Simultaneous Localization and Mapping (SLAM)

Ricardo B. Sousa
João G. Martins
Héber Miguel Sobreira
Marcelo R. Petry
António Paulo Moreira

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Outline

1. Context
2. Probabilities Recap
3. Dynamic Bayesian Networks (DBN)
4. Extended Kalman Filter (EKF) SLAM
5. Data Association
6. EKF SLAM with Unknown Association
7. Unscented Kalman Filter (UKF)
8. Particle Filter
9. Pose Graph SLAM
10. Field Experiments on SLAM and Mobile Robots

Context

- How to achieve autonomous navigation?
  - **Perception** **:** extract meaningful data from the on-board sensors
  - **Localization** **:** determine the robot pose in the environment
  - **Cognition** **:** decide how to act and achieve goals
  - **Motion Control** **:** modulate the motor outputs to achieve the desired trajectory

↓

- The robot should be able to recognize its surroundings by matching the sensor observations to prior knowledge, and then, localizing itself in the environment

↓

- **Mapping** **:** what is the world around me?
  - Sense from various positions
  - Integrate measurements to produce the map
- **Localization** **:** where am I in the world?
  - Sense and relate sensor readings to a world representation
  - Compute location relative to the model
  - Assumes a known model

→ **Chicken and egg problem!**

A map is needed to localize the robot and its pose estimate is needed to build the map…

Context: Simultaneous Localization and Mapping (SLAM)

- **SLAM:** simultaneous estimation of the state of a robot with on-board sensors and the construction of the environment model (i.e., the map) that the robot is perceiving through its sensors

- **Inputs:**
  - Control inputs u_t (wheeled / laser / visual / inertial odometry estimations, reference of the motor outputs, …)
  - Sensor observations z_t
    - Features from observing landmarks (points, geometric and semantic information)
    - Raw data (laser scans and depth data, sonars, radars)
    - …

- **State Estimation:**
  - Map (landmarks, occupancy grid, topological map, hybrid representation, …)
  - Robot State
    - Current robot pose (Online SLAM / Filtering) or even the full robot path (offline SLAM / Max a Posteriori)
    - **Note:** not necessarily only the robot pose! (robot velocity, calibration parameters, …)

Context: Uncertainty

- A deterministic approach to the SLAM problem would provide single-value solutions for the state estimation
- The state estimation is usually dependent on modelling the robotic system
  - A model is typically an abstraction of a more complex entity
  - Disturbances affect both the measurements and the system's model

- Models of the systems themselves are the ones that produce errors! (if perfect model, no errors)

- Predicted ≠ real behaviour
  - Errors in the model and noise in the sensors affect the real actions that the robot will take relative to the desired ones

PROBABILISTIC INFERENCE

(instead of single values, we have a set of possible solutions)

Probabilities Recap

- Probability does not need to fit into a certain shape → only needs to obey certain axioms
- An event E characterizes a subset of the possible outcome
- Probability is a function going from each measurable subset of the event space Ω to the interval $[0,1]$

1. $P(E) \geq 0$
   a. Probability of an event is greater or equal than 0
   b. An event E cannot have negative probability
   c. Event is either true or false (occurred or not)
2. $P(\cup_i E_i) = \sum_i P(E_i)$
   a. Probability of the union of a set of disjoint events is the sum of probabilities of the events
   b. Disjoint events are mutually exclusive
3. $P(\Omega) = 1$
   a. Probability of the outcome of the event being in the set of possible outcomes is 1

Probabilities Recap: Conditional Probability

- **Example:** record samples of measured Wi-Fi signal strengths S_i in different locations L_j
  - In each location L_j, measure the Wi-Fi signal strengths to be able to assign a number $P_{L_j}(S_i) = \frac{N_i}{N}$
  - S_i are the signal strength intervals considered, N_i the number of times the signal strength S_i was observed at the location L_j, and N the total number of experiments at the location L_j

$$P_{L_i}(S_i) := P(S_i|L_j)$$

- $P(S_i|L_j)$ is the conditional probability that characterizes the distribution over the signal strength S_i **GIVEN** the knowledge of the location
  - L_j is the part of the event that controls the shape of the probability function
  - $P(S_i|L_j)$ **REMAINS** the probability distribution **OVER** the signal strength, **NOT** on the location!

Probabilities Recap: Chain Rule

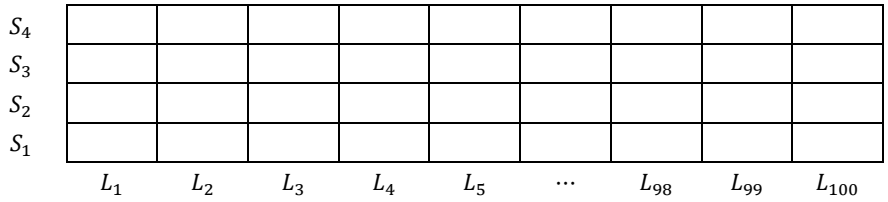
- $P(S_i, L_j)$ represents how likely to visit a location L_j and from there measuring a signal strength in the interval S_i
- $P(S_i, L_j) \leq P(L_j)$
  - $P(L_j)$ can be viewed as the probability of visiting a location L_j and measuring **ANY** signal strength
  - $P(L_j) = \sum_i P(S_i, L_j)$
- Occurring the event $\{S_i, L_j\}$ is as much likely as being at the location L_j and from L_j measuring a signal strength S_i

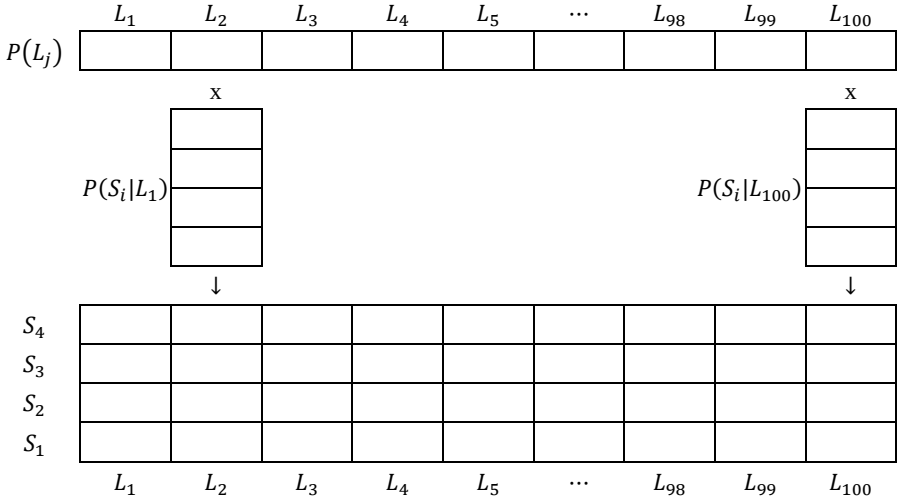
$$P(S_i, L_j) = P(L_j) \cdot P(S_i|L_j)$$

- Experiments of obtaining the statistics $P(S_i|L_j)$ for the signal strength are completely independent from estimating the probability $P(L_j)$

Probabilities Recap: Joint Distribution

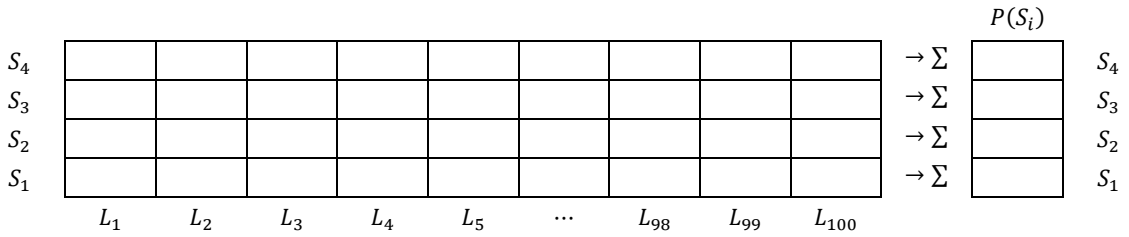
- $P(S_i, L_j)$ represents how likely to visit a location L_j and from there measuring a signal strength in the interval S_i
- $P(S_i, L_j)$ is a joint distribution of the joint event $\{S_i, L_j\}$ characterizing all possible pairs of signal strengths and signals





- Joint distribution can be computed using the chain rule

Probabilities Recap: Marginals



- $P(S_i) = \sum_j P(S_i, L_j)$ is the probability of measuring S_i from **ANY** location
- Joint event $\{S_i, L_j\}$ are disjoint given the physical meaning of measuring a signal strength interval and visiting a location
- **Marginalization:** process of suppressing a variable from the joint distribution
  - Requires summing over the possible values of one variable to determine the marginal contribution of another
  - In the table shown here, marginalization reduces one dimension on the joint distribution (3D → 2D)

Probabilities Recap: Independence

$$P(A|B) = P(A)$$

- A is independent of B if, whatever the value of B, A remains
  - Knowing B tells nothing about A → uncorrelated events!
  - **Example:** an independent phenomenon would be measuring a Wi-Fi signal strength and the floor at which the elevator is in a building (assuming the elevator does not interfere in the signal propagation)
- A is independent of B → $P(A,B) = P(A|B)P(B) = P(A)P(B)$

Independence test:

1. Compute the marginal over the first variable ($P(A_i) = \sum_j P(A_i, B_j)$)
2. Compute the marginal over the second variable ($P(B_j) = \sum_i P(A_i, B_j)$)
3. Compute the joint distribution from the marginals as if the variables were independent ($P(A,B) = P(A)P(B)$?)
   a. If the resulting table is the same as the original one, variables are independent!
   b. Otherwise, they are not

Probabilities Recap: Conditional Independence

- Suppose we have 2 access points A and B at different locations and transmit at different frequencies
  - No interference with each other → only element influencing signal strength is the location
- What about $P(S^A, S^B)$? Are these two variables independent ("if I know S^A, can I tell anything about S^B")?
  - Signal strengths depend on the location → strength and location are correlated
  - If S^A influences L, and S^B is influenced by L → $\{S^A, S^B\}$ are correlated through the location
- What if I know the location?
  - $P(S^A, S^B|L) = P(S^A|L)P(S^B|L)$ → the two signals are independent if I know the location
  - Exploiting the structure of the problem (physical properties, state relation, etc.), we can figure it out if two variables are correlated or not

Probabilities Recap: Probability Identities

$P(A,B) = P(A|B)P(B)$ $\quad$ $P(A,B|C) = P(A|B,C)P(B|C)$

$P(A|B) = \frac{P(A,B)}{P(B)}$ $\quad \Longrightarrow \quad$ $P(A|B,C) = \frac{P(A,B|C)}{P(B|C)}$

$P(A) = \sum_i P(A,B_i)$ $\quad$ $P(A|C) = \sum_i P(A,B_i|C)$

Probabilities Recap: Summary

- Axioms
  - $P(E) \geq 0$
  - $P(\cup_i E_i) = \sum_i P(E_i)$
  - $P(\Omega) = 1$

- Chain rule: $P(A, B) = P(A|B)P(A)$

- Marginalization: $P(A) = \sum_i P(A, B_i)$ (collapse variables from the joint probability)

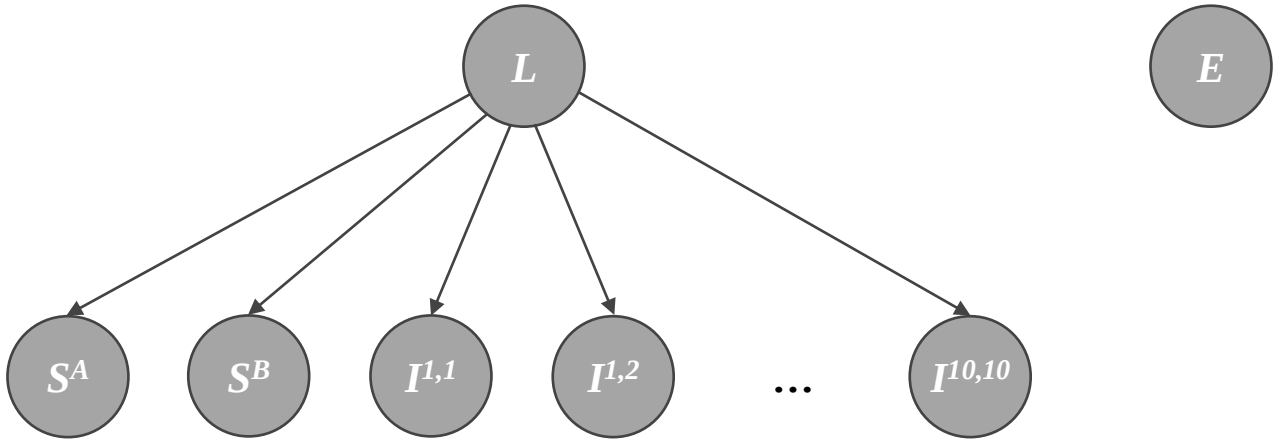
- Conditioning: $P(A|B) = \frac{P(A,B)}{P(B)}$ (slicing the joint probability and selecting the parts of the joint probability that are consistent from the evidence that we got)

Dynamic Bayesian Networks (DBN): Bayesian Networks

- **Bayesian Networks:** probabilistic graphic models (graphs) that are compact representations of joint probability distributions
  - Nodes are random variables
  - Lack or arcs represent conditional independence assumptions
- While Markov Random Fields (MRF) are undirected graphic models, Bayesian Networks are directed graphs:
  - Give an indication of **CAUSE → EFFECT**
  - Represent a set of random variables and their conditional dependencies through a Directed Acyclical Graph (DAG) with no loss of information

Dynamic Bayesian Networks (DBN): Example of a Bayesian Network

- Robot with a grayscale camera (10x10 px resolution, 16 intensity levels) and capable of measuring Wi-Fi signal strength
- 4 possible interval levels of signal strength for each Access Point (AP)
- 2 APs
- 6 floors
- 100 discrete locations



- $P(L, S^A, S^B, I^{1,1}, I^{1,2}, \dots, I^{10,10}, E)$
  1. $\#\text{disjoint events} = 100 \times 4 \times 4 \times 16^{10\times10} \times 6 = 9600 \times 16^{100} = 2.479 \times 10^{124}$

     ↓ *by applying the chain rule and leveraging the conditional independence…*

- $P(L, S^A, S^B, I^{1,1}, I^{1,2}, \dots, I^{10,10}, E) = P(L) \cdot P(S^A|L) \cdot P(S^B|L) \cdot P(I^{1,1}|L) \cdot \dots \cdot P(I^{10,10}|L) \cdot P(E)$
  1. $\#\text{disjoint events} = 100 + (4 \times 100) + (4 \times 100) + (16 \times 10 \times 10 \times 100) + 6 = 160906$
  2. Much less variables for the joint distribution while capturing the **SAME** information

Dynamic Bayesian Networks (DBN): Inference on Bayesian Networks

$$P\left(S^A | I_7^{1,1}\right) = ?$$

1. Compute the joint probability

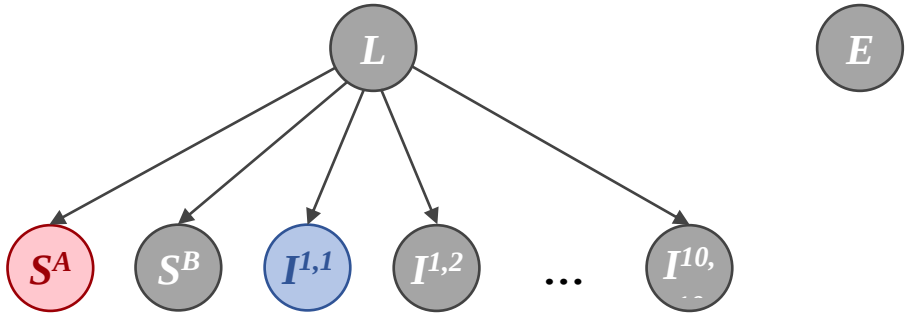
$$P\left(L, S^A, S^B, I_7^{1,1}, I^{1,2}, \dots, I^{10,10}, E\right) = P(L) \cdot P(S_7^A | L) \cdot P(S^B | L) \cdot P(I^{1,1} | L) \cdot \dots \cdot P(I^{10,10} | L) \cdot P(E)$$

2. Marginalize the variables we do not need

$$P\left(S^A, I_7^{1,1}\right) = \sum_L \sum_{S^B} \sum_{I^{1,2}} \dots \sum_{I^{10,10}} \sum_E P\left(L, S^A, S^B, I_7^{1,1}, I^{1,2}, \dots, I^{10,10}, E\right)$$

3. Use the conditioning rule to get the answer

$$P\left(S^A | I_7^{1,1}\right) = \frac{P\left(S^A, I_7^{1,1}\right)}{\sum_{S^A} P\left(S^A, I_7^{1,1}\right)}$$



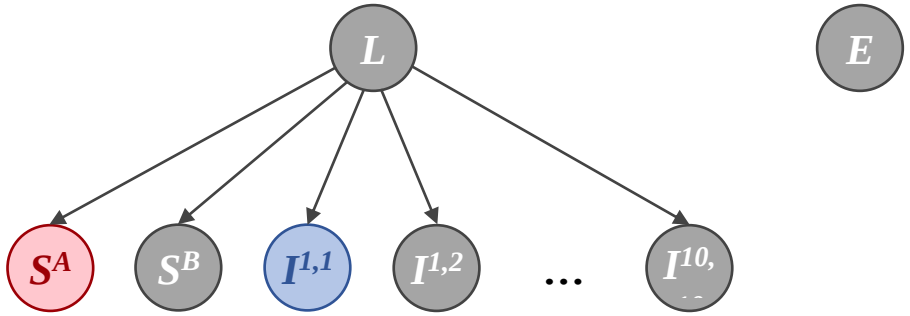
Dynamic Bayesian Networks (DBN): Inference on Bayesian Networks

$$P\left(S^A|I_7^{1,1}\right) =?$$

- What if we know the location? $P\left(S^A|I_7^{1,1}\right) = P(S^A|L)P\left(L|I_7^{1,1}\right)$ (conditional independence)

1. Chain rule: $P\left(L, I_7^{1,1}\right) = P\left(I_7^{1,1}|L\right) \cdot P(L)$
2. Conditioning: $P\left(L|I_7^{1,1}\right) = \frac{P\left(L,I_7^{1,1}\right)}{\sum_{L_j} P\left(L_j,I_7^{1,1}\right)}$
   - Alters the prior about the location
   - Incorporates the knowledge of the intensity $I_7^{1,1}$ in the location estimation
3. Chain rule: $P\left(S^A|I_7^{1,1}\right) = P(S^A|L)P\left(L|I_7^{1,1}\right)$
   - Determines the signal strength from the improved location estimate $L|I_7^{1,1}$

Note: $P(S^A|L)$ and $P\left(I_7^{1,1}|L\right)$ are usually known (or can be modelled) – dependent on the observation model of the sensors.



Dynamic Bayesian Networks (DBN): Probabilistic Dynamical Systems

System Model

$f(x_{t-1}, u_{t-1})$: transition function

$h(x_t)$: observation function

Deterministic solution

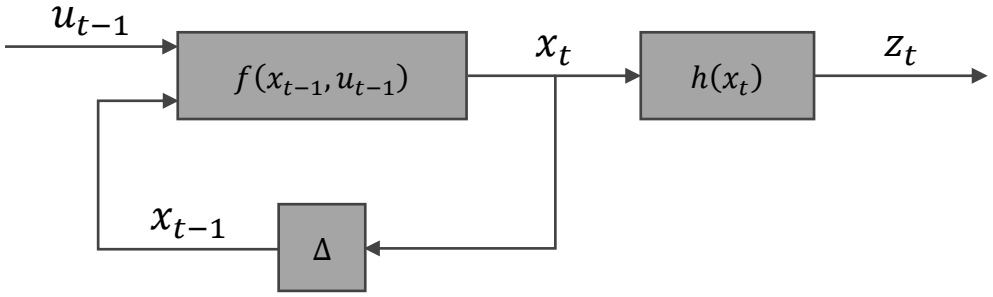
Probabilistic Dynamical System

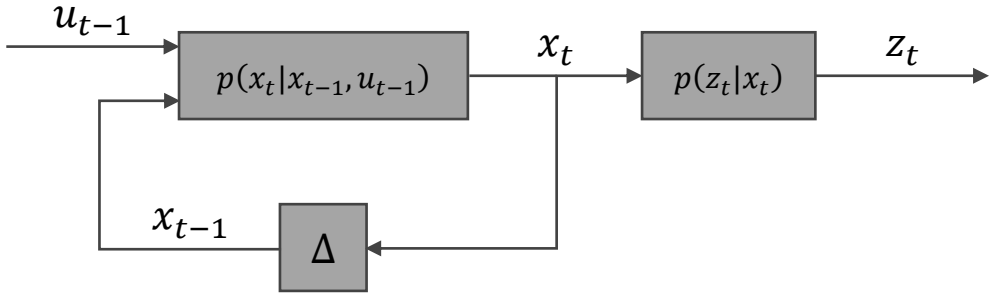
$p(x_t|x_{t-1}, u_{t-1})$: transition probabilistic model

$p(z_t|x_t)$: observation probabilistic model

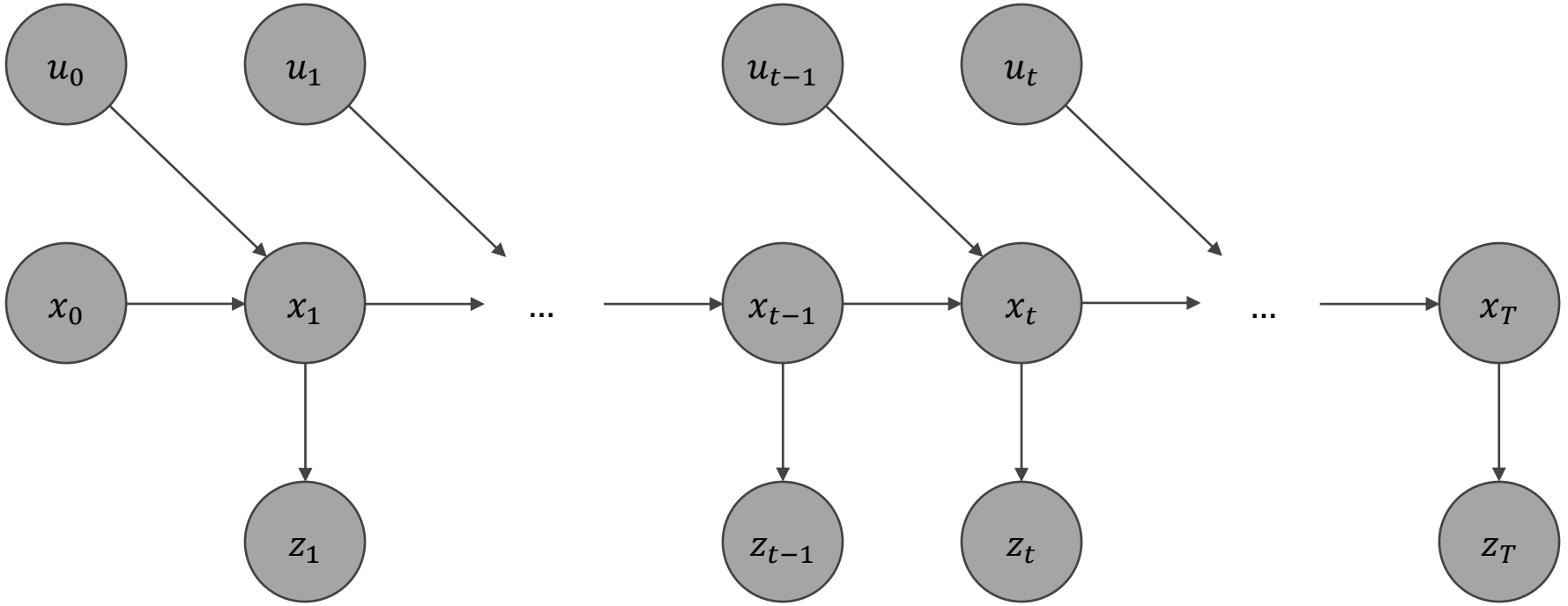
All variables become stochastic (w/ uncertainty)

If no uncertainty present, the probabilistic models degenerate to the deterministic models





Dynamic Bayesian Networks (DBN)



- x_0: known initial state distribution
- $x_{0:T}$: e.g., robot trajectory (or other variables, not necessarily only the robot pose!)
- Transition model correlates previous control u_{t-1} and state x_{t-1} with the current state x_t
- Observation model correlates the observation z_t and the current state x_t
- Recursive structure depending on time T

Dynamic Bayesian Networks (DBN): Inference on a DBN

	Query	***Known***
Filtering	$p(x_T \mid u_{0:T-1}, z_{1:T})$	$u_{0:T-1}, z_{1:T}$
Smoothing	$p(x_t \mid u_{0:T-1}, z_{1:T}), 0 < t < T$	$u_{0:T-1}, z_{1:T}$
Max a Posteriori	$\underset{x_{o:T}}{\text{argmax}}\{ p(x_{0:T} \mid u_{0:T-1}, z_{1:T}) \}$	$u_{0:T-1}, z_{1:T}$

- **Filtering:** know the distribution of the current state x_T knowing everything else
- **Smoothing:**
  - Estimate a location in the past
  - Generally, provides a better estimate than filtering (has more knowledge available)
- **Max a Posteriori:** given all the knowledge, compute the best likely trajectory over the state

Dynamic Bayesian Networks (DBN): Belief

- **Belief:** $b(x_t) \triangleq p(x_t \mid \ldots) \equiv$ current belief on the current state
- Inference on DBNs usually work by updating the belief

- Examples of belief representations:
  - Discrete state: $x \in \{X_{1:n}\}$ → array of n floats, where each cell contains the probability of that state
  - Continuous state + distributed according to a Gaussian → mean array + covariance matrix
  - Continuous state + unknown distribution → approximate representation (e.g., weighted samples of state values)

Dynamic Bayesian Networks (DBN): Filtering

$$p(x_t \mid u_{0:t-1}, z_{1:t})$$

- Known:
  - $u_{0:t-1}$: control inputs
  - $z_{1:t}$: sensor observations
  - $p(x_t \mid x_{t-1}, u_{t-1})$: transitional model
  - $p(z_t \mid x_t)$: observation model
  - $b(x_{t-1}) = p(x_{t-1} \mid u_{0:t-2}, z_{1:t-1})$: current belief about the previous state

$$b(x_t) = p(x_t \mid u_{0:t-1}, z_{1:t}) = \eta_t \cdot p(z_t \mid x_t) \cdot \sum_{x_{t-1}} p(x_t \mid x_{t-1}, u_{t-1}) \cdot b(x_{t-1})$$

Where η_t is a scaling factor to ensure that the current belief about the current state $b(x_t)$ remains a probability distribution:

$$\eta_t = \frac{1}{p(z_t \mid u_{0:t-1}, z_{1:t-1})} = \frac{1}{\sum_{x_t}\left(p(z_t \mid x_t) \cdot \sum_{x_{t-1}} p(x_t \mid x_{t-1}, u_{t-1}) \cdot b(x_{t-1})\right)}$$

Dynamic Bayesian Networks (DBN): Filtering – Alternative Approach

- **Predict:** incorporate in the last belief $b_{t-1|t-1}$ the most recent control u_{t-1}
  1. Chain rule : $p(x_t, x_{t-1}|t-1) = p(x_t \mid x_{t-1}, u_{t-1}) \cdot p(x_{t-1}|t-1) = p(x_t \mid x_{t-1}, u_{t-1}) \cdot b_{t-1|t-1}$
  2. Marginalization : $b_{t|t-1} = p(x_t|t-1) = \sum_{x_{t-1}} p(x_t, x_{t-1}|t-1)$

```
BeliefType b_pred = BeliefType::Zero;    // discrete filtering case
for (x_j : X)
  for (x_i : X)
    b_pred[x_j] += b[x_i] * transitionModel(x_j,x_i,u);
```

- **Update:** incorporate in the predicted belief $b_{t|t-1}$ the new measurement z_t
  - Chain rule : $p(x_t, z_t \mid t) = p(z_t|x_t) \cdot p(x_t| t-1) = p(z_t|x_t) \cdot b_{t|t-1}$
  - Condition on the actual measurement : $b_{t|t} = p(x_t \mid t) = \frac{p(x_t, z_t \mid t)}{p(z_t \mid t)} = \frac{p(x_t, z_t \mid t)}{\sum_{x_t} p(x_t, z_t \mid t)}$

```
float normalizer = 0;                    // discrete filtering case
for (x_i : X)
{
  b[x_i] = b_pred[x_i] * observationModel(z,x_i)
  normalizer += b[x_i]
}
b *= 1 ./ normalizer
```

Extended Kalman Filter (EKF) SLAM

- No prior knowledge of the map (position of the landmarks not known)

- **EKF SLAM:** filtering inference over the state (robot pose and landmarks position) on the DBN
  1. Localization
     a) Predict: incorporate new control
     b) Correct / update: incorporate new measurements – innovation $\left(z_t - h\left(\mu_{t|t-1}\right)\right)$
  2. Mapping: add new landmarks to the state

Extended Kalman Filter (EKF) SLAM: Example

- **Scenario:**
  - Robot in a 2D plane, translational and rotational velocities ($u^{[x]}$ and $u^{[\theta]}$, respectively)
  - Sensor measurements:
    - Uniquely distinguishable landmarks ($\equiv$ w/ known association)
    - 2D landmark sensor (observation is the landmark position relative to the robots)
    - Position of the landmark not known, i.e., no prior knowledge of the map

Extended Kalman Filter (EKF) SLAM: Example

- **Domains:**
  - Robot : $x_t^{[r]} = [R_t|t_t] \in SE(2) \Longrightarrow x_t^{[r]} = \begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} \in \mathbb{R}^3$
  - Landmarks : $x_t^{[n]} = \begin{pmatrix} x_t^{[n]} \\ y_t^{[n]} \end{pmatrix} \in \mathbb{R}^2, n = 1, \dots, N$
  - Full State Vector : $x_t = \begin{pmatrix} x_t^{[r]} \\ x_t^{[1]} \\ \vdots \\ x_t^{[N]} \end{pmatrix} \in \mathbb{R}^{(3+2\times N)}$
  - Controls : $u_t = \begin{pmatrix} u_t^{[x]} \\ u_t^{[\theta]} \end{pmatrix} \in \mathbb{R}^2$ (e.g., translational and rotation displacements of a differential robot)
  - Measurements : $z_t^{[m]} = \begin{pmatrix} x_t^{[m]} \\ y_t^{[m]} \end{pmatrix} \in \mathbb{R}^2, m = 1, \dots, M$

Extended Kalman Filter (EKF) SLAM: Example

- **Transition function:**

$$x_t = f(x_{t-1}, u_{t-1}) = \begin{pmatrix} x_{t-1} + u_t^{[x]} \cdot \cos(\theta_{t-1}) \\ y_{t-1} + u_t^{[x]} \cdot \sin(\theta_{t-1}) \\ \theta_{t-1} + u_t^{[\theta]} \\ x_{t-1}^{[1]} \\ \vdots \\ x_{t-1}^{[N]} \end{pmatrix}$$

  - Assumption that the landmarks do not move!

- **Measurement function:**

$$z_t^{[n]} = h^{[n]}(x_t) = R_t^T \left(x_t^{[n]} - t_t \right) = \begin{pmatrix} \cos(\theta_t) \cdot \left(x_t^{[n]} - x_t \right) + \sin(\theta_t) \cdot \left(y_t^{[n]} - y_t \right) \\ -\sin(\theta_t) \cdot \left(x_t^{[n]} - x_t \right) + \cos(\theta_t) \cdot \left(y_t^{[n]} - y_t \right) \end{pmatrix}$$

  - 1 measurement function for each sensor observation associated to a landmark already seen and stored in the state

Extended Kalman Filter (EKF) SLAM: Example

- **Control Noise:**

$$n_{u,t} \sim \mathcal{N}\left(0, \begin{pmatrix} \sigma_{u,x}^2 + \sigma_{x,t}^2 & 0 \\ 0 & \sigma_{u,\theta}^2 + \sigma_{\theta,t}^2 \end{pmatrix}\right)$$

  - Added components proportional to the robot's velocity $\left(\sigma_{x,t}^2, \sigma_{\theta,t}^2\right)$

- **Measurement Noise:**

$$n_z \sim \mathcal{N}\left(0, \begin{pmatrix} \sigma_z^2 & 0 \\ 0 & \sigma_z^2 \end{pmatrix}\right)$$

  - Assumed constant deviation (e.g., dependent on the standard deviation described in the datasheet of a 2D laser scan)

Extended Kalman Filter (EKF) SLAM: Example

- **Jacobians:**
  - $A_t = \left.\frac{\partial f(\cdot)}{\partial x}\right|_{x=\mu_{t-1|t-1}} = \begin{pmatrix} \frac{\partial f(\cdot)}{\partial x^{[r]}} & \frac{\partial f(\cdot)}{\partial x^{[1]}} & \cdots & \frac{\partial f(\cdot)}{\partial x^{[N]}} \end{pmatrix} = \begin{pmatrix} 1 & 0 & -\sin(\theta_{t-1}) \cdot u_t^{[x]} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cos(\theta_{t-1}) \cdot u_t^{[x]} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}_{(3+2\times N)\times(3+2\times N)}$
  - $B_t = \left.\frac{\partial f(\cdot)}{\partial u}\right|_{u=\mu_{u,t-1}} = \begin{pmatrix} \cos(\theta_{t-1}) & 0 \\ \sin(\theta_{t-1}) & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 0 \end{pmatrix}_{(3+2\times N)\times(2)}$
  - $C_t^{[n]} = \left.\frac{\partial h(\cdot)}{\partial x}\right|_{x=\mu_{t|t-1}} = \begin{pmatrix} -R_t^T & \frac{\partial R_t^T(\cdot)}{\partial \theta_t} \cdot \left(x_t^{[n]} - t_t\right) & 0 & \cdots & \frac{\partial h^{[n]}(\cdot)}{\partial x^{[n]}} & \cdots & 0 \end{pmatrix}$

Extended Kalman Filter (EKF) SLAM: Example

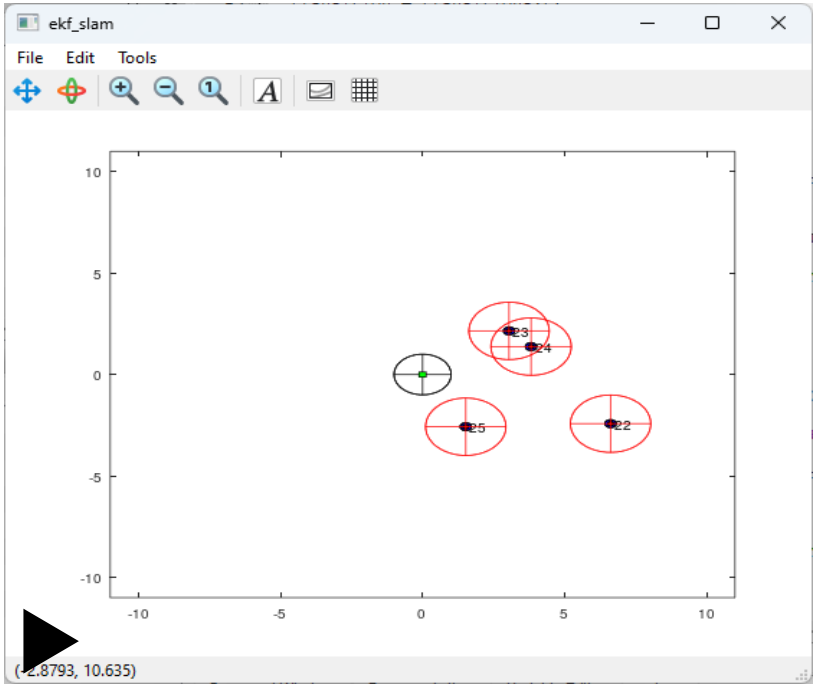
- **Data Association:**
  - z_t originates from a **SUBSET** of landmarks in the state
  - **Goal:** determine association between observations and state
    - *Is the observation I have seen before OR something new?*
    - Determine which measurement is generated by which landmark in the state
  - $j(m) \in [1, \dots, N]$
    - E.g., $j(3) = 5$ represents that the measurement 3 is generated from the landmark 5
    - Data association in this example is estimating the vector $j(m)$ → **for now, we assume a known correspondence (i.e., we know the assignment $j(m)$)**
  - $h(x_t) = \begin{pmatrix} h^{[j(1)]} \\ \vdots \\ h^{[j(M)]} \end{pmatrix}$
  - $C_t = \frac{\partial h(\cdot)}{\partial x} = \begin{pmatrix} \frac{\partial h^{[j(1)]}(\cdot)}{\partial x} \\ \vdots \\ \frac{\partial h^{[j(M)]}(\cdot)}{\partial x} \end{pmatrix}$ → note the ordering remains relative to the measurements

Extended Kalman Filter (EKF) SLAM: Example

- **Populating the Map:**
  - Required every time we see a new landmark
  - For convenience, let's define the vectors id2statemap and state2idmap (two mapping arrays)
  - Example:
    - At the initial iteration:
      - $\text{id2statemap} = (-1 \quad \cdots \quad -1)$
      - $\text{state2idmap} = (-1 \quad \cdots \quad -1)$
    - After 3 new landmarks observations:
      - $\text{id2statemap} = (3 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad 1 \quad -1 \quad 2 \quad -1 \quad \cdots \quad -1)$
      - $\text{state2idmap} = (7 \quad 9 \quad 1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad \cdots \quad -1)$
      - E.g., landmark 1 is represented in the state as index 3 **VERSUS** element 1 in state vector represents landmark 7
  - Updating the map:
    - Already known landmarks (part of the state) → use them to the update step of the EKF
    - New landmarks → add them to state after EKF correction (initialize them with the measurement covariance that originated the landmark)

Extended Kalman Filter (EKF) SLAM: Example

- **Implementation:**
  1. Prediction
  2. Correction
     a) FOR EACH observation
        - id2statemap
        - IF reobserved landmark
          - $\hat{z} = h(x_t)$
          - $C_t^{[j(m)]} = \cdots$
     b) Update filter
  3. Update the map with unobserved landmarks



Data Association: Problem Definition

- Previous example considered that the landmarks were uniquely identifiable! (assume a known association of which landmark caused the observation)
- What if the association is not known?
  - Must deal with distributions having multiple maxima (multi-modal distributions; no EKF / UKF)
  - OR choose the best possible association (less robust, e.g., if a wrong association occurs; but may still use EKF / UKF)
- **Data Association**
  - From the current estimate of the state and the history of the measurements (history especially important when only available bearing-only observations), estimate which state variable was responsible for a certain measurement
  - $\#\text{combinations} = \binom{n}{m} = \frac{n!}{m!(n-m)!}\Big|_{n>m}$
  - $j^* = \underset{j}{\text{argmax}}\left\{p\left(\hat{z}^{[j(1)]} = z^{[1]}, \dots, \hat{z}^{[j(M)]} = z^{[M]}\right)\right\}$
    - $j(i) \in [0,1, \dots, N], i = 1, \dots, M$, where $j(i) = 0$ means no correspondence
    - $z = z^{[1]}, \dots, z^{[M]}$: measurements
    - $\hat{z} = \hat{z}^{[1]}, \dots, \hat{z}^{[N]}$: landmark predictions

Data Association: Assignment Problem

- Assuming independent measurements (if the landmarks are far from each other, this assumption tends to be true) and a Gaussian distribution for the measurements:

$$p\left(z = \hat{z}^{[j]}\right) = \prod_{m=1}^{M} p\left(z^{[m]} = \hat{z}^{[j(m)]}\right)$$

$$\log\left(p\left(z = \hat{z}^{[j]}\right)\right) = \sum_{m=1}^{M} \log\left(p\left(z^{[m]} = \hat{z}^{[j(m)]}\right)\right) \propto -\sum_{m=1}^{M} \left(z^{[m]} - \tilde{\mu}_z\right)^T \cdot \left(\Sigma^{[j(m),j(m)]}\right)^{-1} \cdot \left(z^{[m]} - \tilde{\mu}_z\right)$$

- If we have a measurement m and a landmark n, the log-likelihood of an association becomes as follows:
  - **Cost matrix:** $A = \begin{pmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{pmatrix}$, where $a_{mn} = \left(z^{[m]} - \tilde{\mu}_z^{[n]}\right)^T \cdot \Omega^{n,n} \cdot \left(z^{[m]} - \tilde{\mu}_z^{[n]}\right)$
  - **Goals:** find exactly one element for each column where the sum of all entries is minimized, no column selected more than once, and matrix can be made square by padding a default value dependent on l_{miss}
  - **Log-Likelihood** $\triangleq \sum_m a_{m,j(m)}$

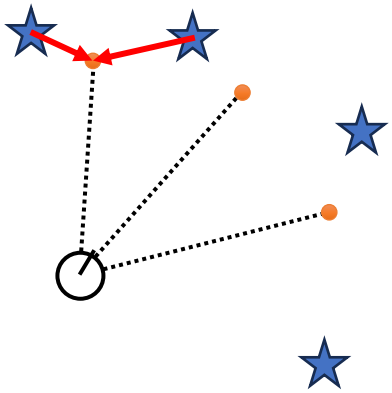
Data Association: Nearest Neighbour

- Nearest neighbour assigns to each landmark the closest measurement
  - $A = \begin{pmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{pmatrix}$
  - $a_{mn} = \left(z^{[m]} - \tilde{\mu}_z^{[n]}\right)^T \cdot \left(z^{[m]} - \tilde{\mu}_z^{[n]}\right)$

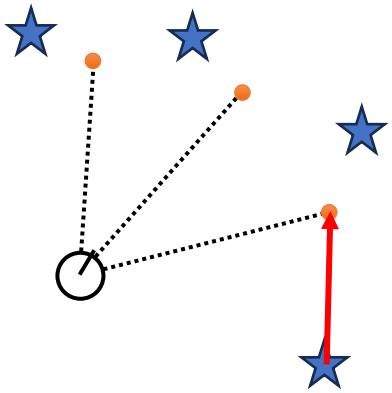
- If the eigen values of the measurement covariances are equal, a greedy association (by searching associations in the original cost matrix A defined in the previous slide) becomes nearest neighbour!

Data Association: Greedy Association

- Select for each column the tow where the association cost a_{mn} is the smaller value
- Might result in undesired configurations…
  - Multiple measurements can be assigned to the same landmark



  - A landmark can be assigned to a far measurement because nothing better is found



Data Association: Pruning Heuristics

- Why? **Wrong associations are bad!** → pruning heuristics can remove potential wrong associations

1. **Gating:** ignore all associations whose association cost is higher than a given threshold

2. **Best Friends:** an association should be the best (i.e., the minimum) on both row and column
   - Get rid of ambiguities

3. **Lonely best friends:** one measurement should be clearly the best
   - Difference between minimum and second minimum of row / column should be higher than a given threshold

Data Association: Other Registration Algorithms

- Normal Distribution Transform (NDT)
- Iterative Closest Point (ICP)

- SIFT features & RANdom Sample Consensus (RANSAC) in Photogrammetry

- …

EKF SLAM with Unknown Association

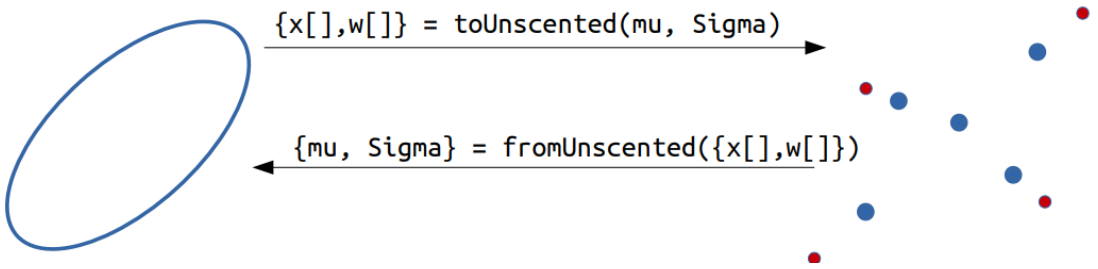
- In this case, the landmarks ID is not observable, when a new landmark appears, should be assigned a new UNIQUE ID

- **Implementation:**
  1. Prediction
  2. **Data Association (e.g., using greedy association w/ pruning heuristics)**
  3. Correction
     a) FOR EACH observation
        - id2statemap
        - IF reobserved landmark
          - $\hat{z} = h(x_t)$
          - $C_t^{[j(m)]} = \cdots$
     b) Update filter
  4. Update the map with unobserved landmarks

Unscented Kalman Filter (UKF): Unscented Transform

- Gaussian distributions can be represented as a set of control points + weights
- Each sigma point $\chi^{(i)}$ is characterized as follows (#sigma points $= 2 \cdot$ #state dimension $+ 1$):
  - Position $x^{(i)} \in \Omega$
  - Weight for the mean $w_m^{(i)} \in \mathbb{R}^+$
  - Weight for the covariance $w_c^{(i)} \in \mathbb{R}^+$
- Unscented Transform:
  - Sigma points → Gaussian parameters
    - $\mu = \sum_i w_m^{(i)} \cdot x^{(i)}$
    - $\Sigma = \sum_i w_c^{(i)} \cdot \left(x^{(i)} - \mu\right) \cdot \left(x^{(i)} - \mu\right)^T$
  - Gaussian parameters → Sigma points based on Cholesky Decomposition ($\alpha = 10^{-3}$, $\beta = 2$)
    - $\lambda = \alpha^2 n$
    - $L' = \text{chol}\left((n + \lambda) \cdot \Sigma\right)$ ⟶
      - $x^{(0)} = \mu$
      - $x^{(i)} = \mu + [L']_i \text{ for } i \in [1..n]$
      - $x^{(i)} = \mu - [L']_{n-i} \text{ for } i \in [n+1..2n]$

      ⟶
      - $w_m^{(0)} = \frac{\lambda}{n+\lambda}$
      - $w_c^{(0)} = w_m^{(0)} + (1 - \alpha^2 + \beta)$
      - $w_c^{(i)} = w_m^{(i)} = \frac{1}{2(n+\lambda)}$



Note: the weights do not depend on the Gaussian parameters, only on the parametrization of the Unscented Transformation!

Unscented Kalman Filter (UKF): Implementation

1. Prediction : `[sigma_pts,w_m,w_c] = predict(mu,sigma,controls)`
   a. Gaussian parameters → Sigma points $\chi_{t-1|t-1}^{(i)}$
   b. FOR EACH sigma point $\chi_{t-1|t-1}^{(i)}$, apply the transition model ($\chi_{t|t-1}^{(i)} = f\left(\chi_{t-1|t-1}^{(i)}\right)$)
2. Correction : `[mu, sigma] = update(sigma_pts,w_m,w_c,landmarks,observations)`
   a. Predict landmark for each sigma point ($z_t^{(i)} = h\left(x_{t|t-1}^{(i)}\right)$)
   b. Predicted sigma points ($z_t^{(i)}$) → Gaussian parameters (μ_z, Σ_z)
   c. State sigma points ($\chi_{t|t-1}^{(i)}$) → Gaussian parameters ($\mu_{t|t-1}, \Sigma_{t|t-1}$)
   d. Compute cross correlation of the joint distribution ($\Sigma_{xz} = \sum_i w_c^{(i)} \cdot \left(x_{t|t-1}^{(i)} - \mu_{t|t-1}\right) \cdot \left(z_t^{(i)} - \mu_z\right)^T$)
   e. Update the filter:
      - $\mu_{t|t} = \mu_{t|t-1} + \Sigma_{xz} \cdot \left(\Sigma_{z|x} + \Sigma_z\right)^{-1} \cdot (z_t - \mu_z)$
      - $\Sigma_{t|t} = \Sigma_{t|t-1} - \Sigma_{xz} \cdot \left(\Sigma_{z|x} + \Sigma_z\right)^{-1} \cdot \Sigma_{zx}$
      - Where $\Sigma_{z|x}$ represents the uncertainty introduced by the current measurement

Unscented Kalman Filter (UKF): Summary

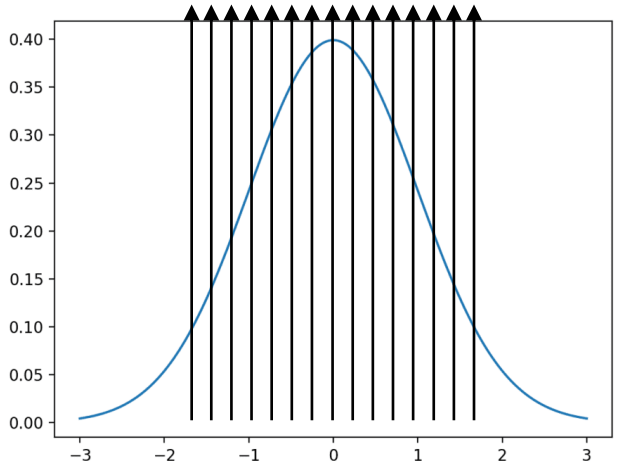
- UKF is a non-linear extension of the EKF
- Reported to better deal with non-linear transition and observation functions (sigma points provide more Degrees of Freedom than mean and covariance to characterize a probabilistic distribution)
- Still relies on the Gaussian assumption and uses pure Gaussian conditioning (conditioning requires a continuous distribution)
- Same complexity as EKF, with a constant factor slower in typical practical applications
- Derivative free! (no Jacobians needed)
- Still not optimal! (as the EKF)

Particle Filter: Introduction

- Particle filters are non-parametric representations of a probability distribution based on sampling
- Each particle is a sample and may contain different information:
  - Pose
  - Trajectory
  - Map
  - …
- An approximation of a density function can be obtained by a set of weighed samples
  - The denser the samples are in a region, the higher the probability of that region
  - $\int_E p(x)dx \cong \sum_{x^{[i]} \in E} w^{(i)}$
- **Why the interest on particles?**
  - Can represent arbitrary distributions
  - Easy to “visualize”
  - Easy to manipulate
  - Good for small state spaces

Particle Filter: Generating Samples

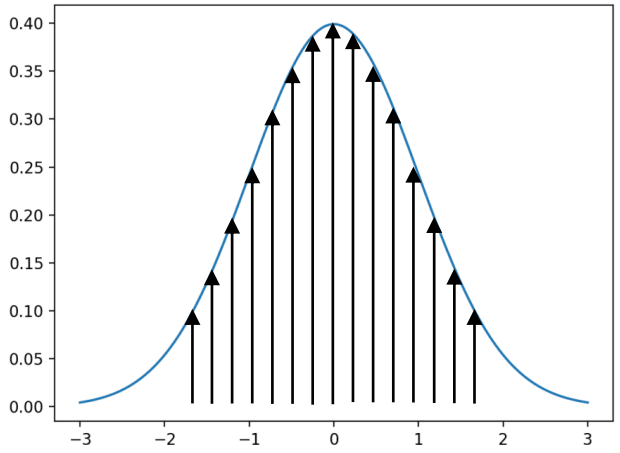
- Assuming the probability density function has a closed form (also valid for the discrete case):
  1. Compute the cumulative distribution : $P(x) = \int_{-\infty}^{x} p(x')dx'$
  2. Draw a sample from uniform distribution : $y^{(i)} \sim U(0,1)$
  3. Compute the inverse of the cumulative : $x^{(i)} = P^{-1}\left(y^{(i)}\right)$



- The sampling distribution may not be known (cannot compute the inverse cumulative) → **Importance Sampling:**
  1. Sample from a known distribution $\pi(x)$ possibly close to $p(x)$: $x^{(i)} \sim \pi(x)$
  2. Compute a weight by evaluation $\pi(x)$ and $p(x)$ at the sampled point:

$$w^{(i)} = \frac{p\left(x^{(i)}\right)}{\pi\left(x^{(i)}\right)}$$

  - $\pi\left(x^{(i)}\right)$: proposal distribution
  - $p\left(x^{(i)}\right)$: target distribution



Note: proposal $\pi(x)$ should cover all relevant portion of target distribution $p(x)$, otherwise, some feasible samples might not be generated!

Particle Filter: Resampling

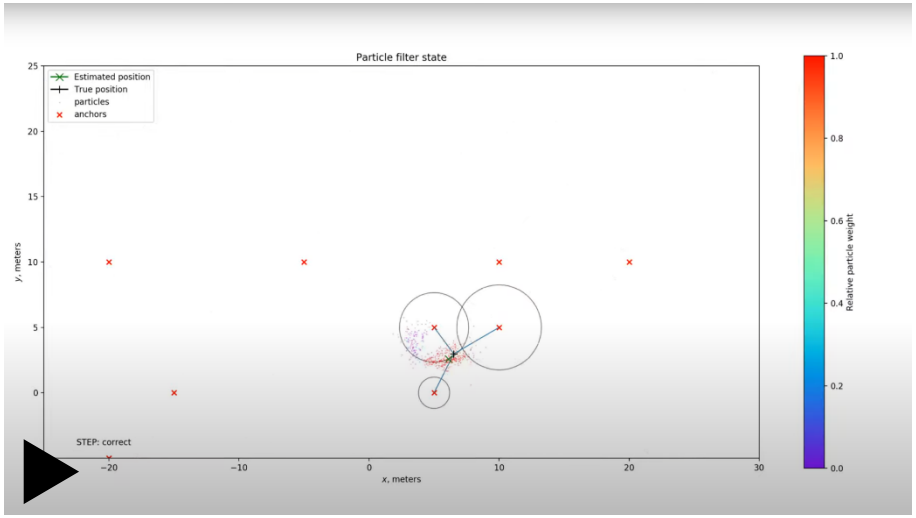
- In order to turn a weighed sample set into an unweighted one (uniform):
  - Repeat samples having high weights
  - Suppress samples with low weight

- Resampling can be done by drawing a set of indices j from the normalized weights distribution, repeating the samples according to the indices generated through the sampling procedure:
  1. Input weighed sample set ($\{\langle w^{(i)}, x^{(i)} \rangle\}$)
  2. Sample set of indices with probability proportional to the weights ($\{j^{(i)}\} = \text{uniformSample}(\{w^{(i)}\})$)
  3. New sample in position i is the old sample in position $j[i]$ ($\tilde{x}^{(i)} = x^{(j(i))}$)
- After resampling, all particles have the same weight

- **Why using resampling?** New generation after resampling allows the filter to focus computation on likely regions of the state space, while suppressing non-meaningful samples!

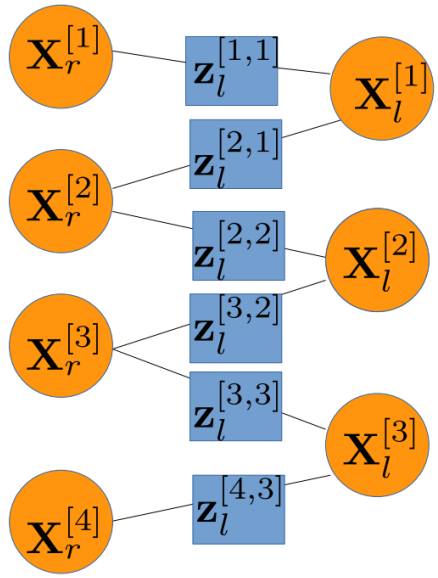
Particle Filter: Localization

- **Prediction**
  1. Generate noise samples ($n_u^{(i)} \sim p(n_u)$)
  2. Transform each sample with its noise from motion model:
     - $x_{t|t-1}^{(i)} = f\left(x_{t-1|t-1}^{(i)},\ u_{t-1} + n_u^{(i)}\right)$
     - $w_{t|t-1}^{(i)} = w_{t-1|t-1}^{(i)}$
- **Update**
  1. Data association for each particle (e.g., greedy approach)
  2. Weights update (multiply by the measurement likelihood):
     - $w_{t|t}^{(i)} = w_{t|t-1}^{(i)} \cdot p\left(z_t | x_{t|t-1}^{(i)}\right)$
     - $x_{t|t}^{(i)} = x_{t|t-1}^{(i)}$ (samples after update remain unchanged!)
- **Resampling with normalization of the weights**

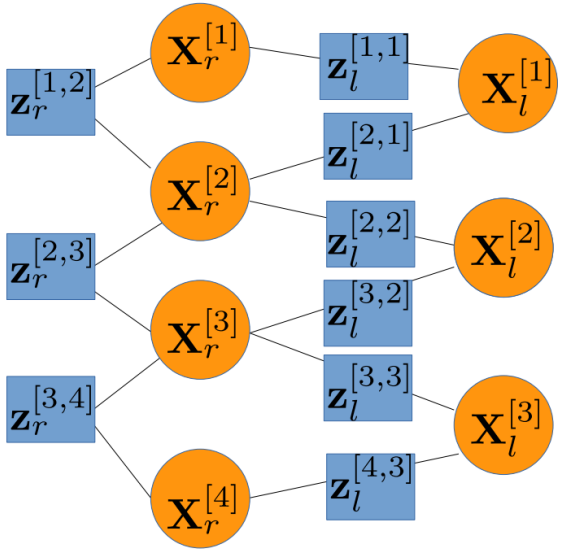


Pose Graph SLAM: Graphical Representation

- Pose-landmark SLAM problem as a graph (node / state variable, node / measurement, edge between a variable and a measurement if they are dependent)



- Landmark + Odometry SLAM as a graph (node / state variable, node / measurement, edge between a variable and a measurement if they are dependent)



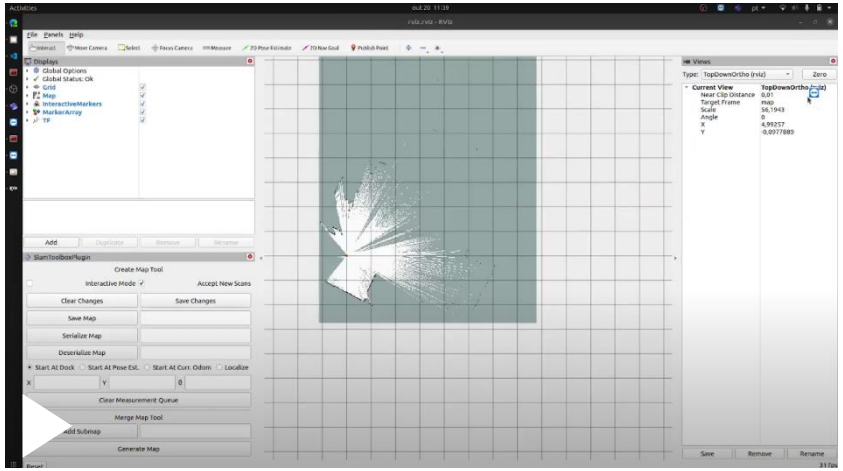
Pose Graph SLAM: Formulation

- State variables : $x = x_{1:n}$
- Log-Likelihood : $F(x) = \sum_i e^{[i]}\left(x^i\right)^T \cdot \Omega^{[i]} \cdot e^{[i]}\left(x^i\right)$
- Graph optimization : $x^* = \underset{x}{\operatorname{argmin}} F(x)$

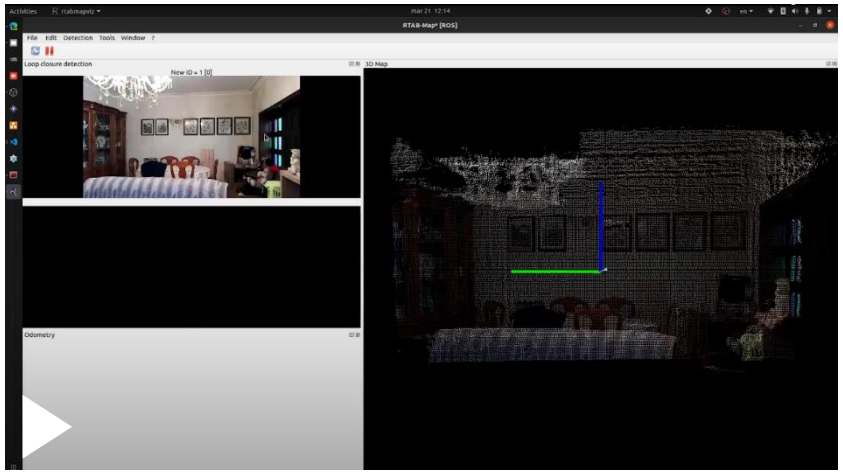
- Example of a Pose-Landmark Constraint:
  - $h_{\text{icp}}^{[i,j]}(X) = {X_r^{[i]}}^{-1} X_l^{[j]} = {R_r^{[i]}}^T \left(X_l^{[j]} - t_r^{[i]}\right)$
  - $e_{\text{icp}}^{[i,j]}\left(X_r^{[i]} \boxplus \Delta x_r^{[i]}, X_l^{[j]} + \Delta x_l^{[j]}\right) = h_{\text{icp}}^{[i,j]}\left(X_r^{[i]} \boxplus \Delta x_r^{[i]}, X_l^{[j]} + \Delta x_l^{[j]}\right) - z_{\text{icp}}^{[i,j]}$

Pose Graph SLAM: Examples

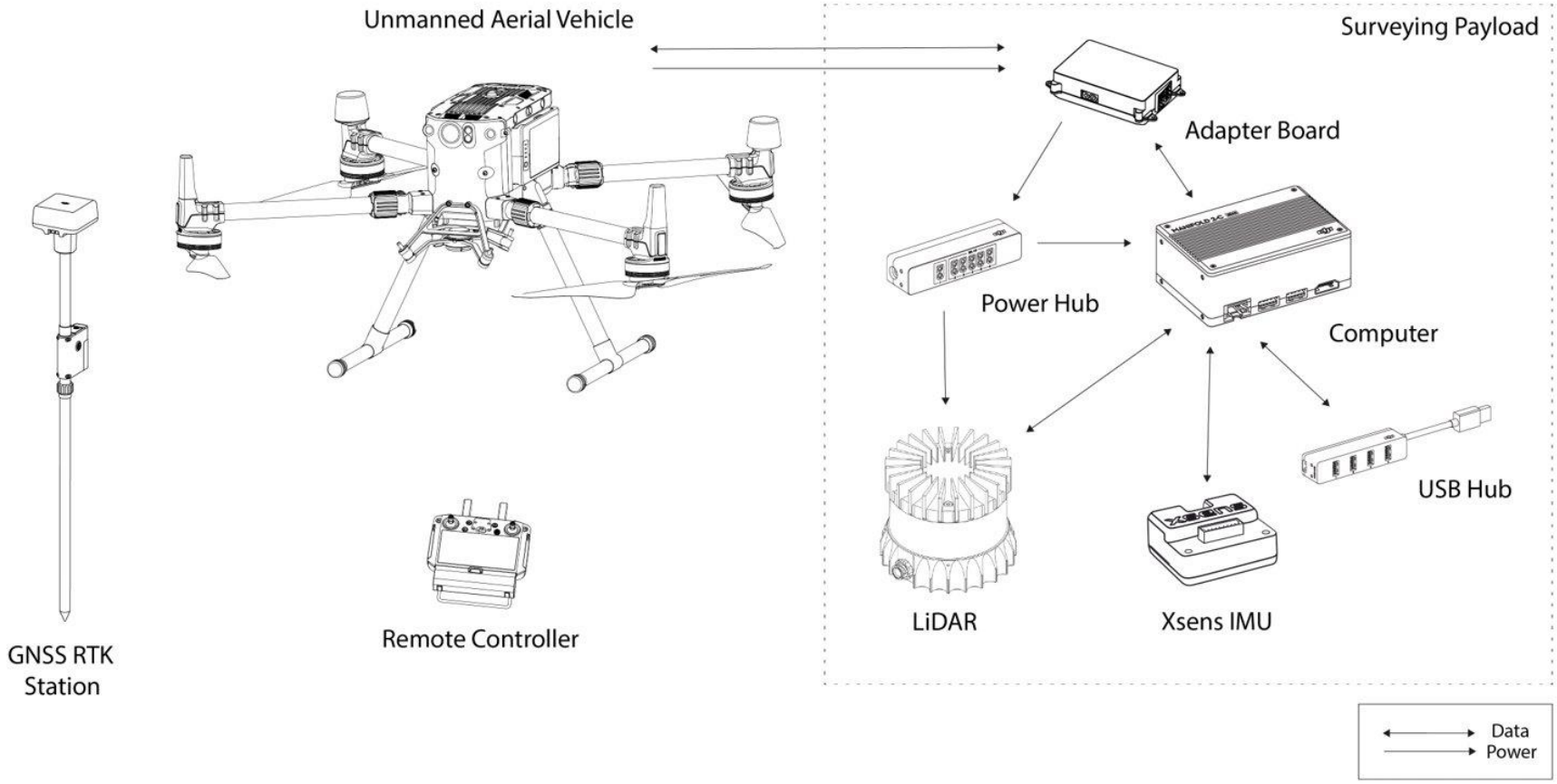
- SLAM Toolbox



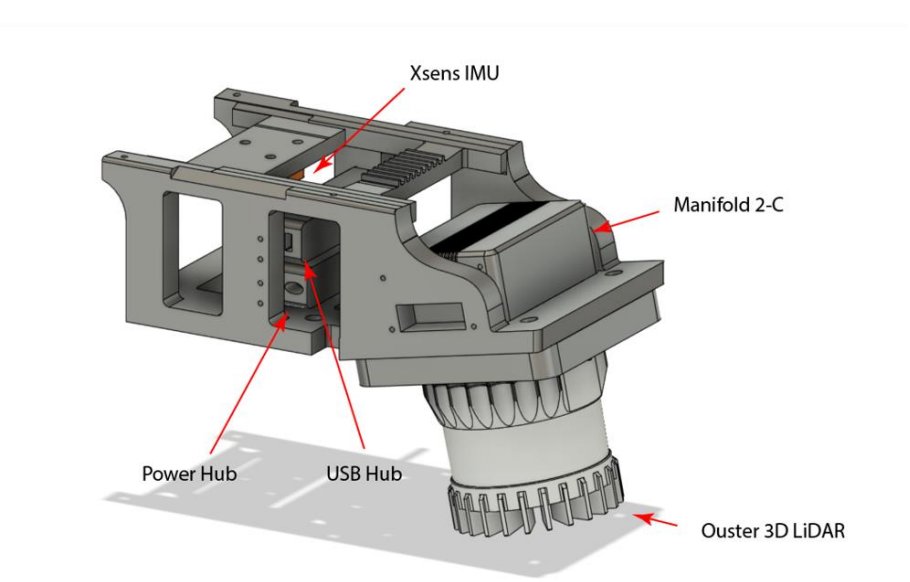
- RTAB-Map



Field Experiments on SLAM with Mobile Robots: Drone Surveying Platform



Field Experiments on SLAM with Mobile Robots: Drone Surveying Platform



INESCTEC

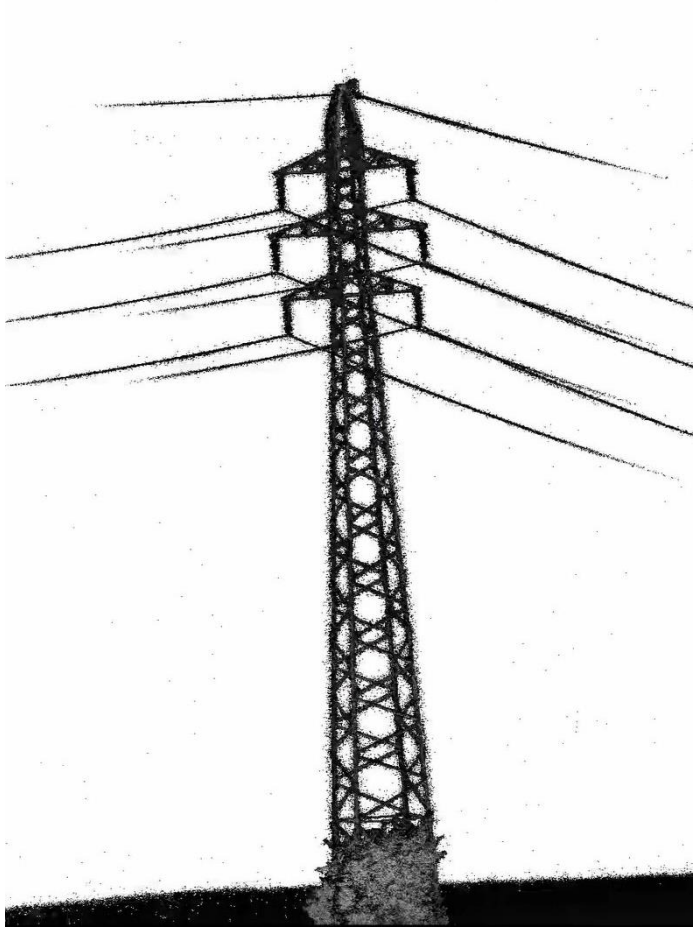
Field Experiments on SLAM with Mobile Robots: Drone Surveying Platform

- **3D mapping through photographic images:**
  - Extract 3D data from multiple 2D images that overlap by triangulating points based on geometric relationships
  - Photogrammetry can reconstruct environments creating coloured 3D models

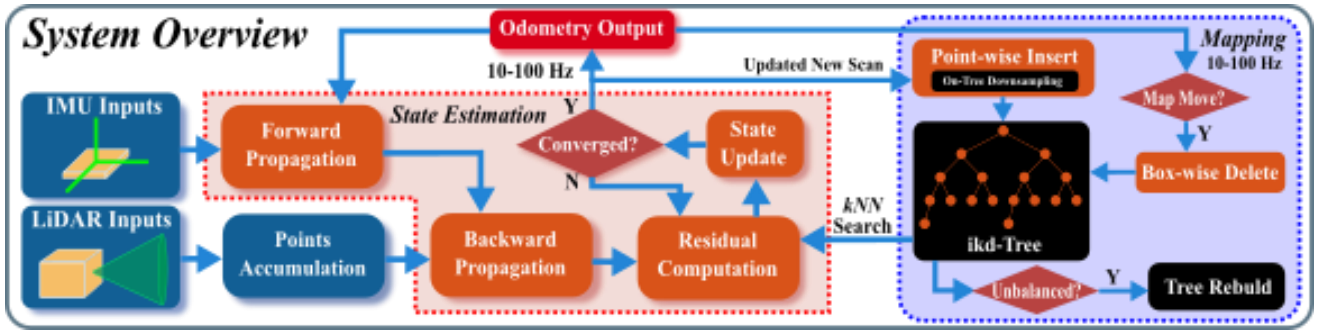
Reconstruction of 3D environment at Quinta do Seixo Vineyard – Coloured Point Cloud
(each point [x,y,z] has a colour assigned)

Field Experiments on SLAM with Mobile Robots: Drone Surveying Platform

- **3D mapping through point cloud registration using the FAST-LIO2 SLAM algorithm:**

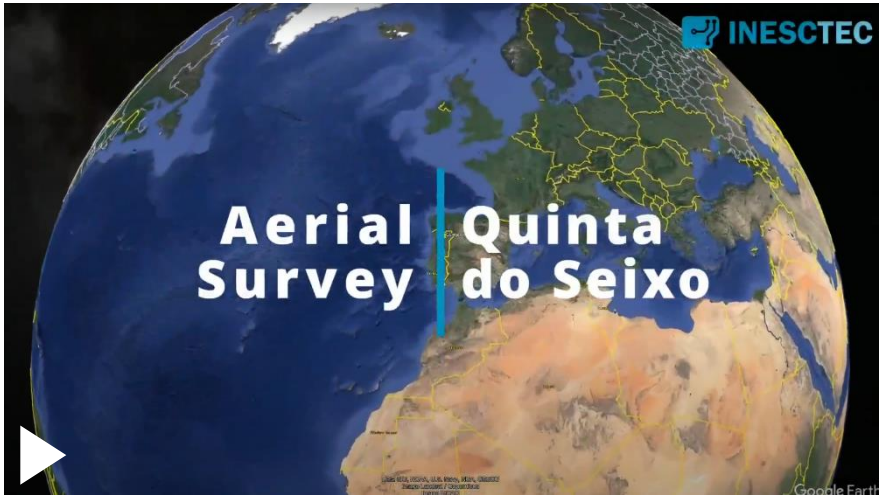


Reconstruction of a Transmission Tower (Point Cloud w/ Intensity as the strength of the laser return, based on reflectivity)



- Direct odometry (scan to map) on raw LiDAR points
- Uses IMU data to correct for in-scan motion error from LiDAR registration (points projected at the scan end-time)
- ikd-Tree mapping achieving faster speeds in mapping
- If map moves, points are deleted

Field Experiments on SLAM with Mobile Robots: Videos



References

[1] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. *Introduction to autonomous mobile robots.* The MIT Press, Cambridge, Massachusetts, Second edition, 2011. ISBN: 978-0-262-01535-6.

[2] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, Massachusetts, First edition, 2005. ISBN: 978-0-262-20162-9.

[3] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. *Advanced Textbooks in Control and Signal Processing*. Springer-Verlag, London, 2009. ISBN: 978-1-84628-641-4. DOI: 10.1007/978-1-84628-642-1

[4] G. Grisetti *et al. Probabilistic Robotics Lectures*. Sapienza University of Rome, Rome, Italy. URL: https://sites.google.com/diag.uniroma1.it/probabilistic-robotics-2023-24

[5] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. *Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age*. IEEE Transactions on Robotics, **32**(6):1309–1332, 2016. DOI: 10.1109/TRO.2016.2624754

[6] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser. *Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving*. IEEE Transactions on Intelligent Vehicles, **2**(3):194–220, 2017. DOI: 10.1109/TIV.2017.2749181

[7] S. Saeedi, M. Trentini, M. Seto, and H. Li. *Multiple-Robot Simultaneous Localization and Mapping: A Review*. Journal of Field Robotics, **33**(1):3–46, 2016. DOI: 10.1002/rob.21620

[8] R. B. Sousa, H. M. Sobreira, and A. P. Moreira. *A systematic literature review on long-term localization and mapping for mobile robots*. Journal of Field Robotics, **40**(5):1245–1322, 2023. DOI: 10.1002/rob.22170

[9] H. Durrant-Whyte and T. Bailey. *Simultaneous localization and mapping: part I*. IEEE Robotics & Automation Magazine, **13**(2)99–110, 2006. DOI: 10.1109/MRA.2006.1638022

[10] T. Bailey and H. Durrant-Whyte *Simultaneous localization and mapping (SLAM): part II*. IEEE Robotics & Automation Magazine, **13**(3):108–117, 2006. DOI: 10.1109/MRA.2006.1678144

References

[11] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. *A Tutorial on Graph-Based SLAM*. IEEE Intelligent Transportation Systems Magazine, **2**(4)31–43, 2010. DOI: 10.1109/MITS.2010.939925

[12] D. Scaramuzza and F. Fraundorfer. *Visual Odometry [Tutorial]*. IEEE Robotics & Automation Magazine, **18**(4):80–92, 2011, DOI: 10.1109/MRA.2011.943233

[13] F. Fraundorfer and D. Scaramuzza. *Visual Odometry: Part II: Matching, Robustness, Optimization, and Applications*. IEEE Robotics & Automation Magazine, **19**(2):78–90, 2012. DOI: 10.1109/MRA.2012.2182810

[14] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang. *FAST-LIO2: Fast Direct LiDAR-Inertial Odometry*. IEEE Transactions on Robotics, **38**(4):2053–2073, 2022. DOI: 10.1109/TRO.2022.3141876. GitHub: https://github.com/hku-mars/FAST_LIO

References: SLAM Algorithms (most known / used)

[A] GMapping — G. Grisetti, C. Stachniss, and W. Burgard. *Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters*. IEEE Transactions on Robotics, **23**(1):34–46, 2007. DOI: 10.1109/TRO.2006.889486. [GitHub] [ROS]

[B] HectorSLAM — S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf. *A flexible and scalable SLAM system with full 3D motion estimation*. In: 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics, pp. 155–160, 2011. DOI: 10.1109/SSRR.2011.6106777. [GitHub] [ROS]

[C] RTAB-Map — M. Labbé and François Michaud. *RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation*. Journal of Field Robotics, **36**(2):416–446, 2018. DOI: 10.1002/rob.21831. [GitHub] [ROS]

[D] SLAM Toolbox — S. Macenski and I. Jambrecic. *SLAM Toolbox: SLAM for the dynamic world*. The Journal of Open Source Software, **6**(61):2783, 2021. DOI: 10.21105/joss.02783. [GitHub] [ROS]

[E] LOAM — J. Zhang and S. Singh. *LOAM: Lidar Odometry and Mapping in Real-time*. In: Robotics: Science and Systems Conference (RSS). Berkeley, CA, July 2014. URL: https://www.roboticsproceedings.org/rss10/p07.pdf. [GitHub] [ROS]

[F] A-LOAM — N/A. [GitHub]

[G] LeGO-LOAM — T. Shan and B. Englot. *LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain*. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4758–4765, 2018. DOI: 10.1109/IROS.2018.8594299. [GitHub]

[H] ORB-SLAM — R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. *ORB-SLAM: A Versatile and Accurate Monocular SLAM System*. IEEE Transactions on Robotics, **31**(5):1147–1163, 2015. DOI: 10.1109/TRO.2015.2463671. [GitHub]

[I] Kimera-VIO — A. Rosinol, M. Abate, Y. Chang, and L. Carlone. *Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping*. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 1689–1696, 2020, doi: 10.1109/ICRA40945.2020.9196885 [GitHub] [ROS]

See https://silenceoverflow.github.io/Awesome-SLAM/, https://openslam-org.github.io/ , and https://github.com/tzutalin/awesome-visual-slam (or even literature reviews on the topic such as the ones referred in the previous slide) for more SLAM algorithms if you are interested in the topic.