DTU

Rasmus Andersen

34761 – Robot Autonomy

Simultaneous Localization and Mapping

Overview of 34761 – Robot Autonomy

- 3 lectures on software frameworks
- 7 lectures on building your own autonomy stack for a mobile robot
- 1 lecture on DL/RL – an overview of black-box approaches to what you have done
- 2 lectures of project work before hand in + guest lectures

Software frameworks			Robot autonomy stack									Project work + guest lectures	
30th Jan	6th Feb	13th Feb	20th Feb	27th Feb	6th Mar	13th Mar	20th Mar	27th Mar	3rd Apr	10th Apr	17th Apr	24th Apr	1st May
Software architecture	Intro to ROS2	More on ROS2	Localization	Mapping	Localization wrt. maps	SLAM	Navigation	Easter	Exploration	Behavior Trees	Black Box Approaches	Group work	

Today

Overview of 34761 – Robot Autonomy

- 3 lectures on software frameworks
- 7 lectures on building your own autonomy stack for a mobile robot
- 1 lecture on DL/RL – an overview of black-box approaches to what you have done
- 2 lectures of project work before hand in + guest lectures

Software frameworks			Robot autonomy stack									Project work + guest lectures	
30th Jan	6th Feb	13th Feb	20th Feb	27th Feb	6th Mar	13th Mar	20th Mar	27th Mar	3rd Apr	10th Apr	17th Apr	24th Apr	1st May
Software architecture	Intro to ROS2	More on ROS2	Localization	Mapping	Localization wrt. maps	SLAM	Navigation	Easter	Exploration	Behavior Trees	Black Box Approaches	Group work	

Next week

Nicholas Hansen

Recalling from last lecture

- We have a map; we want to know where in the map we are
  – Estimates the location and orientation of the robot in the environment as it moves

- How do we get the initial position?
  – Bayes filtering
  – Particle filter / monte carlo localization

$$Bel(x_t) = \eta P(z_t|x_t) \int P(x_t|x_{t-1}) \cdot Bel(x_{t-1}) dx_{t-1}$$

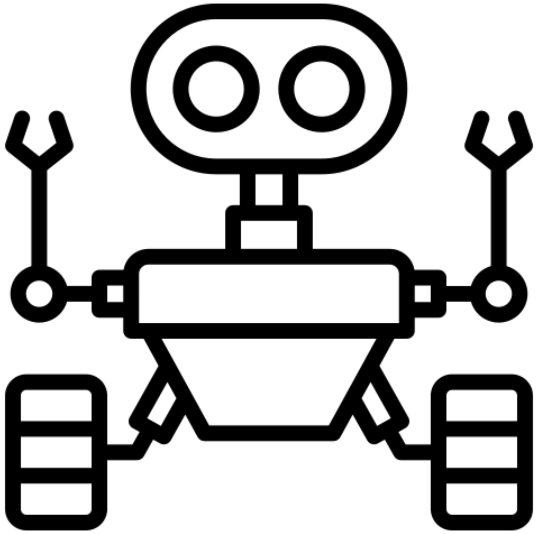
What is SLAM

- Estimate the pose of a robot and the map of the environment at the same time

- SLAM is hard, because
  – a map is needed for localization and
  – a good pose estimate is needed for mapping

- So if
  – **Localization** is inferring location given a map and
  – **Mapping** is inferring a map given locations
  – **SLAM** is learning a map and locating the robot simultaneously

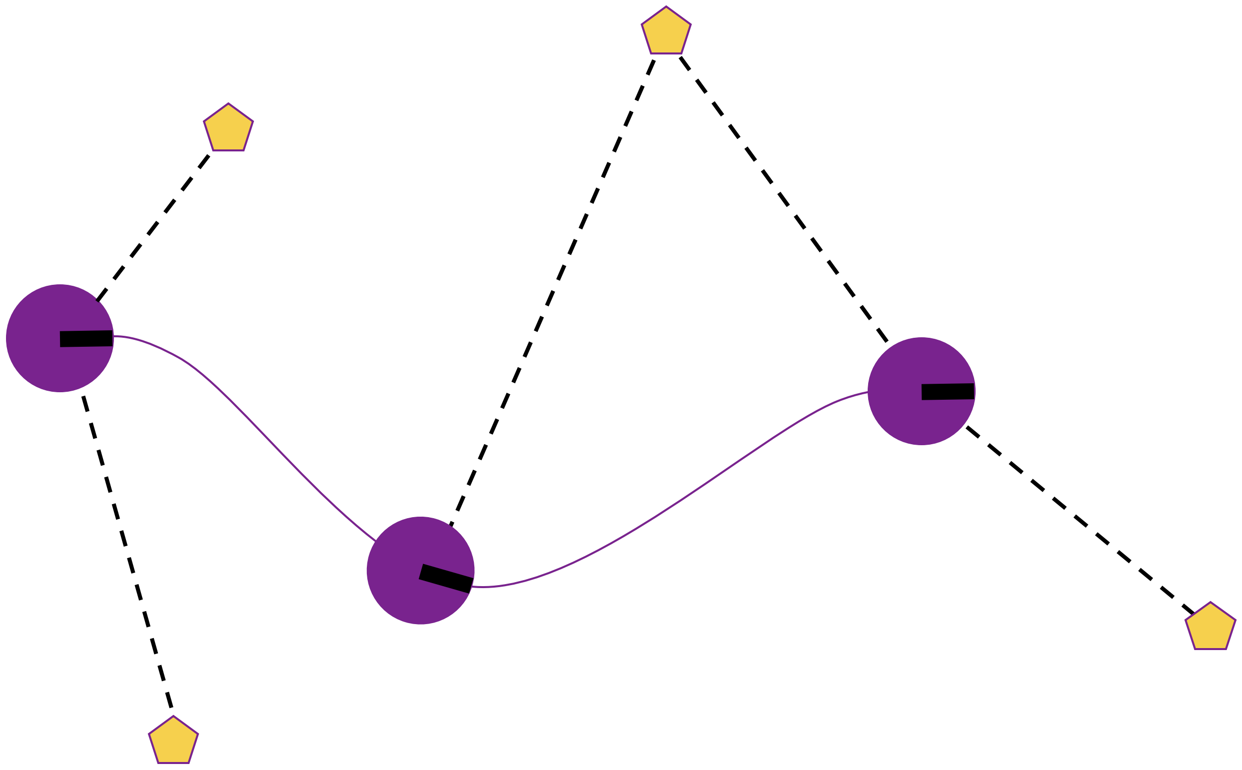
Passive vs Active SLAM



30th Jan	6th Feb	13th Feb	20th Feb	27th Feb	6th Mar	13th Mar	20th Mar	27th Mar	3rd Apr	10th Apr	17th Apr	24th Apr	1st May
Software architecture	Intro to ROS2	More on ROS2	Localization	Mapping	Localization wrt. maps	SLAM	Navigation	Easter	Exploration	Behavior Trees	Black Box Approaches	Group work	

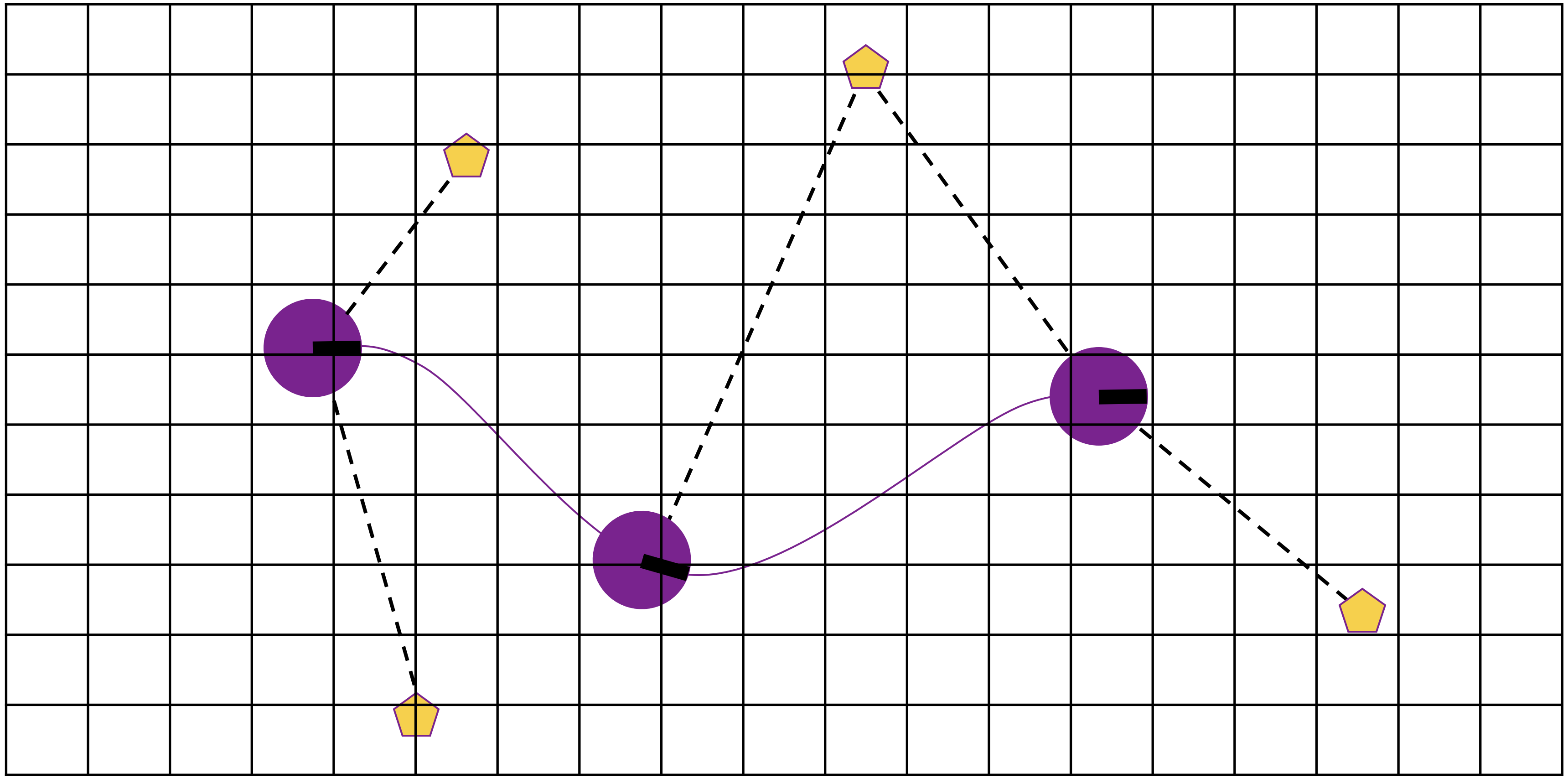
Localization in a map

- Estimating the robot pose given landmarks in a map



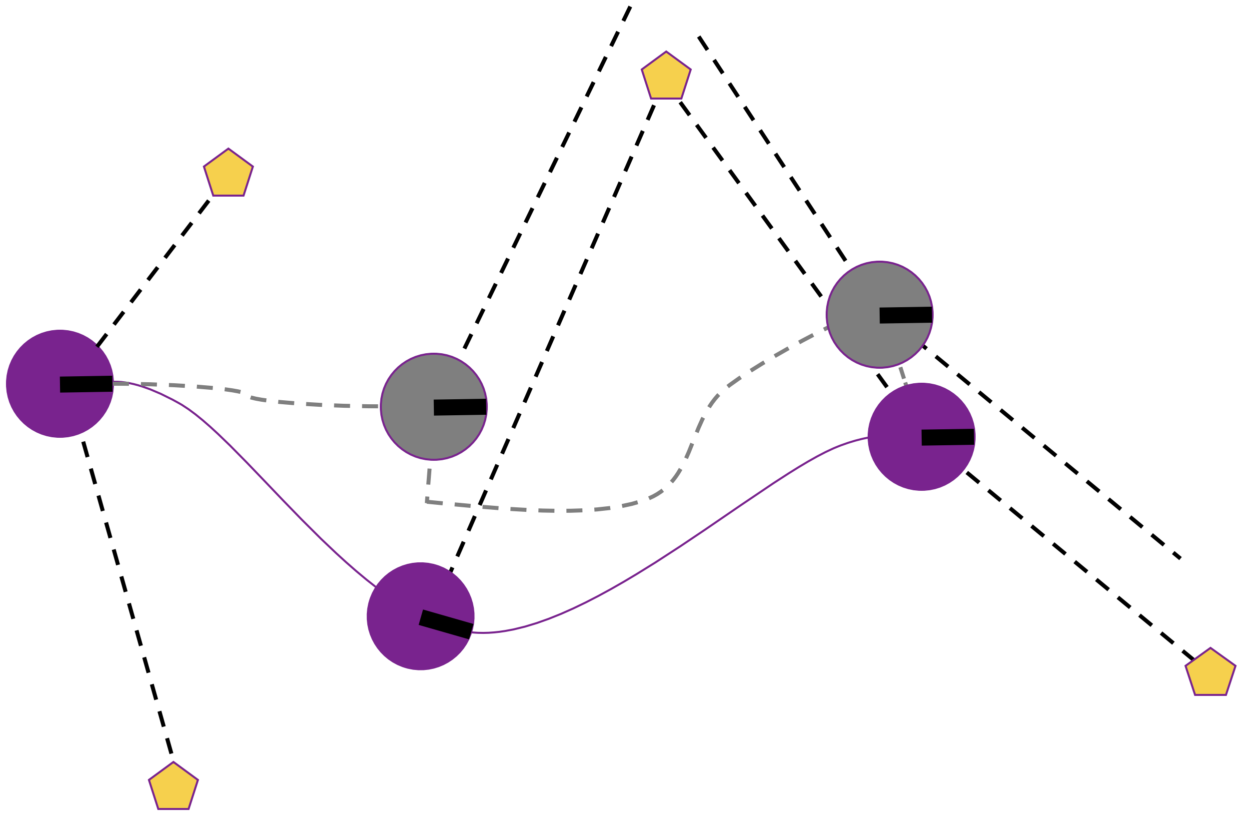
Localization in a map

- Estimating the robot pose given landmarks in a map

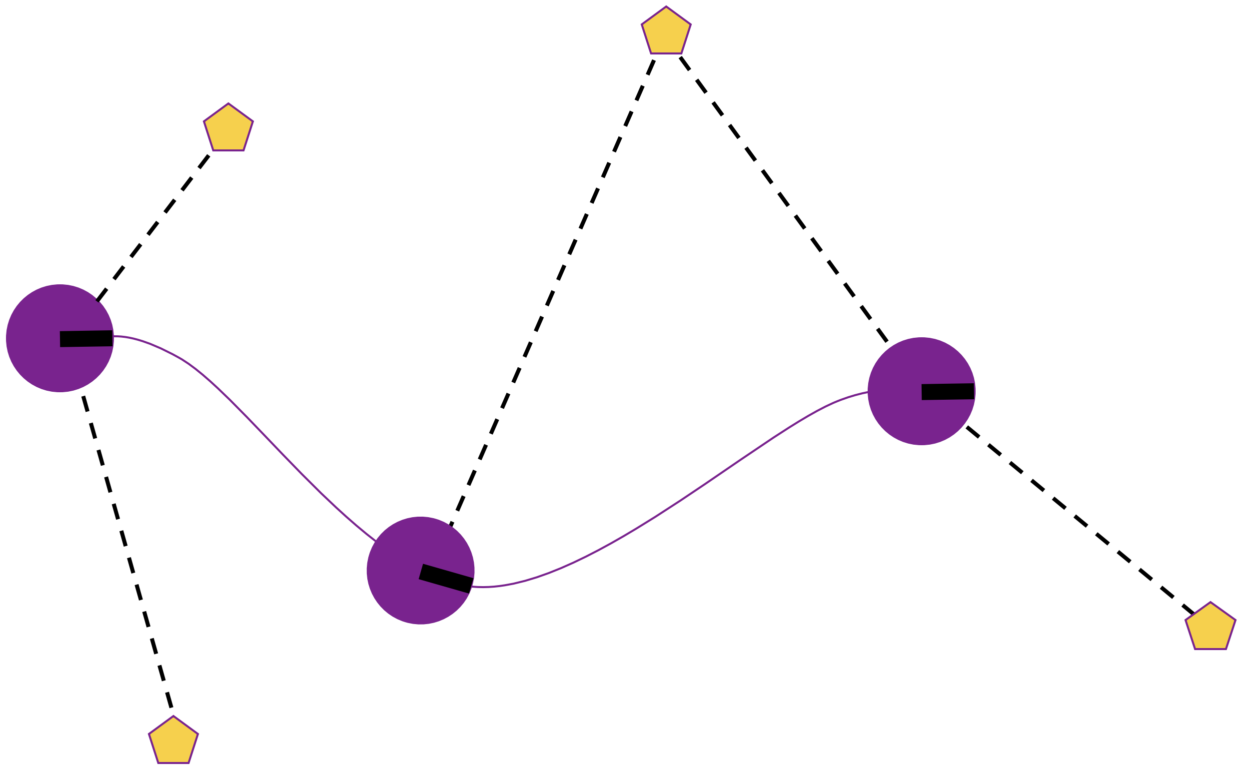


Localization in a map

- Estimating the robot pose given landmarks in a map

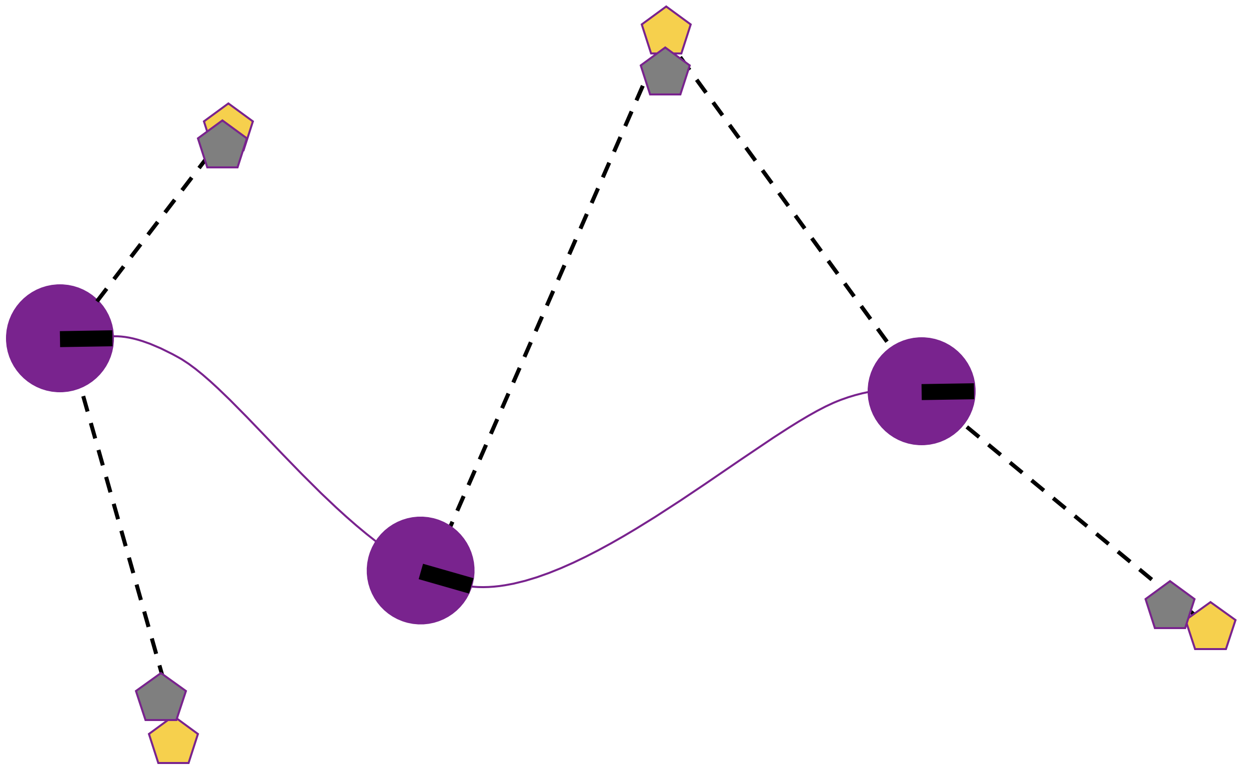


Mapping of the environment



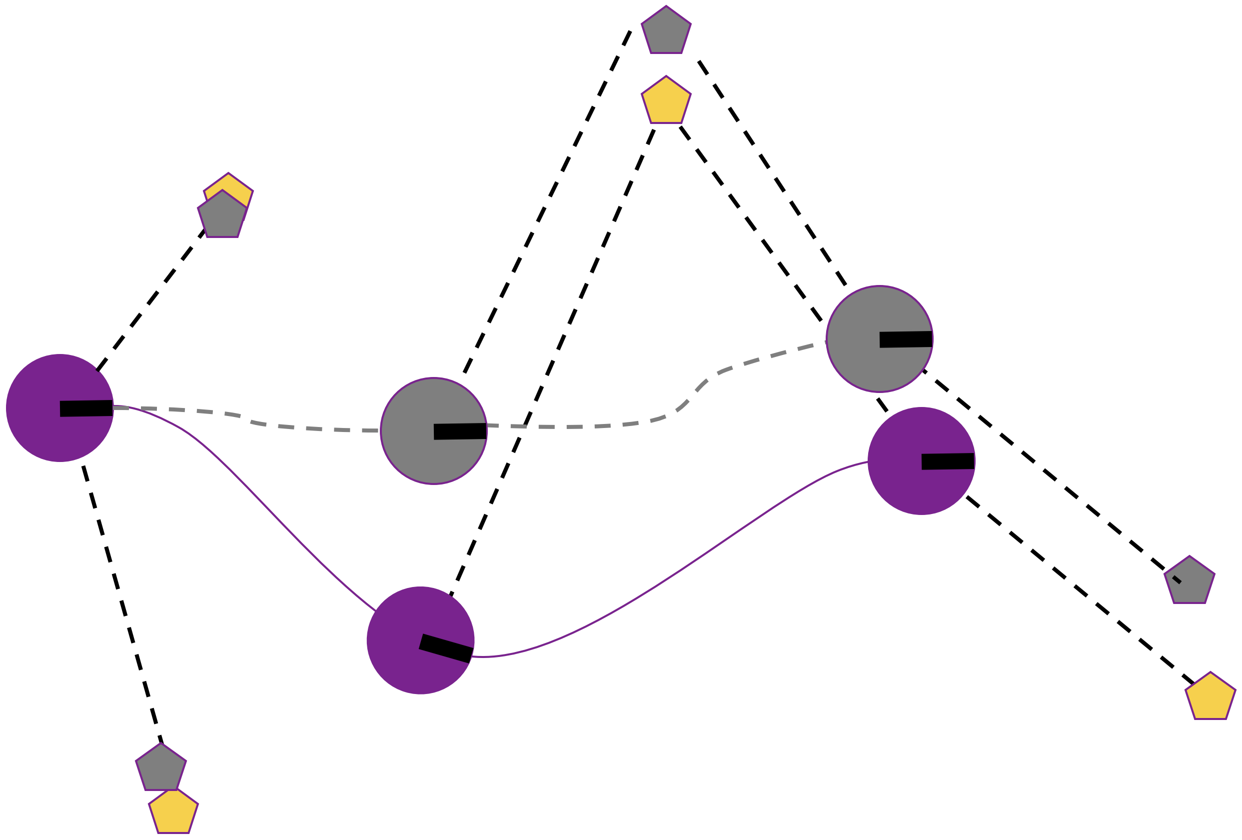
Mapping of the environment

- Estimating the landmarks given the robot’s pose



Simultaneous localization and mapping

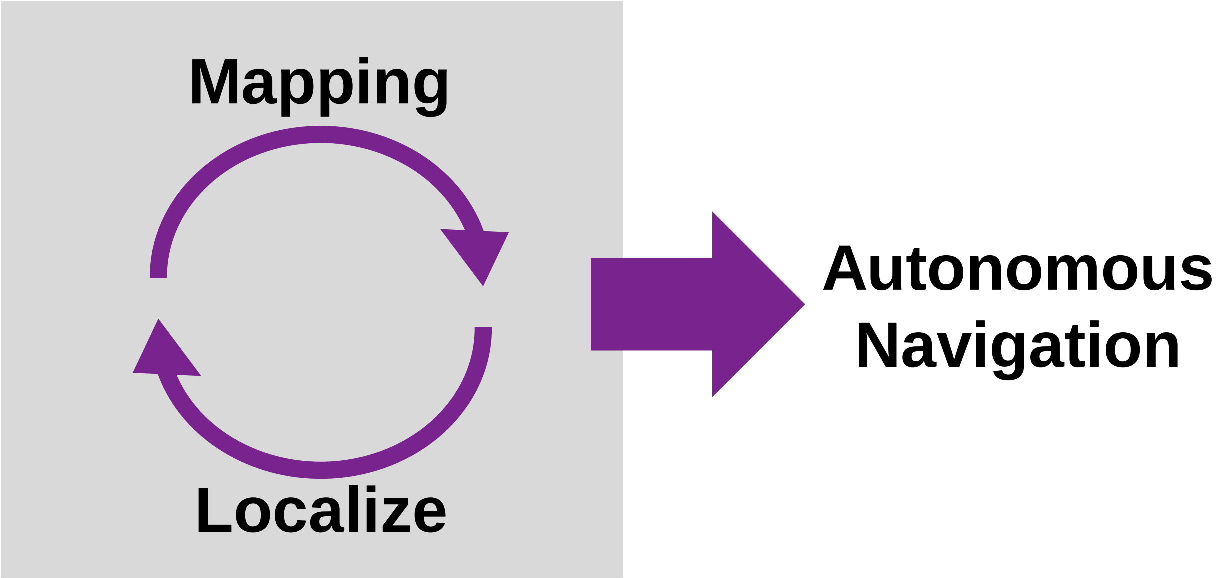
- Estimate the robot's pose and the landmarks at the same time



- We can generate a more accurate map if we know the robot's location
  – Through GPS, motion capture or other external tools

- We can know the robot's location more accurately if we know the map
  – E.g. through previous SLAM or other map generation tools

- In SLAM we have neither!

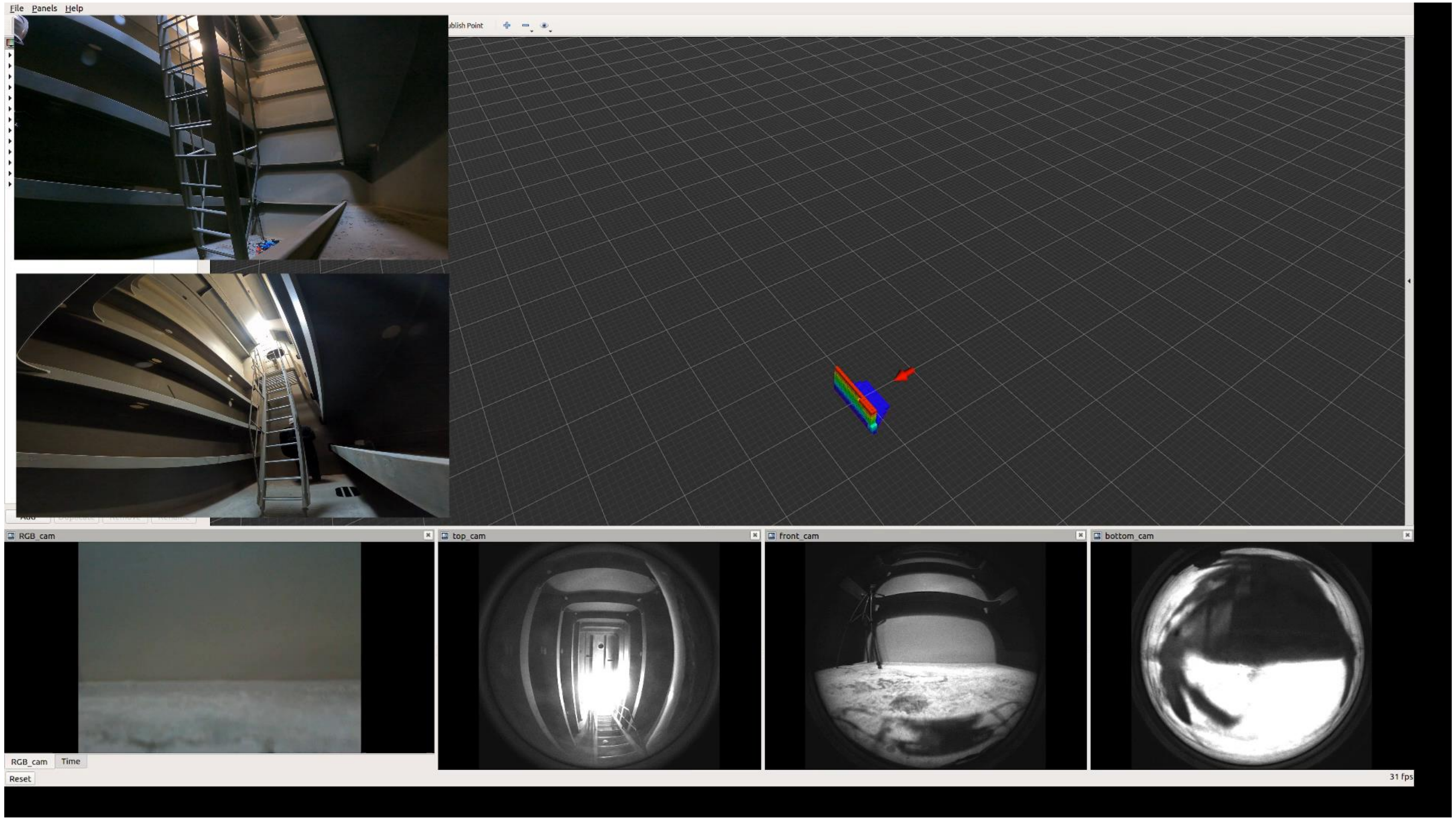


SLAM applications

- SLAM is central to a range of indoor, outdoor, in-air and underwater applications for both manned and autonomous vehicles.
- At home: vacuum cleaner, lawn mower
- Air: surveillance with unmanned air vehicles
- Underwater: reef monitoring
- Underground: exploration of mines
- Space: terrain mapping for localization

Example

Example


File Panels Help
RGB_cam
top_cam
front_cam
bottom_cam
RGB_cam Time
Reset
31 fps

Example


CERBERUS
Winning Run in the
DARPA Subterranean Challenge Final Event

The SLAM problem

- What we have:
  - The robot's control: $u_{1:T} = \{u_1, u_2, \dots, u_T\}$
  - Sensor observations: $z_{1:T} = \{z_1, z_2, \dots, z_T\}$

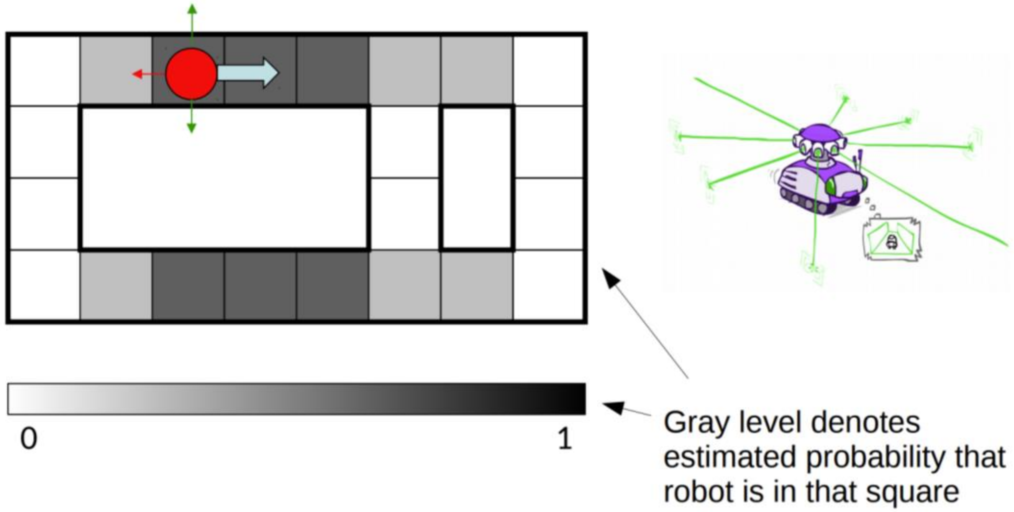
- What we want:
  - A map of the environment: m
  - Path of the robot: $x_{0:T} = \{x_0, z_1, \dots, x_T\}$

The SLAM problem

- SLAM is considered a fundamental problem to overcome for robots to become truly autonomous

- Large variety of different SLAM approaches have been developed
  – Massive field of research (you have read some of the papers for today)

- The majority uses probabilistic concepts, similar to those we introduced last week
  – It's very hard to say with 100% certainty where the robot exactly is
  – That's why we used a particle filter to get hypotheses of where it *could* be



In probabilistic form

- What we have:
  - The robot's control: $u_{1:T} = \{u_1, u_2, \dots, u_T\}$
  - Sensor observations: $z_{1:T} = \{z_1, z_2, \dots, z_T\}$

- What we want:
  - A map of the environment: m
  - Path of the robot: $x_{0:T} = \{x_0, z_1, \dots, x_T\}$

- What we are trying to estimate:

$$P(x_{0:T}, m | z_{1:T}, u_{1:T})$$

In probabilistic form

- The full SLAM problem (offline)
  - Estimate the entire path + map of the robot

$$P(x_{0:T}, m | z_{1:T}, u_{1:T})$$

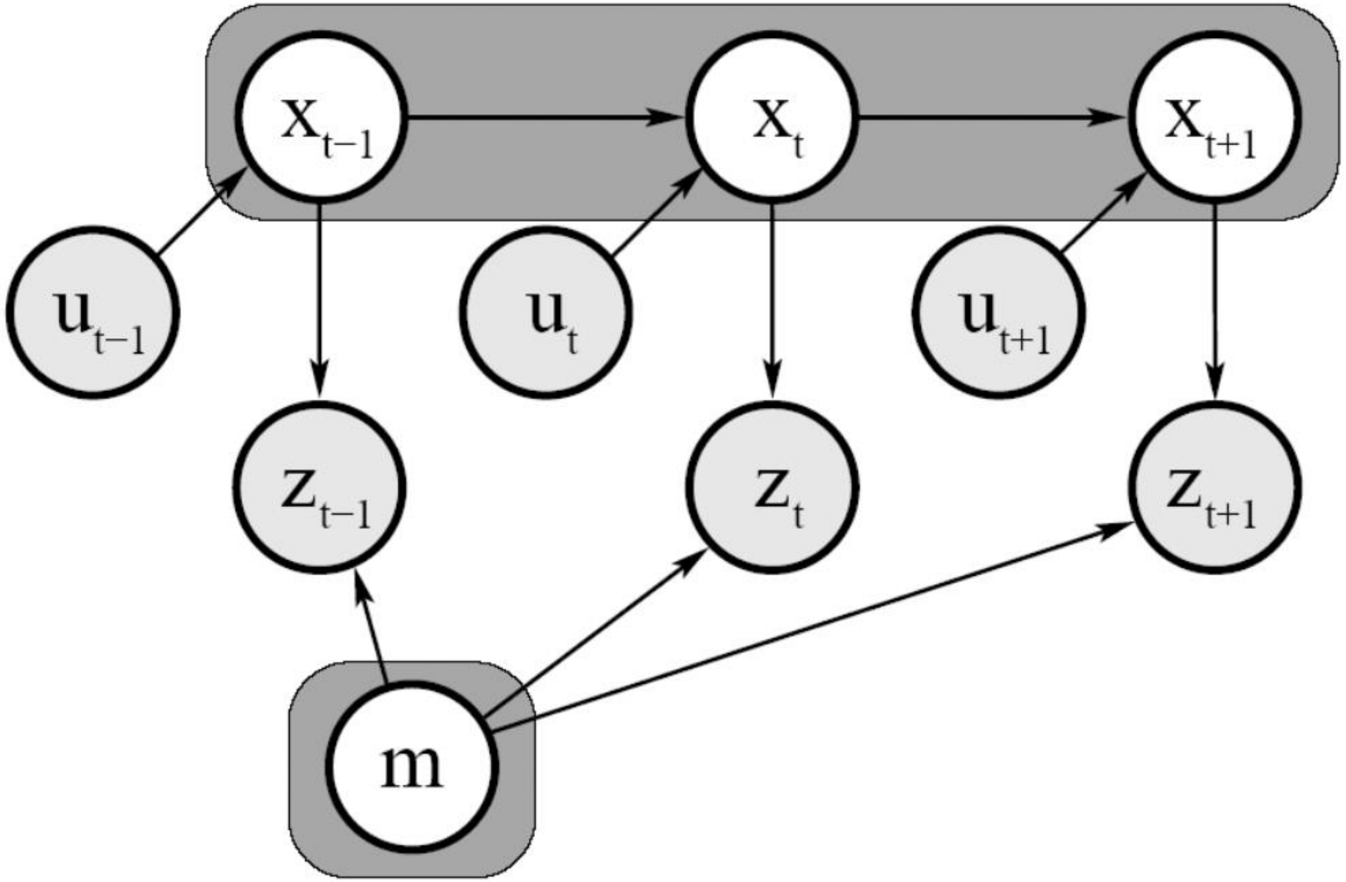
  - Relevant for joystick operated robots that generate their map after finishing their movement

- Online SLAM
  - The same goal, but now we have to update iteratively
  - Estimate the most recent pose and map

$$P(x_t, m | z_{1:t}, u_{1:t})$$

  - For most real-world autonomous systems, this is the most relevant approach

Graphical model of full SLAM



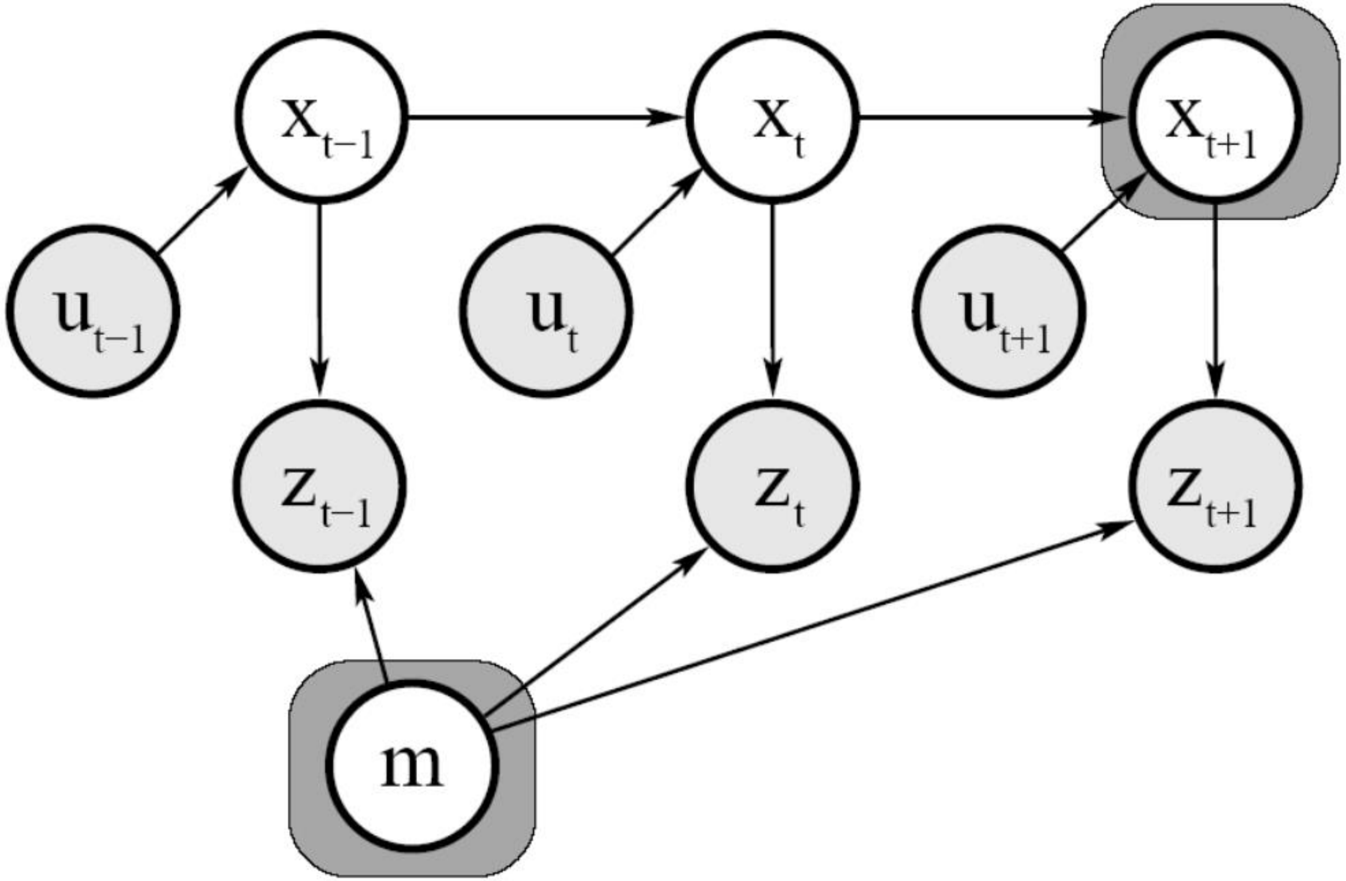
Unknown

Observed

Unknown

$$P(x_{0:T}, m | z_{1:T}, u_{1:T})$$

Graphical model of online SLAM



$$P(x_{t+1}, m | z_{1:t+1}, u_{1:t+1})$$

Online SLAM

- We are not interested in the previous poses of the robot (unlike in the full SLAM case)

$$P(x_t, m | z_{1:t}, u_{1:t})$$

- But the current position depends on the previous position, hence the arrow $x_{t-1} \to x_t$ in the graph

- The solution is to integrate the previous poses out

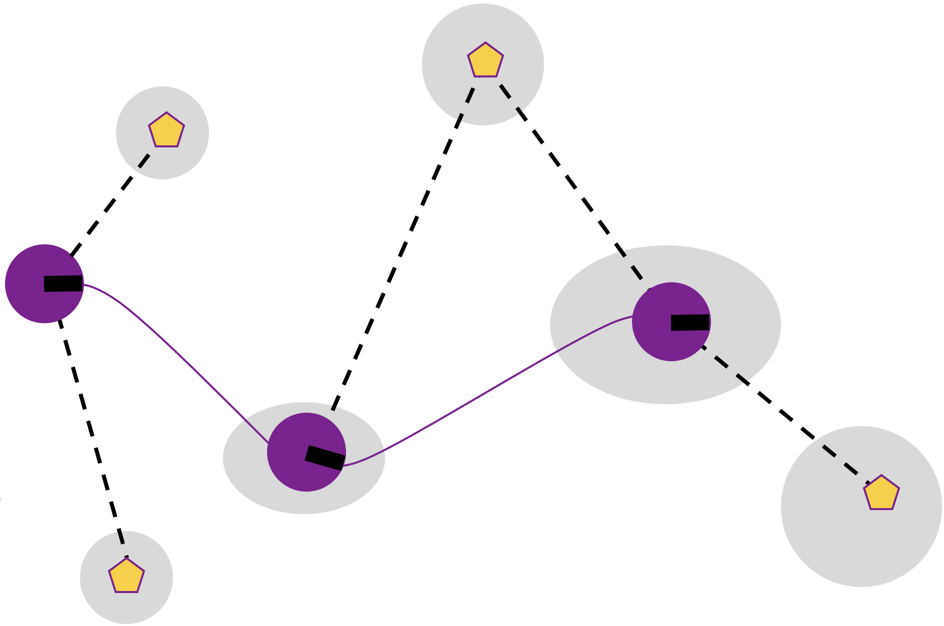
$$P(x_t, m | z_{1:t}, u_{1:t}) = \int_{x_0} \int_{x_1} \dots \int_{x_{t-1}} p(x_{0:t}, m | z_{1:t}, u_{1:t})\, dx_{t-1} \dots dx_1 dx_0$$

  – i.e. accumulate all possible locations over time

Online SLAM

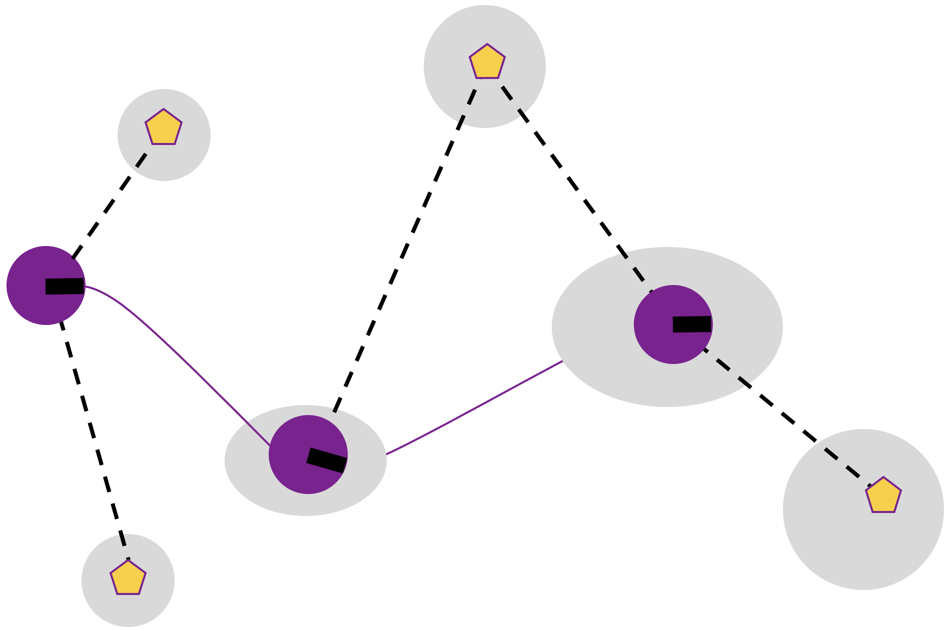
- When we move the robot, we introduce pose uncertainty
- We have inherent uncertainty in our observations
- These uncertainties needs to be combined
- This is the integration of the probability distribution

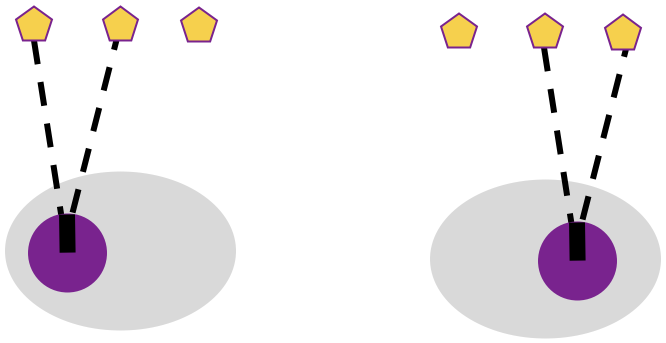
$$P(x_t, m|z_{1:t}, u_{1:t}) = \int_{x_0} \int_{x_1} \dots \int_{x_{t-1}} p(x_{0:t}, m|z_{1:t}, u_{1:t})\, dx_{t-1} \dots dx_1 dx_0$$

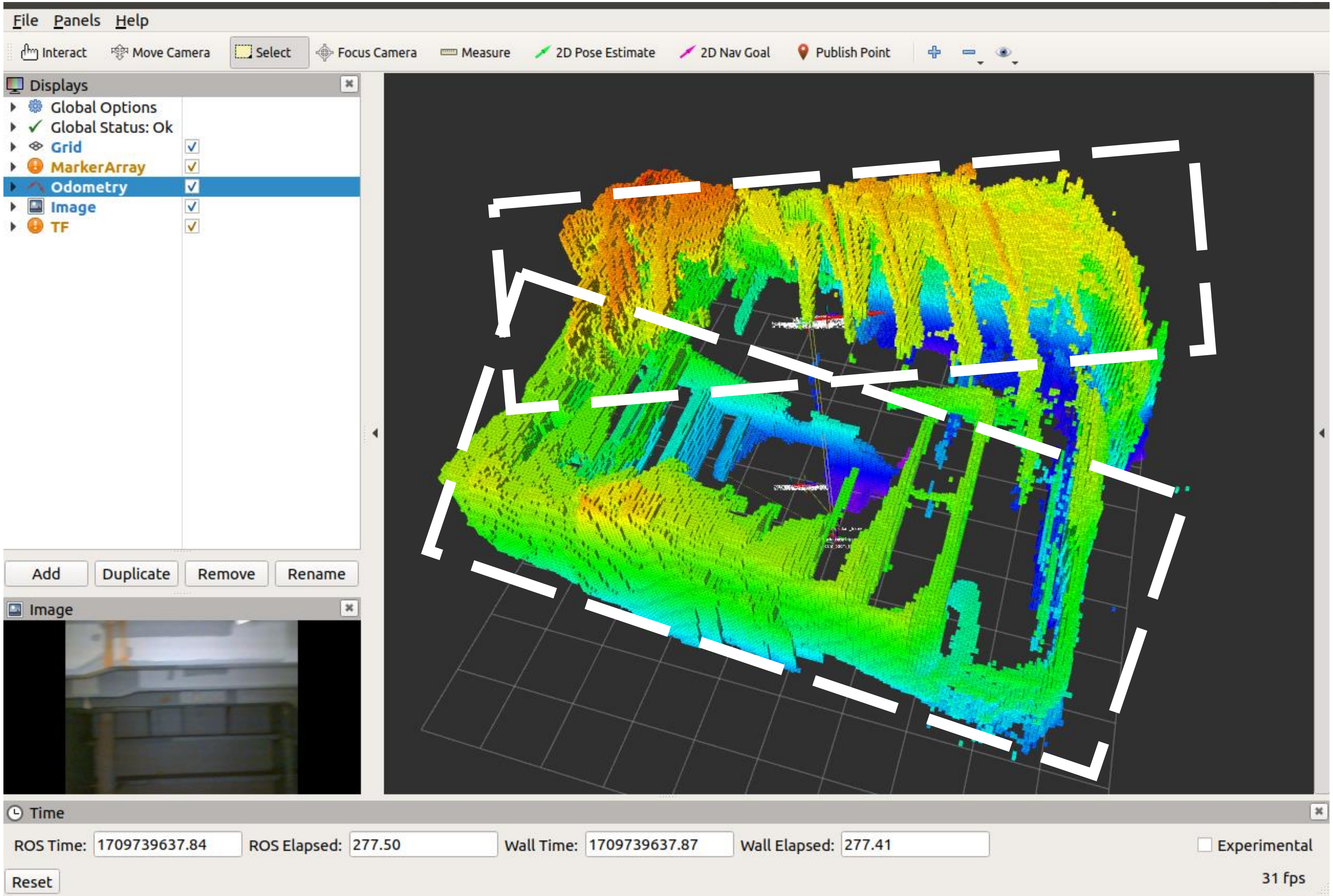


Data association

- Robot path and map are both unknown
- Errors in map and pose estimates are correlated
- The mapping between observations and landmarks is unknown (i.e. we only observe the environment partially through our sensors)
- Picking wrong data associations can have consequences (divergence)

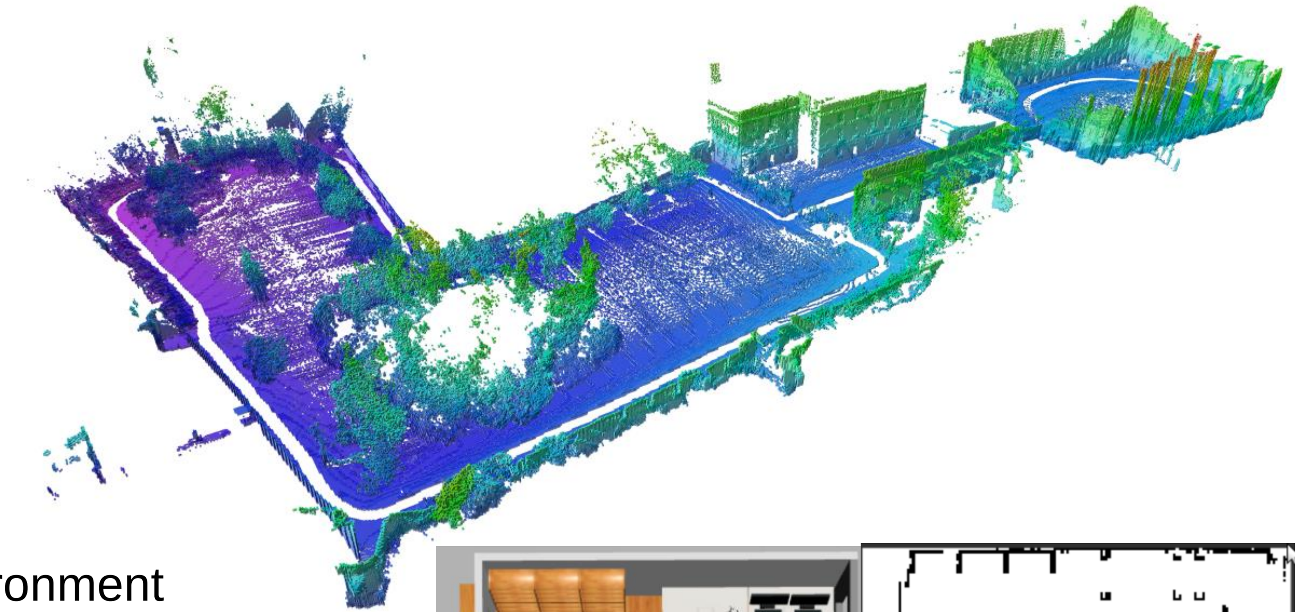


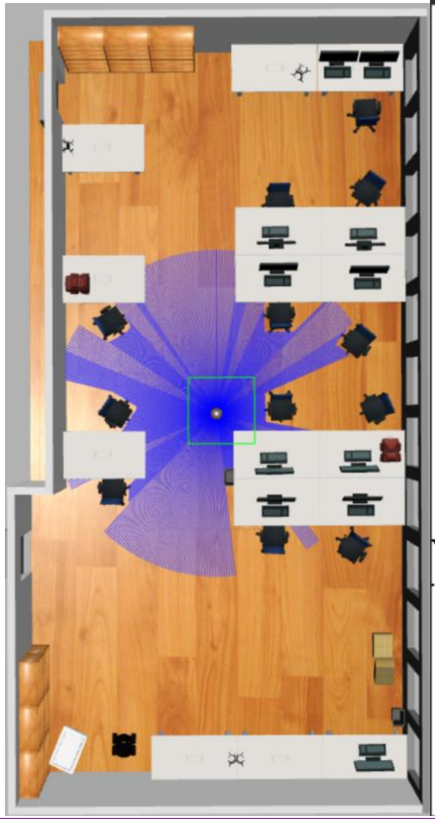


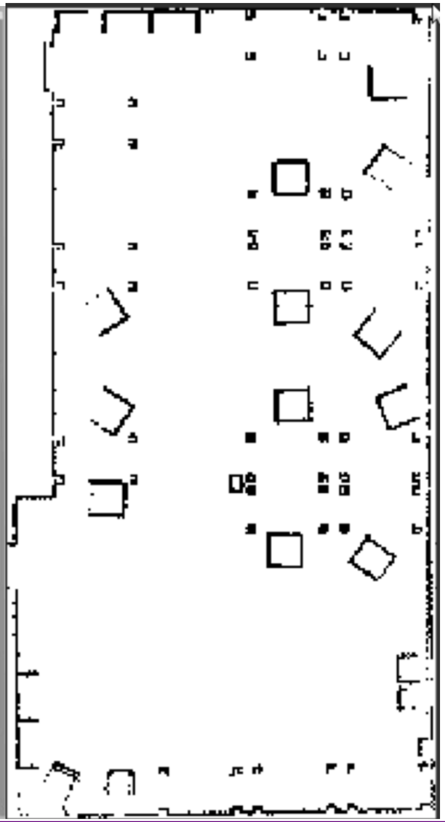

File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Global Status: Ok
Grid
MarkerArray
Odometry
Image
TF
Add
Duplicate
Remove
Rename
Image
Time
ROS Time: 1709739637.84
ROS Elapsed: 277.50
Wall Time: 1709739637.87
Wall Elapsed: 277.41
Experimental
Reset
31 fps

Map types

- Topological maps
- **Metric maps**
  - Full 2D/3D representation of the environment
  - Contains all information to do path planning, navigation, mapping, etc.
  - Map size is directly proportional to the environment (computational heavy)
  - Landmark-based maps
    - Like topological maps, but here the landmarks are unique and incorporate scale
  - Occupancy grid maps
    - A grid of cells that contains occupancy information
- Geometric maps







Three main paradigms of SLAM

KALMAN FILTER

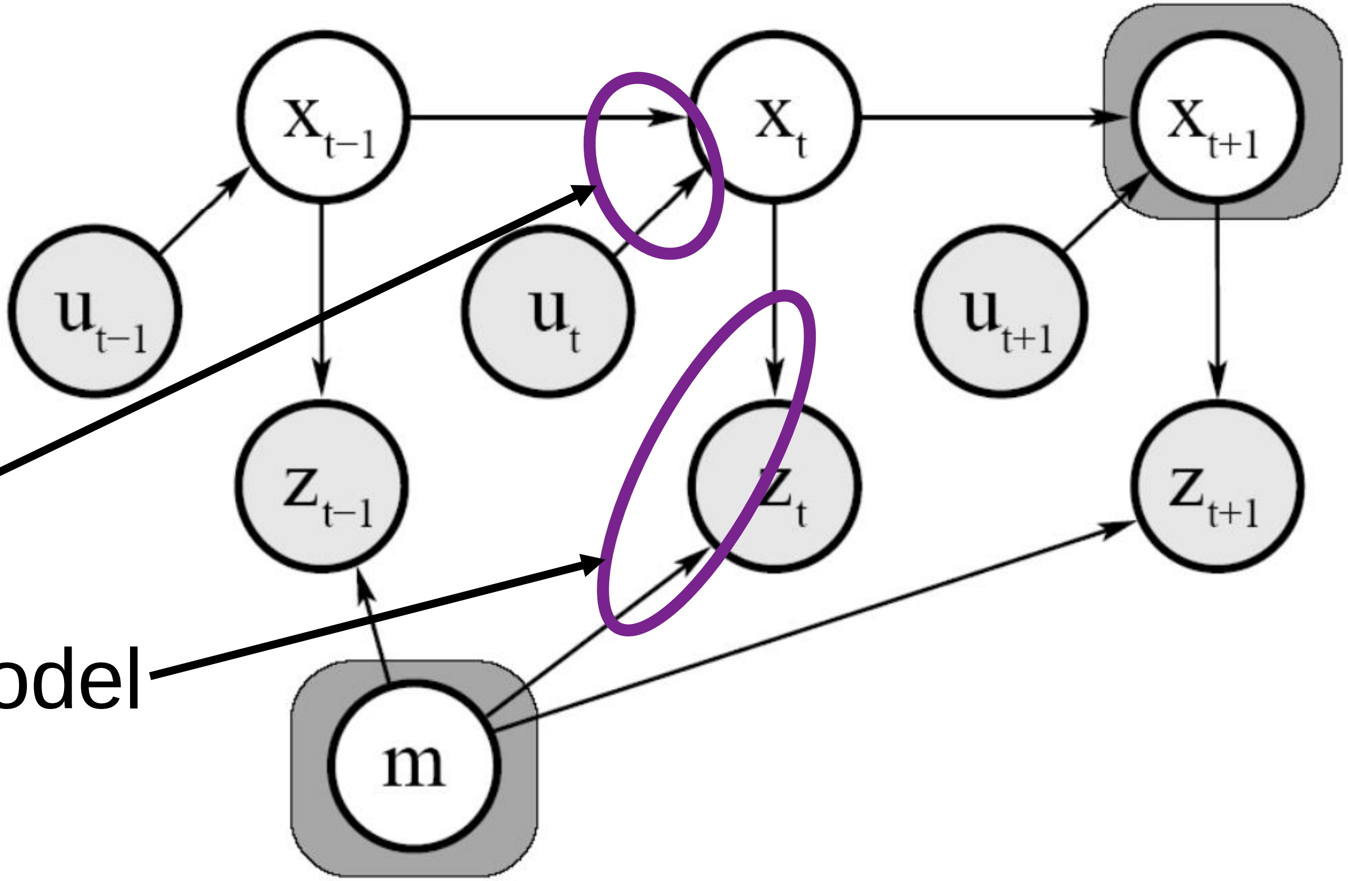
PARTICLE FILTER

GRAPH-BASED

The models common for all paradigms

Motion model

Observation model

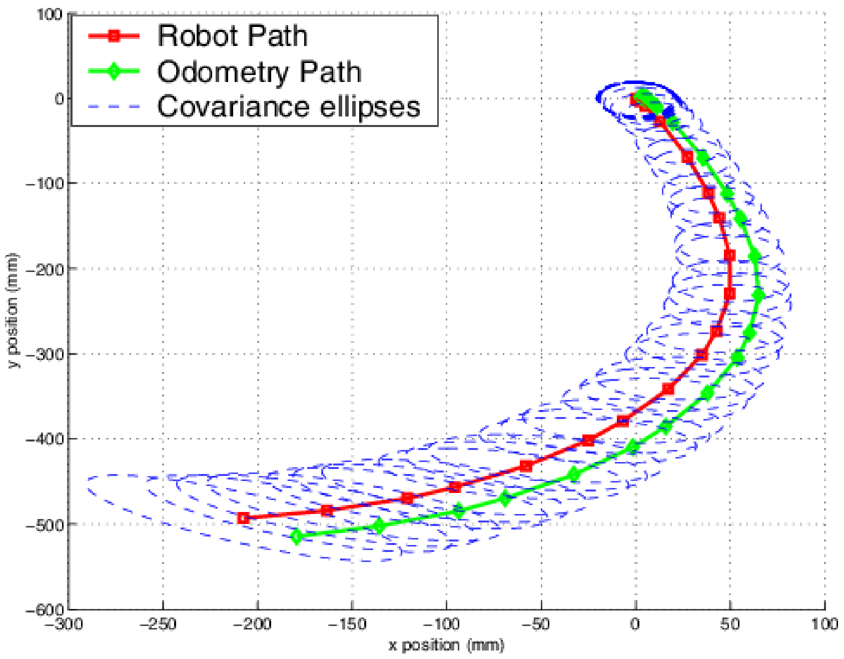


Motion model

- Remember the environment dynamics from the previous lecture

$$P(x_t|x_{t-1}, u_t) \rightarrow \text{Environment dynamics}$$

  – Tells us the distribution of where we end up based on previous state and control input



(Doesn’t have to be gaussian)

Wheel Odometry (from lecture 4)

- For a differential drive robot:
  - The current position of the robot $\boldsymbol{p} = [x, y, \theta]^T$

$$\Delta x = \Delta s \cos\left(\theta + \frac{\Delta\theta}{2}\right)$$
$$\Delta y = \Delta s \sin\left(\theta + \frac{\Delta\theta}{2}\right)$$
$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b}$$
$$\Delta \mathrm{s} = \frac{\Delta \mathrm{s_r} + \Delta s_l}{2}$$

Where b is the distance between the two wheels

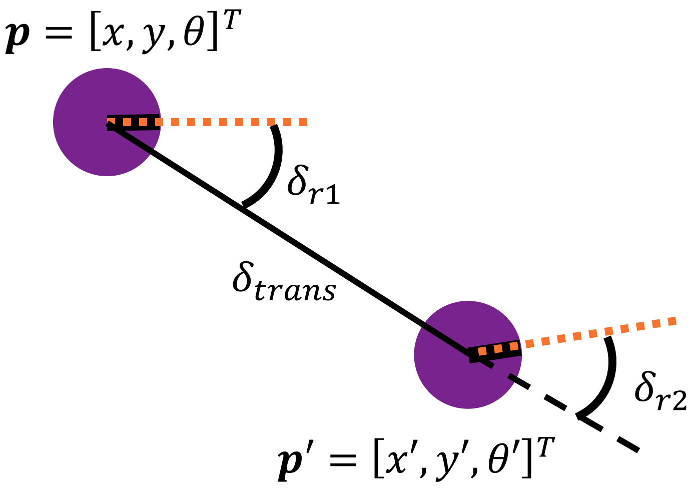
By using the relationship between Δs and $\Delta\theta$, we can substitute this further:

$$\boldsymbol{p}' = \boldsymbol{p} + \begin{bmatrix} \frac{\Delta \mathrm{s_r} + \Delta s_l}{2} \cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta \mathrm{s_r} + \Delta s_l}{2} \sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix}$$

The odometry model for a mobile robot

- The robot moves from $\boldsymbol{p}$ to $\boldsymbol{p}'$
- Odometry information: $u = (\delta_{r1}, \delta_{trans}, \delta_{r2})$
  – Here we assume the controls are angles and distance, not positions

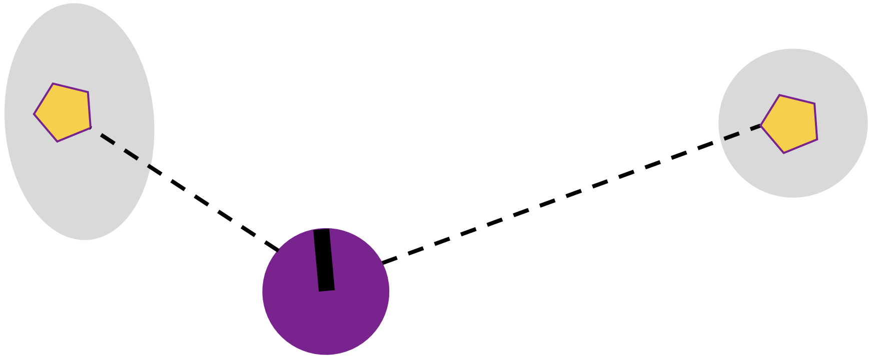
$$\delta_{trans} = \sqrt{(x' - x)^2 + (y' - y)^2}$$
$$\delta_{r1} = \text{atan2}(y' - y, x' - x) - \theta$$
$$\delta_{r2} = \theta - \theta' - \delta_{r1}$$



Observation model

- Remember the observation dynamics from the previous lecture

$$P(z_t|x_t) \rightarrow \text{Observation dynamics}$$



(Doesn’t have to be gaussian)

Three main paradigms of SLAM

KALMAN FILTER

PARTICLE FILTER

GRAPH-BASED

KALMAN FILTER
PARTICLE FILTER
GRAPH-BASED

The Kalman filter

- Is a recursive Bayes filter

- **Prediction**

$$\overline{Bel}(x_t) = \int P(x_t | x_{t-1}, u_t) \cdot Bel(x_{t-1}) dx_{t-1}$$

- **Correction / update**

$$Bel(x_t) = \eta P(z_t | x_t) \overline{Bel}(x_t)$$

The Extended Kalman filter

KALMAN FILTER PARTICLE FILTER GRAPH-BASED

Prediction Step:

- $\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_{k-1})$
- $P_{k|k-1} = F_{k-1} P_{k-1|k-1} F_{k-1}^T + Q_{k-1}$

Update Step:

- $y_k = z_k - h(\hat{x}_{k|k-1})$
- $H_k = \frac{\partial h}{\partial x}\Big|_{\hat{x}_{k|k-1}}$
- $S_k = H_k P_{k|k-1} H_k^T + R_k$
- $K_k = P_{k|k-1} H_k^T (S_k)^{-1}$
- $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k y_k$
- $P_{k|k} = (I - K_k H_k) P_{k|k-1}$

- $\hat{x}_{k|k-1}$: Predicted state estimate
- $P_{k|k-1}$: Predicted error covariance matrix
- f: Nonlinear system function
- u_{k-1}: Control input at time $k-1$
- F_{k-1}: Jacobian of f at $\hat{x}_{k-1|k-1}$
- Q_{k-1}: Process noise covariance matrix
- z_k: Measurement at time k
- h: Nonlinear measurement function
- R_k: Measurement noise covariance matrix
- H_k: Jacobian of h at $\hat{x}_{k|k-1}$
- S_k: Innovation covariance
- K_k: Kalman gain
- I: Identity matrix

Using EKF for SLAM

KALMAN FILTER | PARTICLE FILTER | GRAPH-BASED

- ..how?

Using EKF for SLAM

KALMAN FILTER PARTICLE FILTER GRAPH-BASED

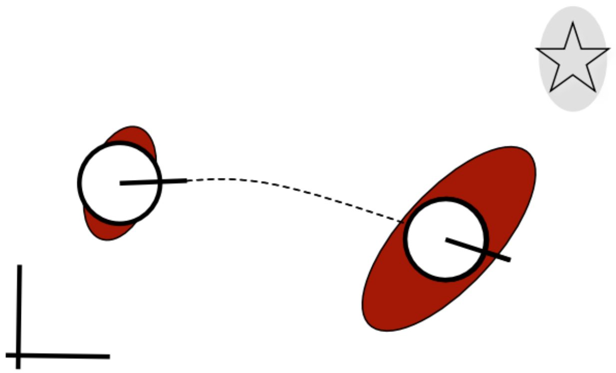
- For localization, we just have two variables
  - 3x1 pose vector $\boldsymbol{x}_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}$
  - 3x3 covariance matrix $C_k = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 & \sigma_{x\theta}^2 \\ \sigma_{yx}^2 & \sigma_y^2 & \sigma_{y\theta}^2 \\ \sigma_{\theta x}^2 & \sigma_{\theta y}^2 & \sigma_\theta^2 \end{bmatrix}$
- In SLAM we simply add more landmarks to the state
  - We predict the movement of the landmarks in addition to our own movements
- We can add hundreds of dimensions

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} \end{bmatrix}_k$$

SLAM – building a map

- State prediction
- Measurement prediction
- Observation
- Data association
- Update
- Integration of new landmarks

- Use our motion model
  - How has the robot moved w.r.t. our landmarks



$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} \end{bmatrix}_k$$

State prediction

Measurement prediction

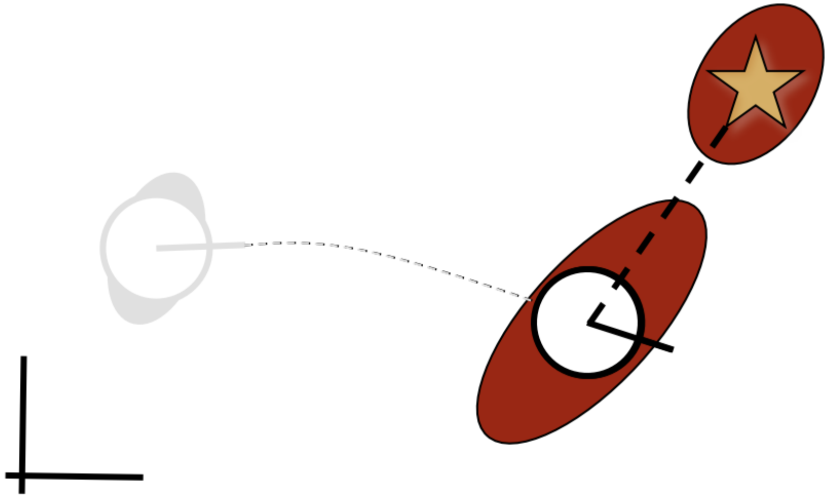
Observation

Data association

Update

Integration of new landmarks

- The expected observation $\overline{z_k}$
  - Where do we expect to find our landmarks



State prediction

Measurement prediction

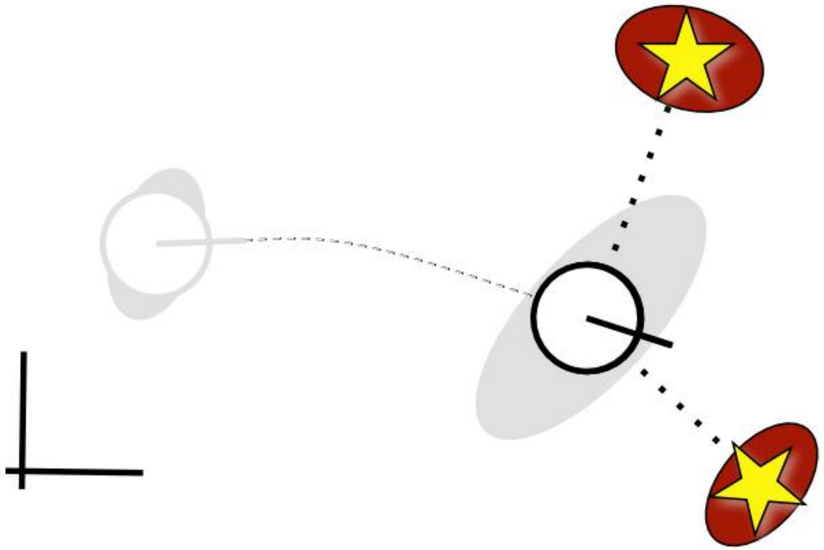
Observation

Data association

Update

Integration of new landmarks

- Observe the landmarks using your sensors z_k



$$z_k = [z_1, z_2]^T = [x_1, y_1, x_2 y_2]^T$$
$$R_k = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}$$

State prediction

Measurement prediction

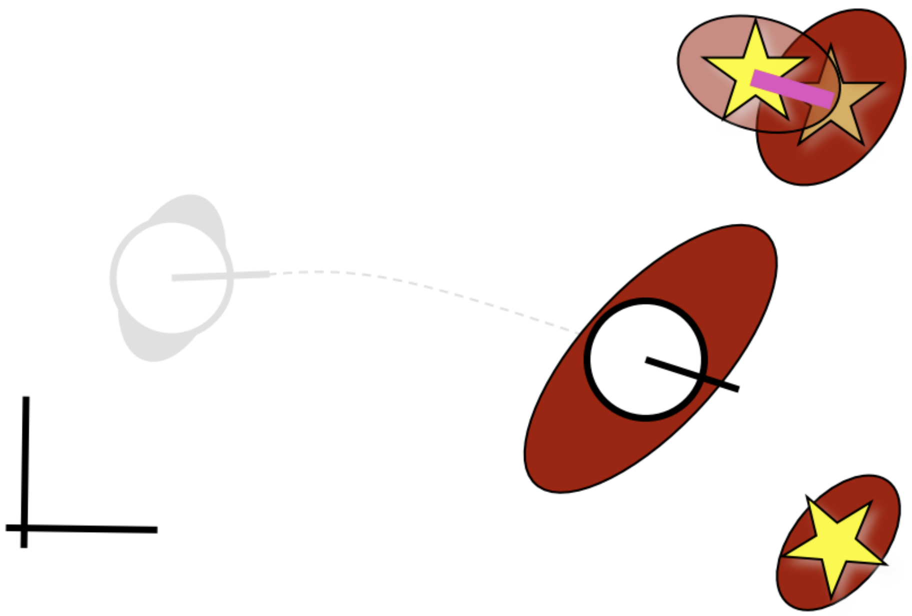
Observation

Data association

Update

Integration of new landmarks

- Associate the predicted measurements $\overline{z_k}$



State prediction

Measurement prediction

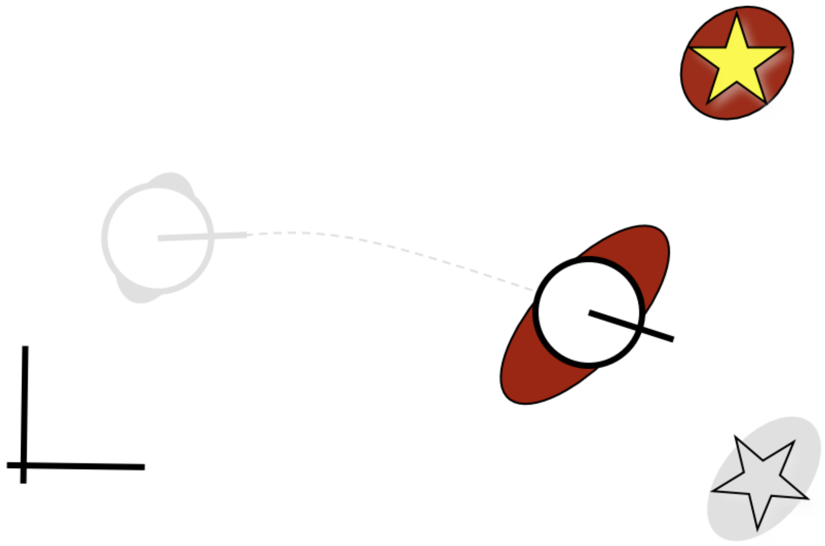
Observation

Data association

Update

Integration of new landmarks

- Update the filter – the correction step



State prediction

Measurement prediction

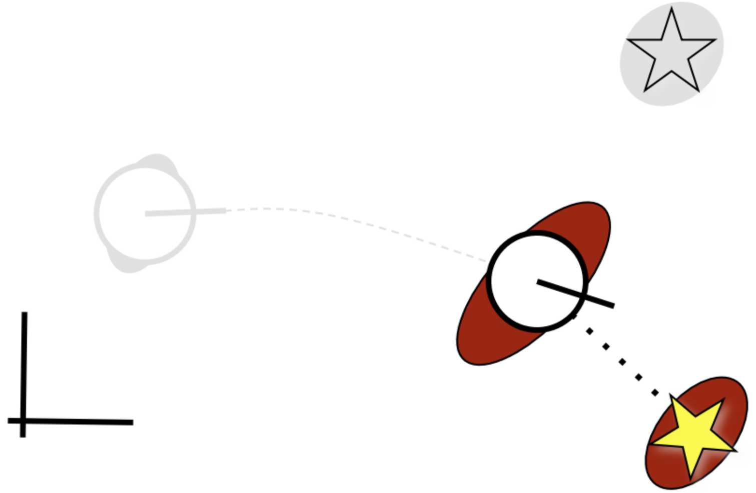
Observation

Data association

Update

Integration of new landmarks

- Add the new landmarks to the state vector and covariance matrix



$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \\ \mathbf{m}_{n+1} \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} & C_{RM_{n+1}} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} & C_{M_1M_{n+1}} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} & C_{M_2M_{n+1}} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} & C_{M_nM_{n+1}} \\ C_{M_{n+1}R} & C_{M_{n+1}M_1} & C_{M_{n+1}M_2} & \cdots & C_{M_{n+1}M_n} & C_{M_{n+1}} \end{bmatrix}_k$$

State prediction

Measurement prediction

Observation

Data association

Update

Integration of new landmarks

Three main paradigms of SLAM

KALMAN FILTER

PARTICLE FILTER

GRAPH-BASED

KALMAN FILTER

PARTICLE FILTER

GRAPH-BASED

Particle filter

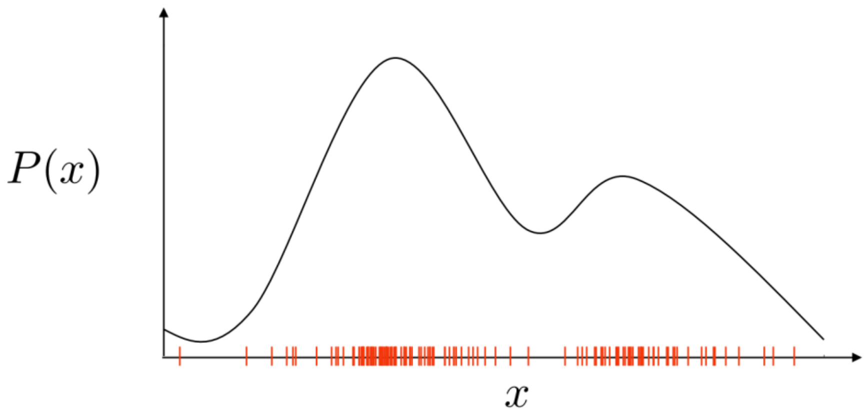
- Dense particles means higher probability mass

- Weigh the importance of each particle to modulate our distribution

- Weighted particles
  - $S = \{(s^i, w^i) | i = 1, \dots, N\}$

State hypothesis (particle)

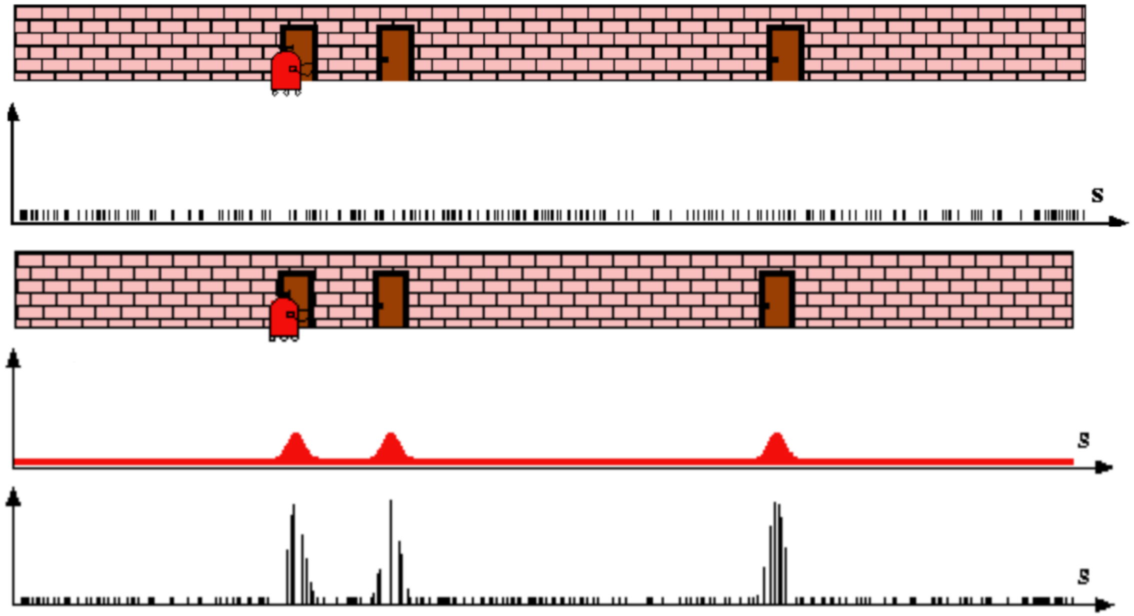
Importance weight



- The samples of hypotheses can then be our posterior
  - $P(x) = \sum_{i=1}^{N} w^i f(s^i)$

Particle filter

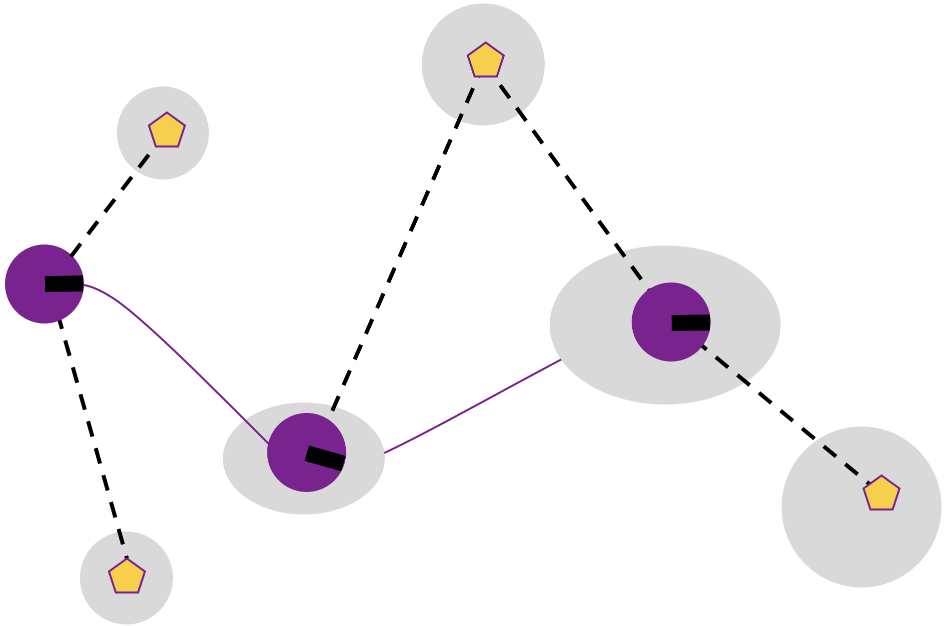
- Estimate the robot location in the existing, known map



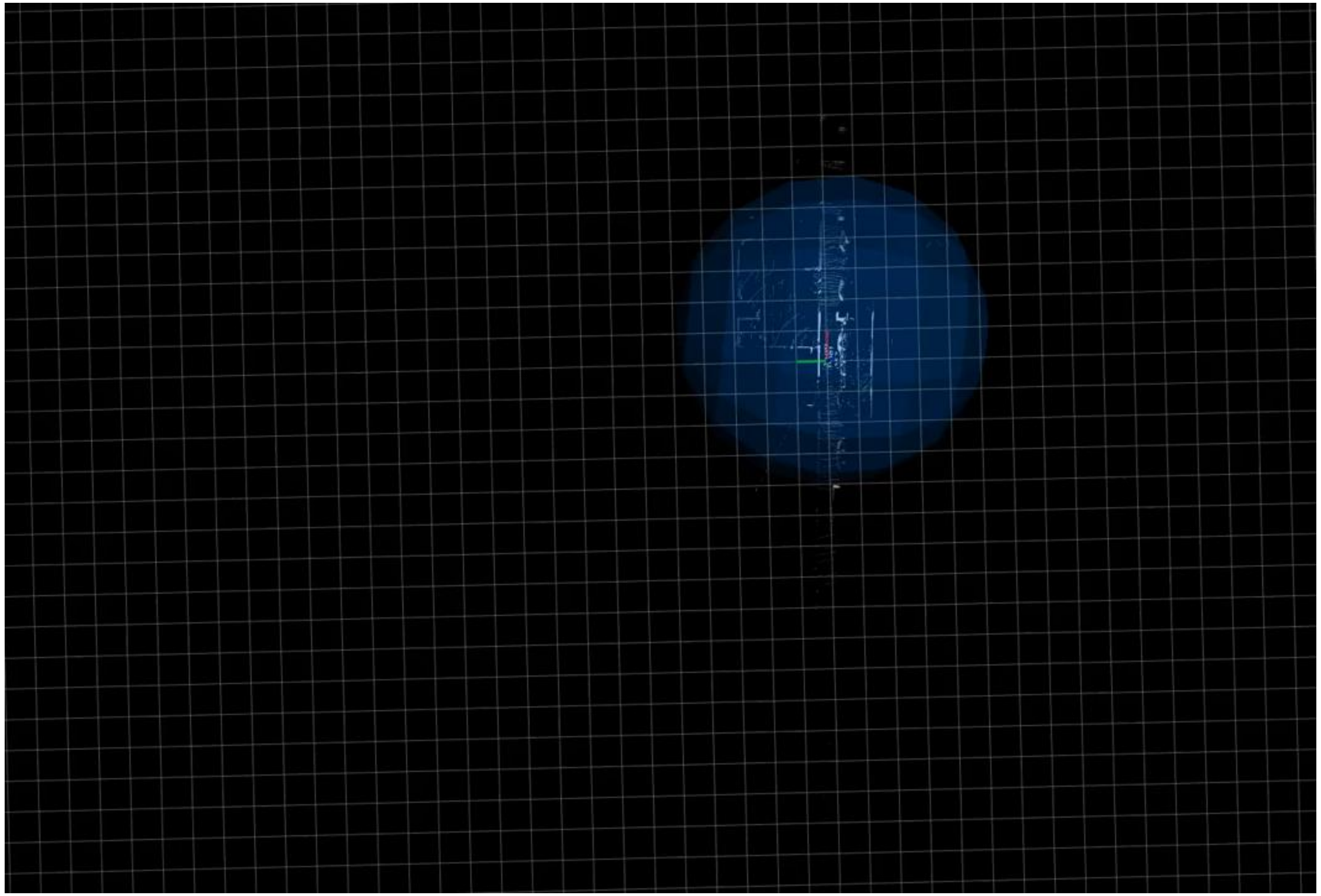
- Or track the landmarks with (more) particles

Loop-closure

- After a long exploration trajectory, the robot revisits a previously mapped area
  - Gives us an opportunity to correct our path for potential drift
  - … if we can provide robust data association
  - Similar to running full SLAM when you re-observe your landmarks



Loop-closure



Exercises

- Create your own map in the simulation using SLAM
  - Launch the simulation with the slam argument set to True

    ros2 launch my_turtlebot turtlebot_simulation.launch.py slam:=True
  - Save the map

    ros2 run nav2_map_server map_saver_cli -f map
  - You should get two files (map.pgm and map.yaml) – put them in the map directory of the my_turtlebot package and launch the simulation again without the slam argument
  - For more information, see the official guide: https://ros2-industrial-workshop.readthedocs.io/en/latest/_source/navigation/ROS2-Cartographer.html

- Use the odometry topic provided by the simulation to accumulate a map (if you don't already have your own localization/odometry)
- Use a counter to define if a cell is free or occupied

- Continue on your own localization and map integration
  - Using your own localization, accumulate a map when you drive around in the environment