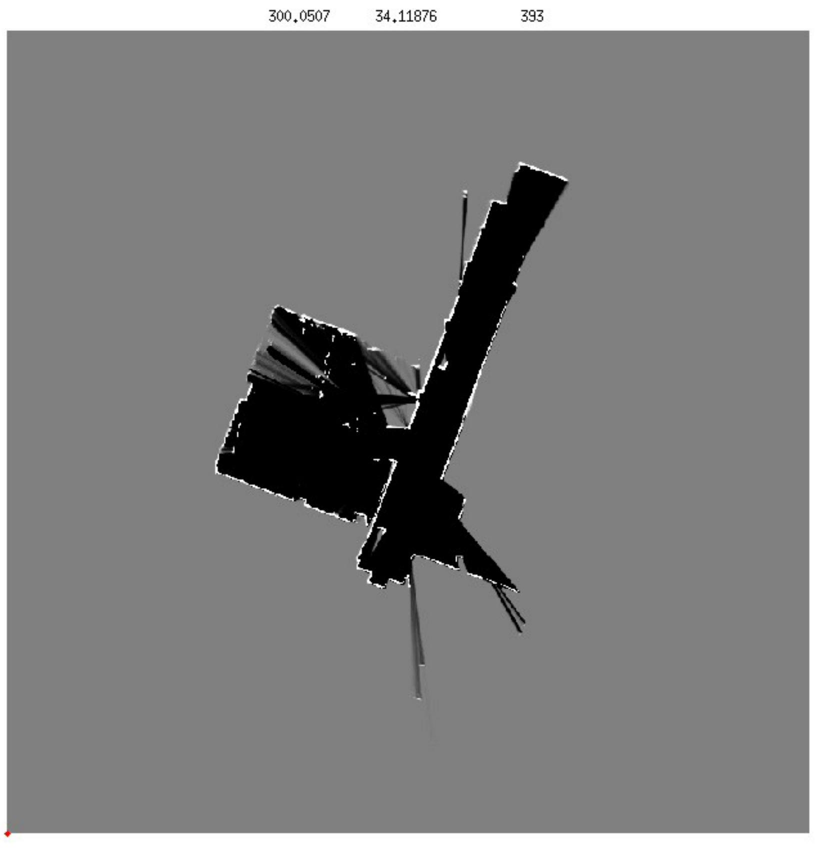
- Previous good examples





Project #4 SLAM-PF

- Previous good examples



Project #4 SLAM-PF

- Rubrics

Criteria	Pts
Performance on training data	4 pts
Performance on test data	7 pts
Textured map	4 pts
Report	5 pts
Total Points: 20	