ROBOT INTELLIGENCE LAB

Simultaneous Localization And Mapping

Sensor-based SLAM

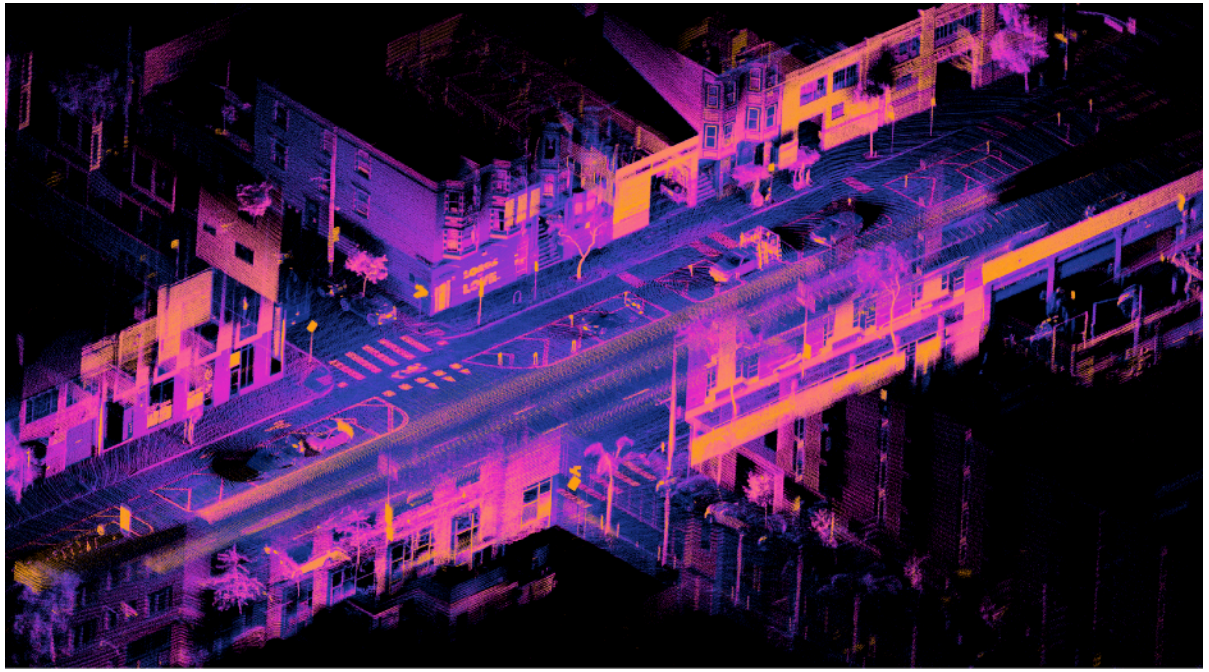
Sungjoon Choi, Korea University

kakaobrain

SLAM

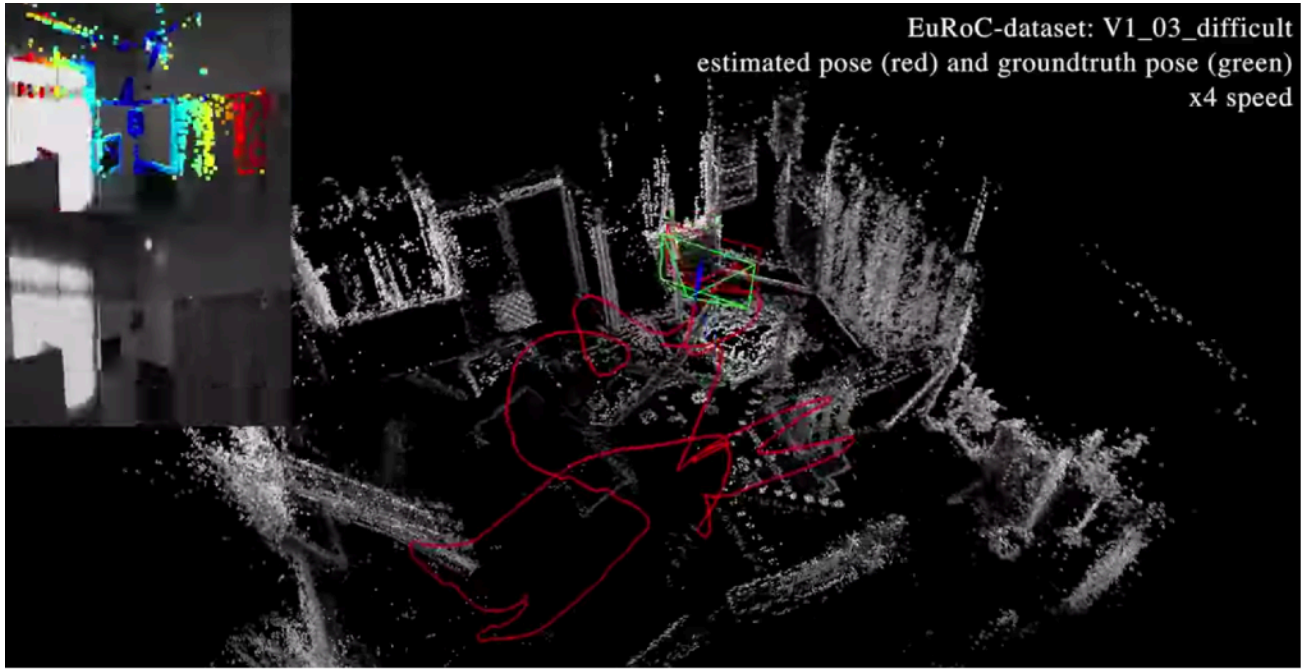
ROBOT INTELLIGENCE LAB

LiDar-based SLAM



https://en.wikipedia.org/wiki/File:Ouster_OS1-64_lidar_point_cloud_of_intersection_of_Folsom_and_Dore_St,_San_Francisco.png

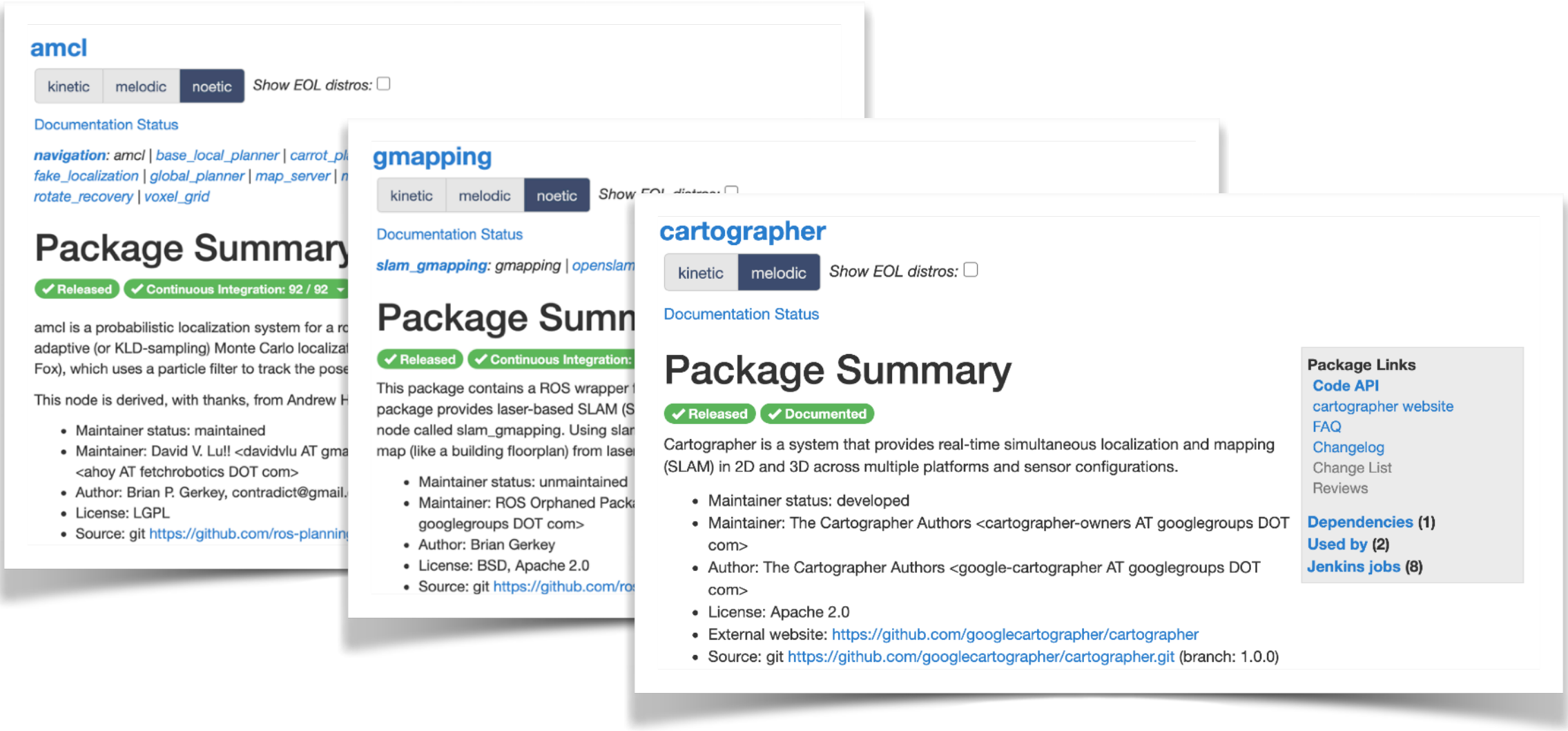
Visual Inertia Odometry



https://youtu.be/GoqnXDS7jbA

Introduction

ROBOT INTELLIGENCE LAB



ROBOT INTELLIGENCE LAB

KLD-Sampling

"KLD-Sampling: Adaptive Particle Filters," 1998

KLD-Sampling

ROBOT INTELLIGENCE LAB

- "KLD-Sampling: Adaptive Particle Filters," 1998
  - It is for **localization**.
  - It presents a statistical approach to increasing the efficiency of **particle filters** by adapting the size of sample sets on-the-fly.
  - The key idea of the KLD-sampling method is to bound the approximate error, measured by the Kullback-Leibler distance, introduced by the sample-based representation of the particle filter.
  - KLD-sampling chooses a small number of samples if the density is focused on a small part of the state space and a large number of samples if the state uncertainty is high.

KLD-Sampling

ROBOT INTELLIGENCE LAB

- This is the method behind the amcl package in ROS.

amcl

kinetic | melodic | noetic | *Show EOL distros:* ☐

Documentation Status

navigation: *amcl | base_local_planner | carrot_planner | clear_costmap_recovery | costmap_2d | dwa_local_planner | fake_localization | global_planner | map_server | move_base | move_base_msgs | move_slow_and_clear | nav_core | navfn | rotate_recovery | voxel_grid*

Package Summary

✔ Released ✔ Continuous Integration: 92 / 92 ✔ Documented

amcl is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses a particle filter to track the pose of a robot against a known map.

This node is derived, with thanks, from Andrew Howard's excellent 'amcl' Player driver.

Package Links

- **Code API**
- Tutorials
- FAQ
- Changelog
- Change List
- Reviews

Dependencies (14)
Used by (5)
Jenkins jobs (10)

- Maintainer status: maintained
- Maintainer: David V. Lu!! <davidvlu AT gmail DOT com>, Michael Ferguson <mfergs7 AT gmail DOT com>, Aaron Hoy <ahoy AT fetchrobotics DOT com>
- Author: Brian P. Gerkey, contradict@gmail.com
- License: LGPL
- Source: git https://github.com/ros-planning/navigation.git (branch: noetic-devel)

Particle Filter

- **Particle filter (PF)** is a method used to solve **filtering** problems such as estimating the internal states in dynamical systems when partial observations are made.
  - For example, we want to estimate the location of the sensor given the sensory observation and a map (i.e., **localization** problems).
- PF represents **belief** (of the random variable) using samples.
- PF can use actual (nonlinear) dynamics and (complicated) measurement models.
- It is a sampling-based method, hence the performance depends on the number of samples.

Particle Filter

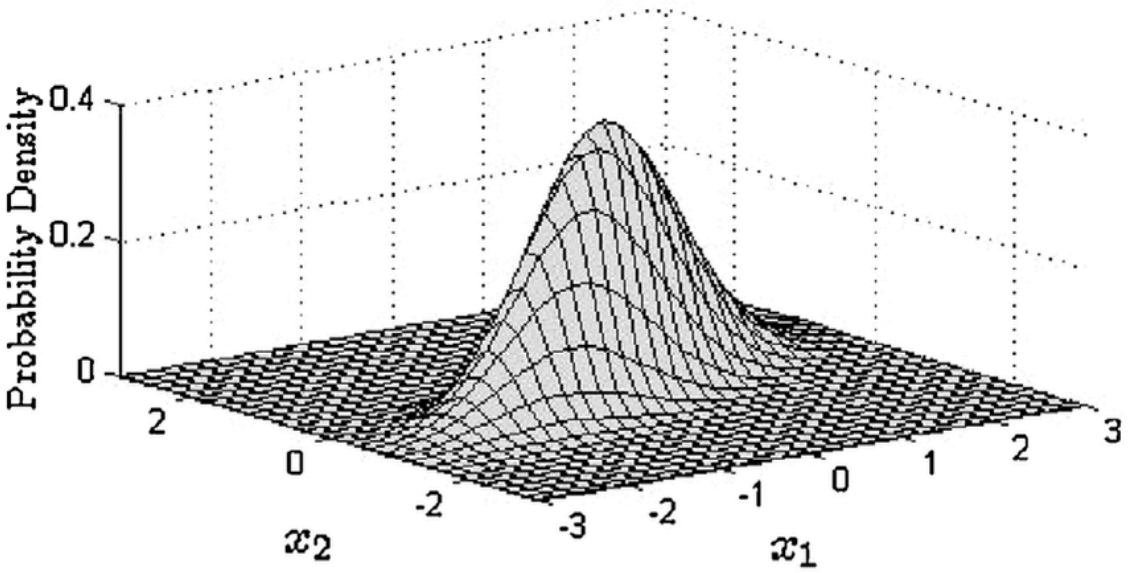
ROBOT INTELLIGENCE LAB

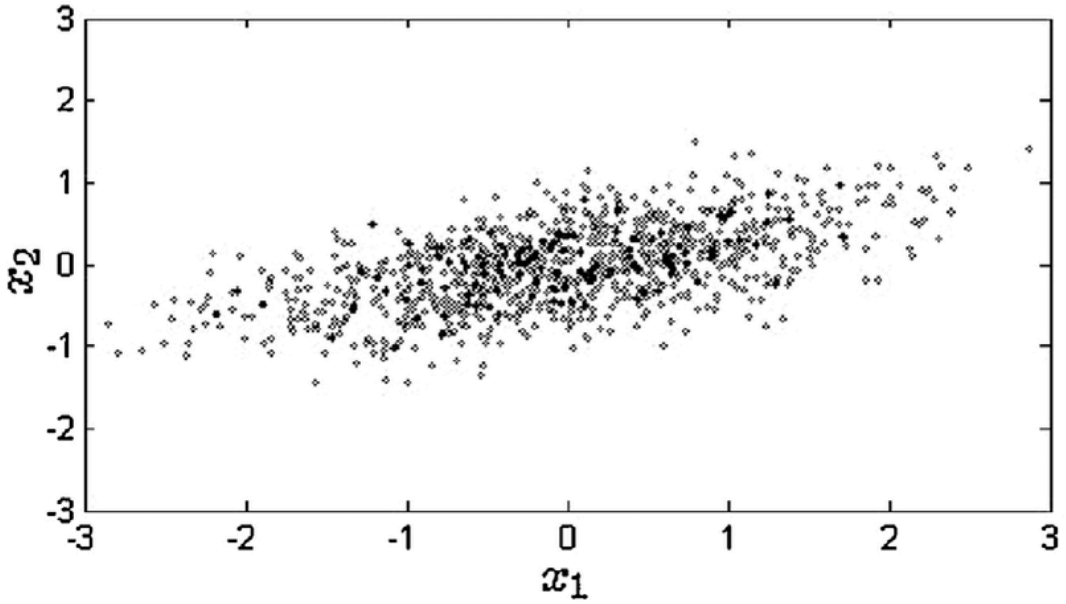
- **Problem to be Solved**
  - Let X_t be the internal state and Z_t be the observation at time t. Then the goal of a particle filter is to estimate:
    - Given a sample-based representation $S_t = \{x_t^1, \ldots, x_t^N\}$ of $\text{Bel}(X_t) = P(X_t \mid Z_1, \ldots, Z_t, u_1, \ldots, u_t)$ at time t,
    - Find a sample-based representation $S_{t+1} = \{x_{t+1}^1, \ldots, x_{t+1}^N\}$ of $\text{Bel}(X_{t+1}) = P(X_{t+1} \mid Z_1, \ldots, Z_{t+1}, u_1, \ldots, u_{t+1})$ of the next time $t+1$.

Particle Filter

ROBOT INTELLIGENCE LAB

- Sample-based representation?





Particle Filter

ROBOT INTELLIGENCE LAB

- **Sequential Importance Sampling (SIS) Particle Filter**
  - Initialize particles $\{x_1^i\}_{i=1}^N$ and weights $\{w_1^i\}_{i=1}^N$
  - For $t = 1,2,\ldots$
    - Dynamics update:
      - For $i = 1,2,\ldots,N$:
        - Sample $x_{t=1}^i \sim P(X_{t+1} | X_t = x_t^i, u_t + 1)$
    - Observation update:
      - For $i = 1,2,\ldots,N$:
        - $w_{t+1}^i = w_t^i \times P(Z_{t+1} | X_{t+1} = x_{t+1}^i)$
  - At any time t, the distribution $P(X_t)$ is represented by the weighted set of samples $\{(x_t^i, w_t^i)\}_{i=1}^N$.

Particle Filter

ROBOT INTELLIGENCE LAB

- **Sequential Importance Sampling (SIS) Particle Filter**
  - Initialize particles $\{x_1^i\}_{i=1}^N$ and weights $\{w_1^i\}_{i=1}^N$
  - For $t = 1,2,\ldots$
    - Dynamics update:
      - For $i = 1,2,\ldots,N$:
        - Sample $x_{t=1}^i \sim P(X_{t+1} | X_t = x_t^i, u_t + 1)$

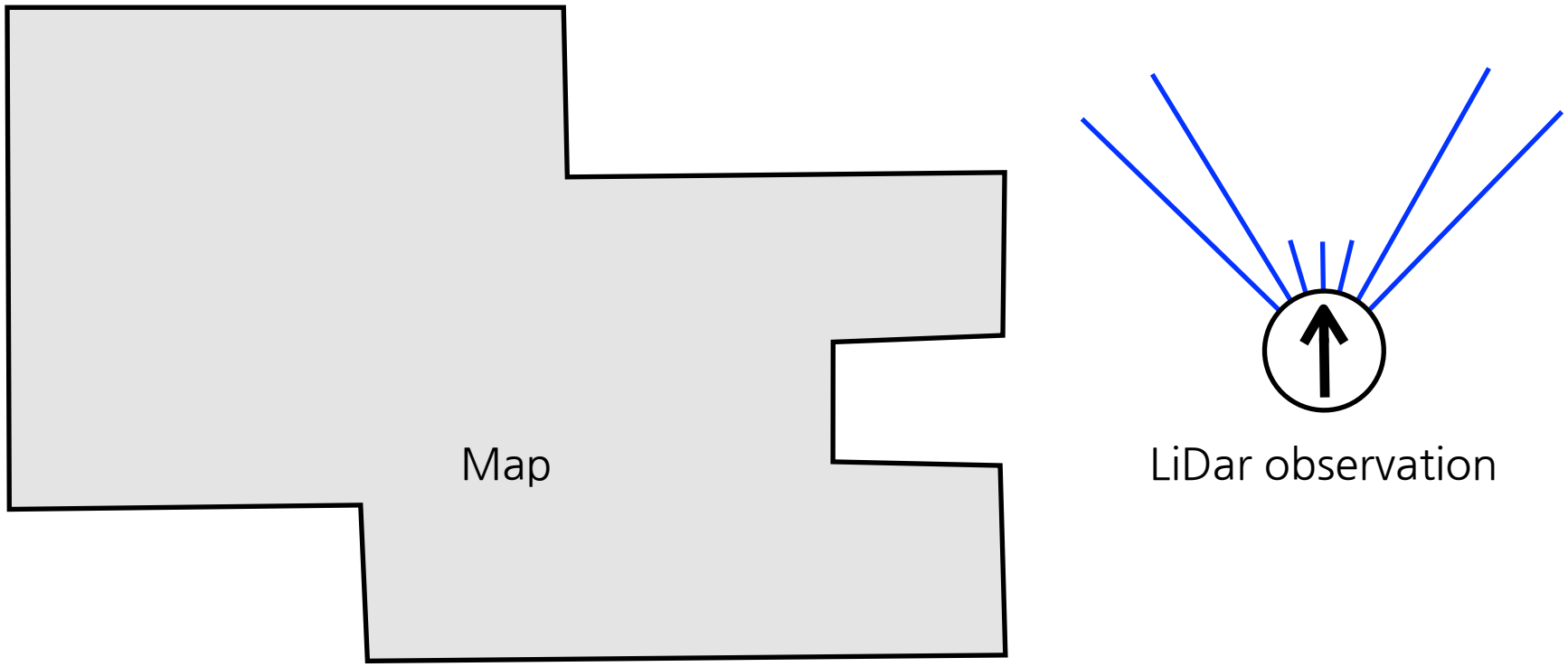
        We can use the **nonlinear dynamic model** without worrying about the integration!
    - Observation update:
      - For $i = 1,2,\ldots,N$:
        - $w_{t+1}^i = w_t^i \times P(Z_{t+1} | X_{t+1} = x_{t+1}^i)$

        We can use **complicated observation model** (likelihood) without worrying about the conjugacy.
  - At any time t, the distribution $P(X_t)$ is represented by the weighted set of samples $\{(x_t^i, w_t^i)\}_{i=1}^N$.

Particle Filter

ROBOT INTELLIGENCE LAB

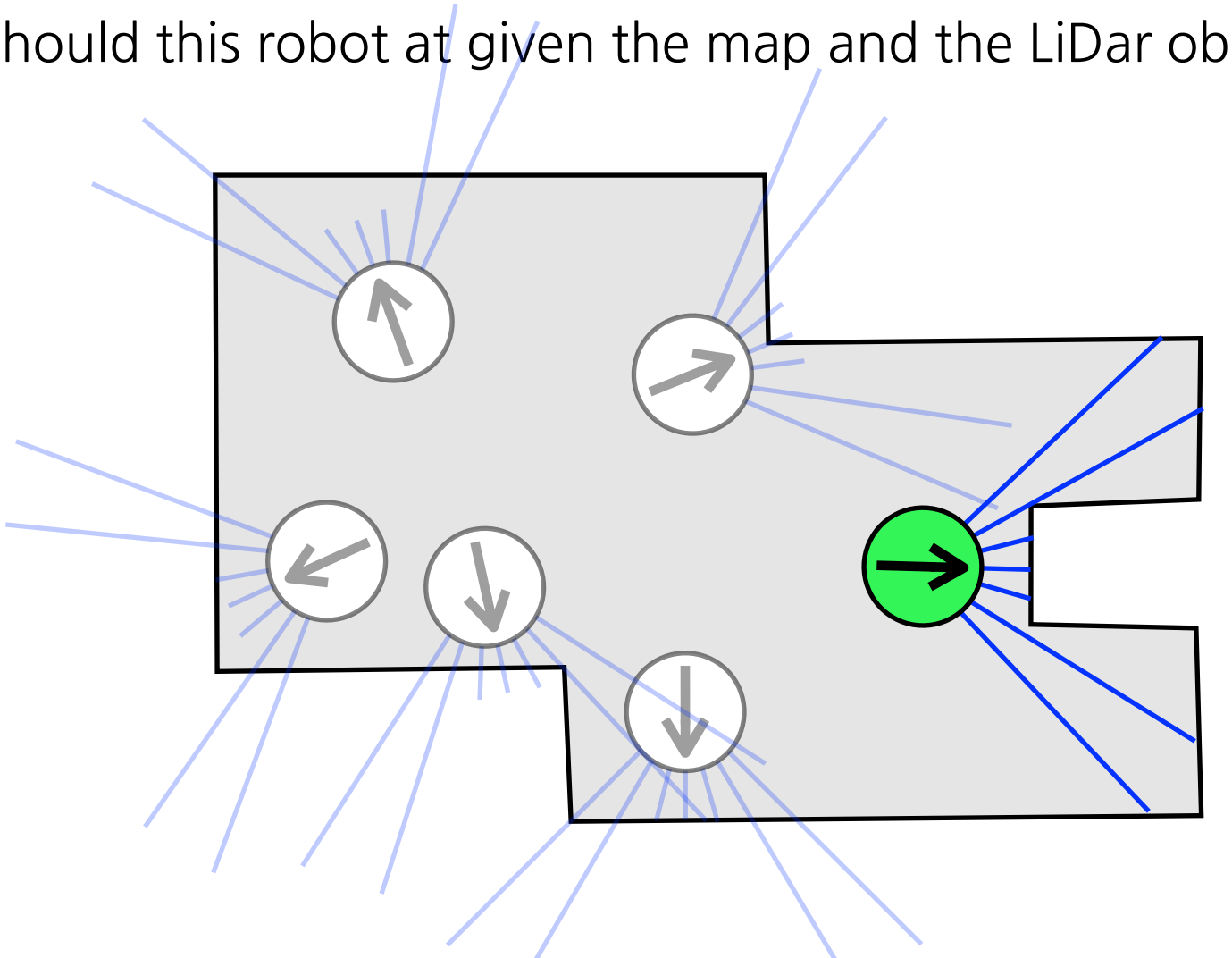
- Observation model
  - Where should this robot at given the map and the LiDar observation?



Particle Filter

ROBOT INTELLIGENCE LAB

- Observation model
  - Where should this robot at given the map and the LiDar observation?



Particle Filter

ROBOT INTELLIGENCE LAB

- The state distribution $P(X_t)$ is solely defined by a set of particles.
- The particles are only weighted by the evidence (likelihood).
- In other words, the location of the samples themselves are **never affected** by the evidence.
  - Hence, it often **fails to concentrate** particles in the high probability regions of the posterior distribution $P(X_t | Z_1, \ldots, Z_t)$.
- To mitigate this issue, **resampling** comes in to rescue.

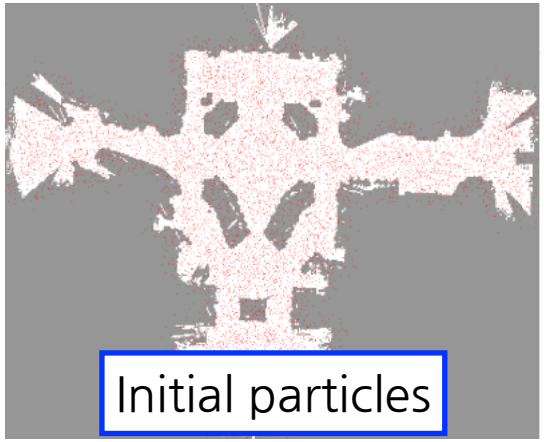
Particle Filter

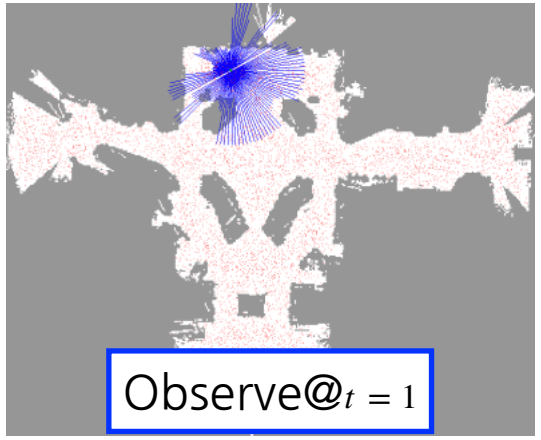
ROBOT INTELLIGENCE LAB

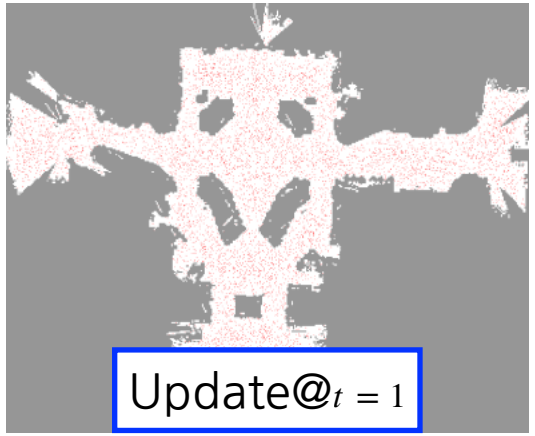
- **Sequential Importance Resampling (SIR)**
  - Any any time t, the distribution is represented by the weighted set of N samples, i.e., $\{(x_t^i, w_t^i)\}_{i=1}^N$.
  - We simply **re-sample** N times from the existing set of particles.
  - The probability of drawing each particle is given by its importance weight w_t^i.
- **SIR** enables more particles to **focus** on the parts of the state space with high probability mass by **resampling**.

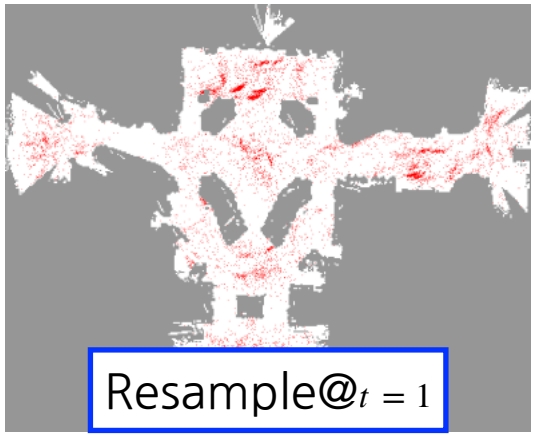
Particle Filter

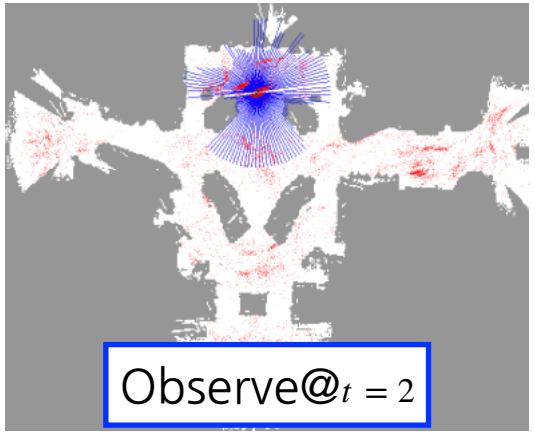
ROBOT INTELLIGENCE LAB

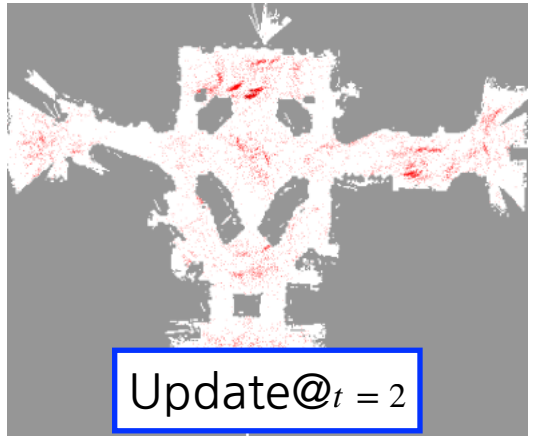


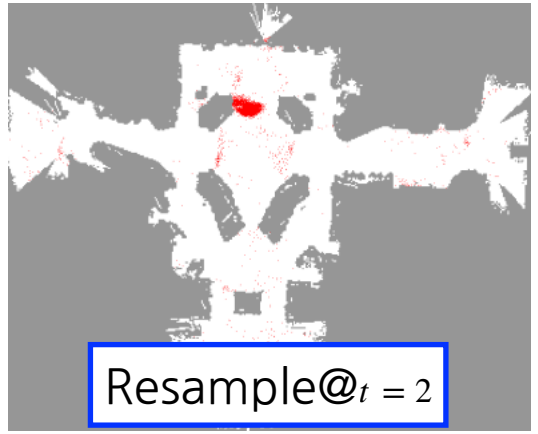












Particle Filter

ROBOT INTELLIGENCE LAB

- Little more on **importance sampling**

f can be **whether a grid cell is occupied or not**, etc.

$$
\begin{aligned}
\mathrm{E}_{X\sim p}[f(X)] &= \int_x f(x)p(x)dx \\
&\qquad\qquad \pi \text{ is a } \textbf{proposal} \text{ distribution.} \\
&= \int_x f(x)p(x)\frac{\pi(x)}{\pi(x)}dx \quad \text{if} \pi(x)=0 \Rightarrow p(x)=0 \\
&= \int_x f(x)\frac{p(x)}{\pi(x)}\pi(x)dx \\
&= \mathrm{E}_{X\sim\pi}[\frac{p(X)}{\pi(X)}f(X)] \\
&\approx \frac{1}{m}\sum_{i=1}^{m}\frac{p(x^{(i)})}{\pi(x^{(i)})}f(x^{(i)}) \quad \text{with } x^{(i)}\sim\pi
\end{aligned}
$$

Particle Filter

ROBOT INTELLIGENCE LAB

- Little more on **resampling**
  - Consider running a particle filter for localization with **long corridors**.

  - While no information is obtained that favors one particle over another, due to **resampling** some particles will disappear and after running sufficiently long with very high probability all particles will have become **identical**.
  - Resampling induces the loss of **diversity**. The variance of particles decreases and the variance of the set **as an estimator** of the true belief increase.

Particle Filter

ROBOT INTELLIGENCE LAB

- Some solutions to the drawbacks of **resampling**
  1. Resample only when the **effective sampling size** is low where the **effective sampling size** is proportional to the entropy of the normalized weights $\{w_t^i\}_{i=1}^N$ of the particles (e.g., if all weights are close to $1/N$, do not resample).
  2. Instead of resampling from the discrete distribution, sample from the **continuous density** (e.g., add a little perturbation to the particle x_t^i while doing resampling).

Particle Filter

ROBOT INTELLIGENCE LAB

- **Sensor noise level**
  - What happens if we assume the **perfect** sensory model?
  - Then, most of the likelihood will be zero leading to unstable filtering performances (**why?**).

- **Number of particles**
  - The number of particles is an important hyper-parameter to tune.
  - Typically more particles are needed at the **beginning** of localization run (**why?**).
  - **Adaptive** number particles?

KLD-Sampling

- Finally, back to **KLD-sampling**.
- **Key Ideas of KLD-sampling**
  - Adaptive number of particles
  - Partition the state-space.
  - When sampling, keep track of the number of bins occupied.
  - Stop sampling when a **threshold** that depends on the number of occupied bins (as well as the guarantee level δ) is reached.
    - It chooses a small number of samples if the density is focused on a small subspace, and chooses a large number of samples if the samples have to cover a major part of the state space.

KLD-Sampling

ROBOT INTELLIGENCE LAB

Inputs: $S_{t-1} = \{\langle x_{t-1}^{(i)}, w_{t-1}^{(i)} \rangle \mid i = 1, \ldots, n\}$ representing belief $Bel(x_{t-1})$, control measurement u_{t-1}, observation z_t, bounds ε and δ, bin size Δ

$S_t := \emptyset$, $n = 0$, $k = 0$, $\alpha = 0$ /* *Initialize* */

do /* *Generate samples . . .* */

Sample an index $j(n)$ from the discrete distribution given by the weights in S_{t-1}

Sample $x_t^{(n)}$ from $p(x_t \mid x_{t-1}, u_{t-1})$ using $x_{t-1}^{(j(n))}$ and u_{t-1}

$w_t^{(n)} := p(z_t \mid x_t^{(n)})$; /* *Compute importance weight* */

$\alpha := \alpha + w_t^{(n)}$ /* *Update normalization factor* */

$S_t := S_t \cup \{\langle x_t^{(n)}, w_t^{(n)} \rangle\}$ /* *Insert sample into sample set* */

if ($x_t^{(n)}$ falls into empty bin b) **then** /* *Update number of bins with support* */

$k := k + 1$

$b :=$ non-empty

$n := n + 1$ /* *Update number of generated samples* */

while ($n < \frac{1}{2\epsilon}\chi^2_{k-1,1-\delta}$) /* *. . . until K-L bound is reached* */

for $i := 1, \ldots, n$ **do** /* *Normalize importance weights* */

$w_t^{(i)} := w_t^{(i)} / \alpha$

return S_t

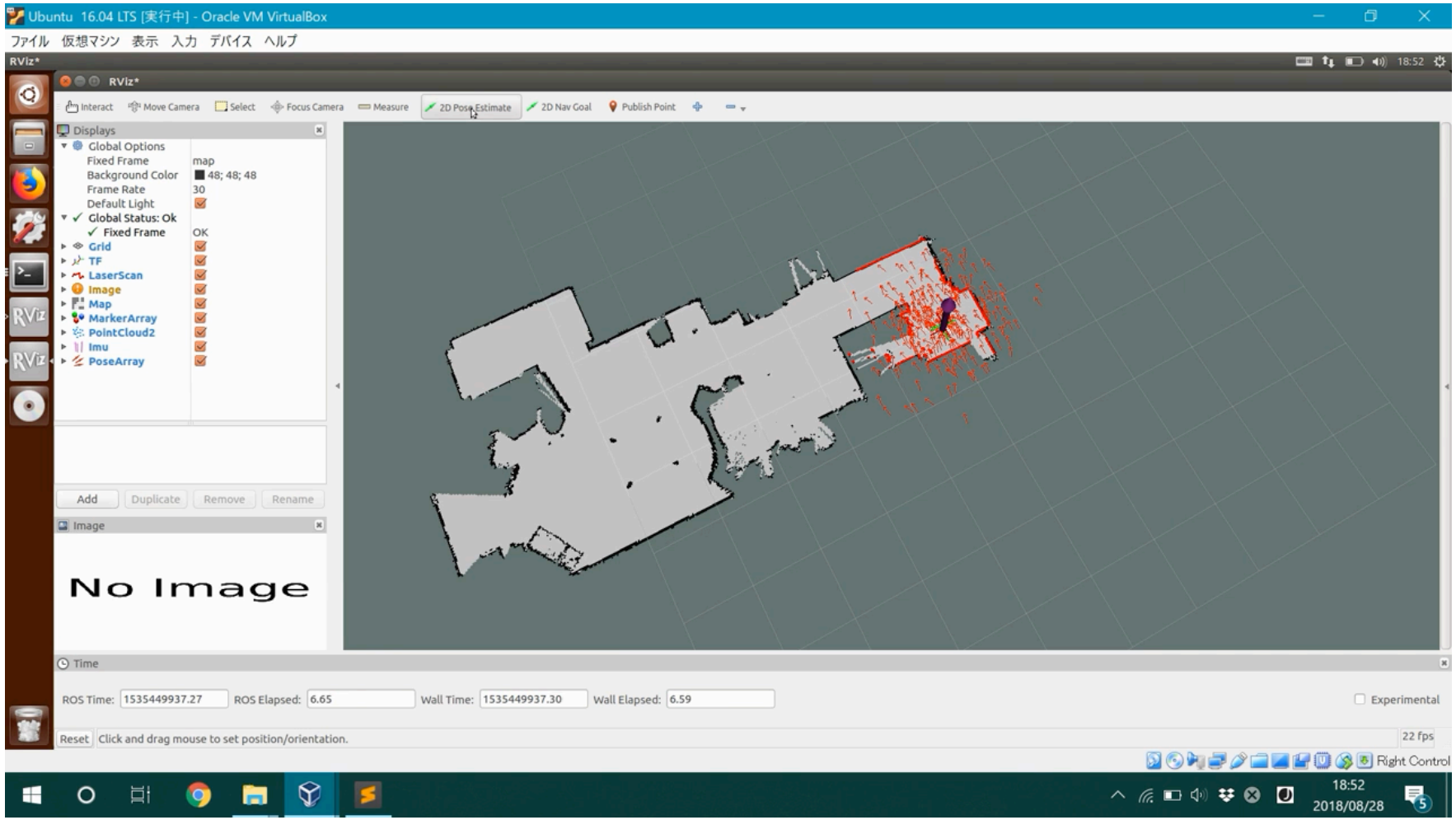
Table 1: KLD-sampling algorithm.

Ordinary Particle Filter

Adaptive number of Particles

AMCL

ROBOT INTELLIGENCE LAB



https://youtu.be/_sN_Yho9Ck8

ROBOT INTELLIGENCE LAB

Monte Carlo Localization

"Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," 1999

Monte Carlo Localization

- "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," 1999
  - **Monte Carlo Localization (MCL)** is a sample-based algorithm for mobile robot localization based on a particle filter with adaptive sampling scheme similar to KLD-sampling.
  - **MCL** and **PF with KLD-sampling** seem almost identical to me (the only difference is on how the number of particles are adaptively selected).

ROBOT INTELLIGENCE LAB

Rao-Blackwellized Particle Filter

"Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters," 2007

Rao-Blackwellized Particle Filter

- "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters," 2007
  - This paper presents Rao-Blackwellized particle filters (**RBPF**) as an effective means to solve a SLAM problem in which each particle carries an individual map of the environment.
  - An accurate proposal distribution is presented that not only takes the movement of the robot but also consider the most recent observation.

Rao-Blackwellized Particle Filter

ROBOT INTELLIGENCE LAB

- This is the method behind the **gmapping** package in ROS.

gmapping

kinetic | melodic | noetic *Show EOL distros:* ☐

Documentation Status

***slam_gmapping**: gmapping | openslam_gmapping*

Package Summary

✔ Released ✔ Continuous Integration: 18 / 18 ✔ Documented

This package contains a ROS wrapper for OpenSlam's Gmapping. The gmapping package provides laser-based SLAM (Simultaneous Localization and Mapping), as a ROS node called slam_gmapping. Using slam_gmapping, you can create a 2-D occupancy grid map (like a building floorplan) from laser and pose data collected by a mobile robot.

- Maintainer status: unmaintained
- Maintainer: ROS Orphaned Package Maintainers <ros-orphaned-packages AT googlegroups DOT com>
- Author: Brian Gerkey
- License: BSD, Apache 2.0
- Source: git https://github.com/ros-perception/slam_gmapping.git (branch: melodic-devel)

Package Links

- **Code API**
- Tutorials
- Troubleshooting
- FAQ
- Changelog
- Change List
- Reviews

Dependencies (7)

Used by (1)

Jenkins jobs (10)

gmapping

ROBOT INTELLIGENCE LAB

- **Problem formulation**
  - **Given**
    - observations $z_{1:t} = z_1, \ldots, z_t$
    - odometry measurements $u_{2:t} = u_2, \ldots, u_t$
  - **Find**
    - Posterior $p(x_{1:t}, M \mid z_{1:t}, u_{2:t})$ of state x and a grid map M.

gmapping

ROBOT INTELLIGENCE LAB

- **Key Ideas of gmapping**
  - Rao-Blackwellized Particle Filter (RBPF)
    - In RBPF, each particle contains:
      - samples of the history of robot poses (states) + posterior over maps given the sample pose history
  - **Proposal distribution π**
    - Approximate the optimal sequential proposal distribution

$$p^*(x_t) = p(x_t \mid x^i_{1:t-1}, z_{1:t}, u_{1:t}) \propto p(z_t \mid m^i_{t-1}, x_t) p(x_t \mid x^i_{t-1}, u_t)$$

1. Find the local optimum $\arg\max_x p^*(x)$
2. Sample x^k around to local optimum with weights $w^k = p^*(x^k)$ and fit a Gaussian over the weighted samples
3. This Gaussian is an approximation of the **optimal sequential proposal** distribution.

gmapping

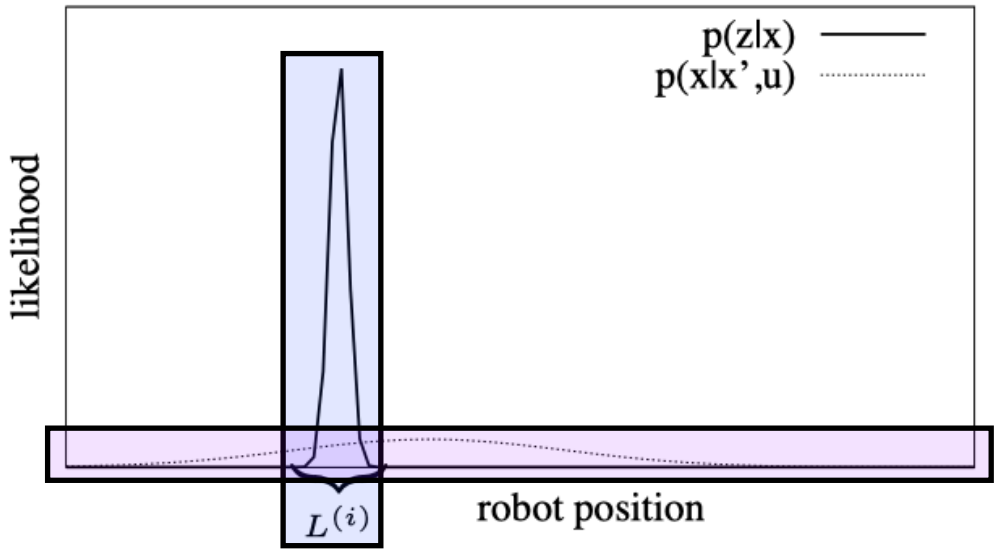
ROBOT INTELLIGENCE LAB

- **Mapping with RBPF**
  - RBPF for SLAM makes use of the following factorization (Rao-Blackwellization)

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = \underbrace{p(m \mid x_{1:t}, z_{1:t})}_{\text{Then, compute map}} \cdot \underbrace{p(x_{1:t} \mid z_{1:t}, u_{1:t-1})}_{\text{Estimate only the robot trajectory first}}.$$

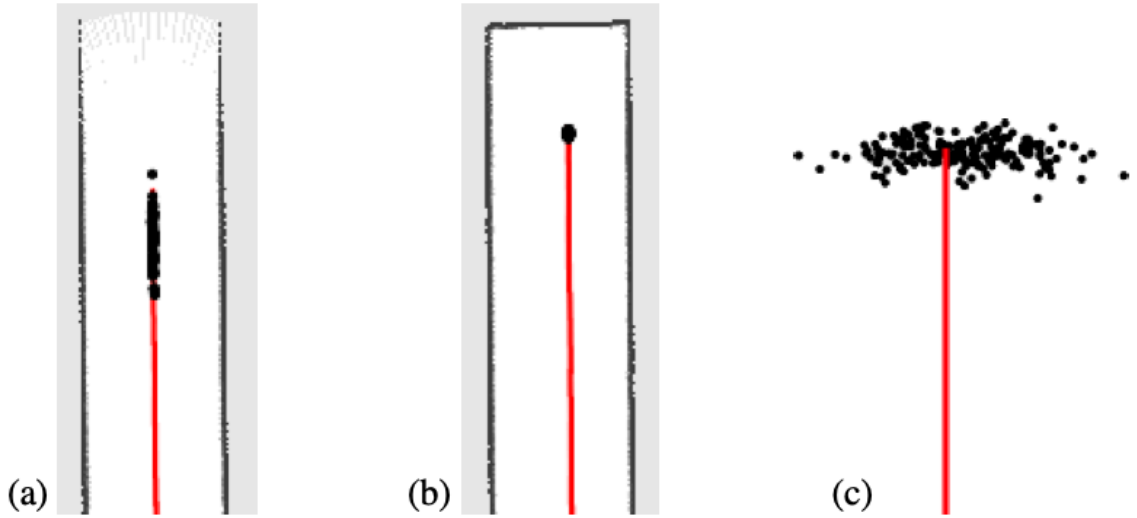
  - The **mapping** part is easy given the trajectory $x_{1:t}$ and observations $z_{1:t}$.
  - To estimate the trajectory from the **posterior** $p(x_{1:t} \mid z_{1:t}, u_{1:t-1})$, the sampling importance resampling (SIR) particle filter is used.
  - Sampling -> Importance Weighting -> Resampling -> Map Estimation

gmapping



- In many cases, sensor information is **significantly more precise** than the **motion estimate** of the robot based on the odometry.
- A common approach is to use a **smoothed likelihood** function, but it often leads to less accurate maps.
- To overcome this, one can consider the most recent sensor observation z_t when generating the **next generation of samples** $\pi(x)$.

gmapping



- Three typical **particle distributions**:
  - (a) In an **open** corridor, the particles distribute along the corridor.
  - (b) Tn a dead end corridor, the uncertainty is small.
  - (c) The raw odometry motion model (without observation) leads less peaked posteriors.

gmapping

ROBOT INTELLIGENCE LAB

- **Algorithm summary**
  1. Make an initial guess on i-th particle $x_t^{\prime(i)} = x_{t-1}^{(i)} \oplus u_{t-1}$ where $\oplus$ corresponds to a dynamic model.
  2. A **scan matching** algorithm is executed based on the map $m_{t-1}^{(i)}$ starting from the initial guess.
  3. A set of sampling points is selected from the **scan-matcher** and a Gaussian is fitted (improved proposal, aka RBPF).
  4. The new pose $x_t^{(i)}$ of particle i is drawn from the Gaussian.
  5. Update the importance weights of each pose based on observations.
  6. The map $m^{(i)}$ of particle i is updated from to the pose $x_t^{(i)}$ and the observation z_t.

gmapping

ROBOT INTELLIGENCE LAB

- **Motion Model**
  - Gaussian (EKF) approximation of the odometry model is used.

- **Scan-Matching**
  - The goal is to find $\arg\max_{x_t} p(z_t \mid m_{t-1}^i, x_t) p(x_t \mid x_{t-1}^i, u_t)$
    - $p(x_t \mid x_{t-1}^i, u_t)$: Gaussian approximation of motion model
    - $p(z_t \mid m_{t-1}^i, x_t)$: Likelihood from scan-match. (gmapping uses "beam endpoint model=likelihood field)

Scan Matching

ROBOT INTELLIGENCE LAB

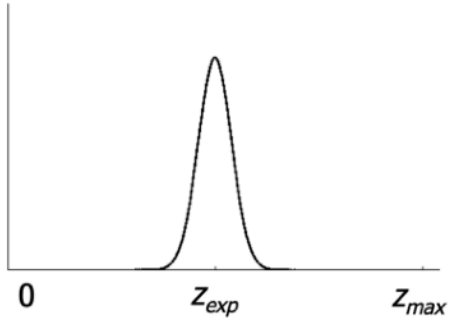
- **Problem statement**
  - Given a (laser) scan and a map, or a scan and a scan, or a map and a map, find the **rigid-body transformation** ($SE(3)$) that aligns them best.
- **Benefits**
  - Improved proposal distribution for SLAM (e.g., gmapping)
- **Approaches**
  - Beam Sensor Model
  - Likelihood Field Model (used by gmapping)
  - Map Matching
  - Iterated Closest Points (ICP)

Scan Matching (1. Beam Sensor)

ROBOT INTELLIGENCE LAB

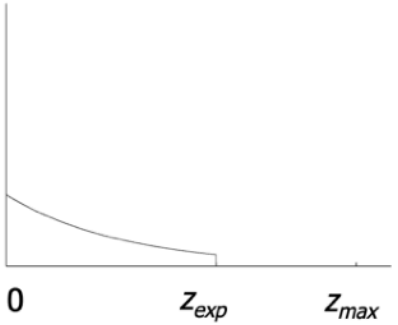
- **Beam Sensor Model**
  - The beam sensor model models how the measurements are acquired.

Measurement noise



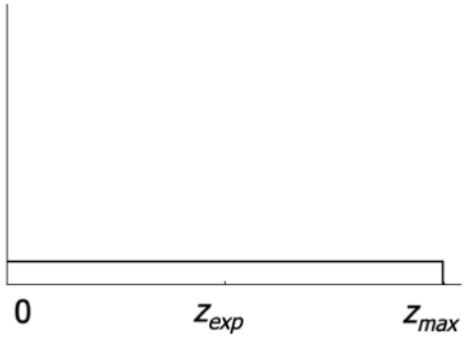
$$P_{hit}(z \mid x,m) = \eta \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2}\frac{(z-z_{exp})^2}{b}}$$

Unexpected obstacles



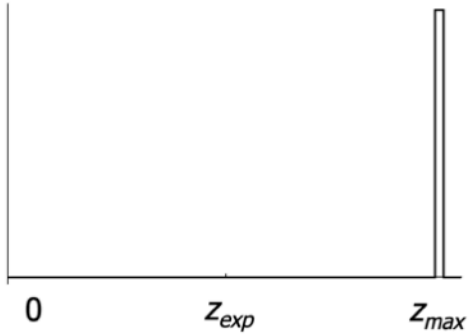
$$P_{unexp}(z \mid x,m) = \begin{cases} \eta \lambda e^{-\lambda z} & z < z_{exp} \\ 0 & otherwise \end{cases}$$

Random measurement



$$P_{rand}(z \mid x,m) = \eta \frac{1}{z_{max}}$$

Max range



$$P_{max}(z \mid x,m) = \eta \frac{1}{z_{small}}$$

  - Learn the parameters based on real data (per each angle)
  - Determine expected distances by ray-tracing.

Scan Matching (1. Beam Sensor)

ROBOT INTELLIGENCE LAB

- **Drawbacks** of **Beam Sensor Model**
  - Lack of smoothness
    - The likelihood $p(z \mid x_t, m)$ is not smooth w.r.t. x_t.
    - When using PF, nearby particle points have very different likelihoods, which could result in **requiring large numbers of samples** to hit some "reasonably likely" states.
    - Also, gradient-based method also suffers from multiple local optima.
  - Computationally expensive
    - Need to ray-cast for **every** sensory reading

Scan Matching (2. Likelihood Field)

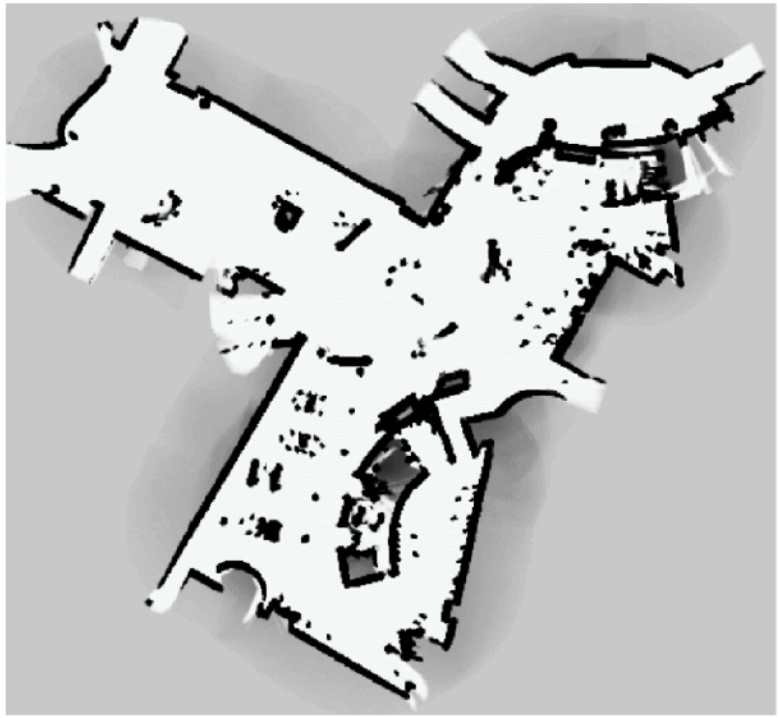
ROBOT INTELLIGENCE LAB

- **Likelihood Field Model**
  - also known as **Beam Endpoint Model** or **Scan-based model**
  - It mainly seeks to overcome lack-of-smoothness and computational limitations of **Beam Sensor Model**.
  - It is an **ad-hoc** algorithm that works well in practice.
  - The main idea is that instead of following along the beam (which is expensive), just check the end-point.
  - Many drawbacks
    - No explicit modeling of people, dynamics, beam.
    - Cannot handle unexplored areas.

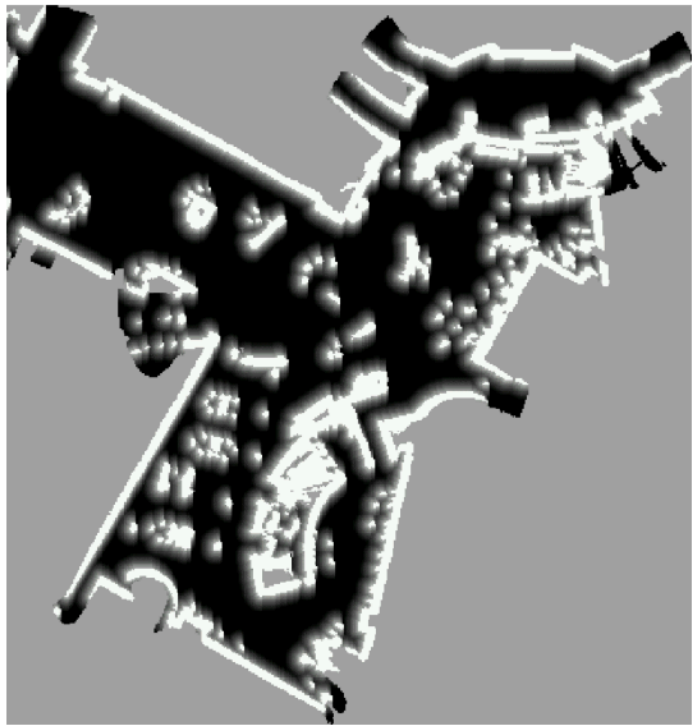
Scan Matching (2. Likelihood Field)

ROBOT INTELLIGENCE LAB

- One can use the likelihood field as a score objective $p(z_t \mid x_t, m)$ for scan matching.



Occupancy grid map



Likelihood field

Scan Matching (2. Likelihood Field)

ROBOT INTELLIGENCE LAB

- **Properties of Scan-based Model**
  - Highly efficient, used 2D tables only.
  - Smooth w.r.t. small changes in robot position
  - Allow gradient-based methods
  - Ignore physical properties of beams.

Scan Matching (3. Map Matching)

ROBOT INTELLIGENCE LAB

- **Map Matching**
  - It generates **small, local maps** from the sensor data and match the generated **local maps** against the global model.
  - The **correlation** score:

$$\rho_{m,m_{local},x_t} = \frac{\sum_{x,y}(m_{x,y} - \bar{m})(m_{x,y,local}(x_t) - \bar{m})}{\sqrt{\sum_{x,y}(m_{x,y} - \bar{m})^2}\sqrt{\sum_{x,y}(m_{x,y,local}(x_t) - \bar{m})^2}}$$

where $\bar{m} = \frac{1}{2N}\sum_{x,y}(m_{x,y} + m_{x,y,local})$

Scan Matching (4. ICP)

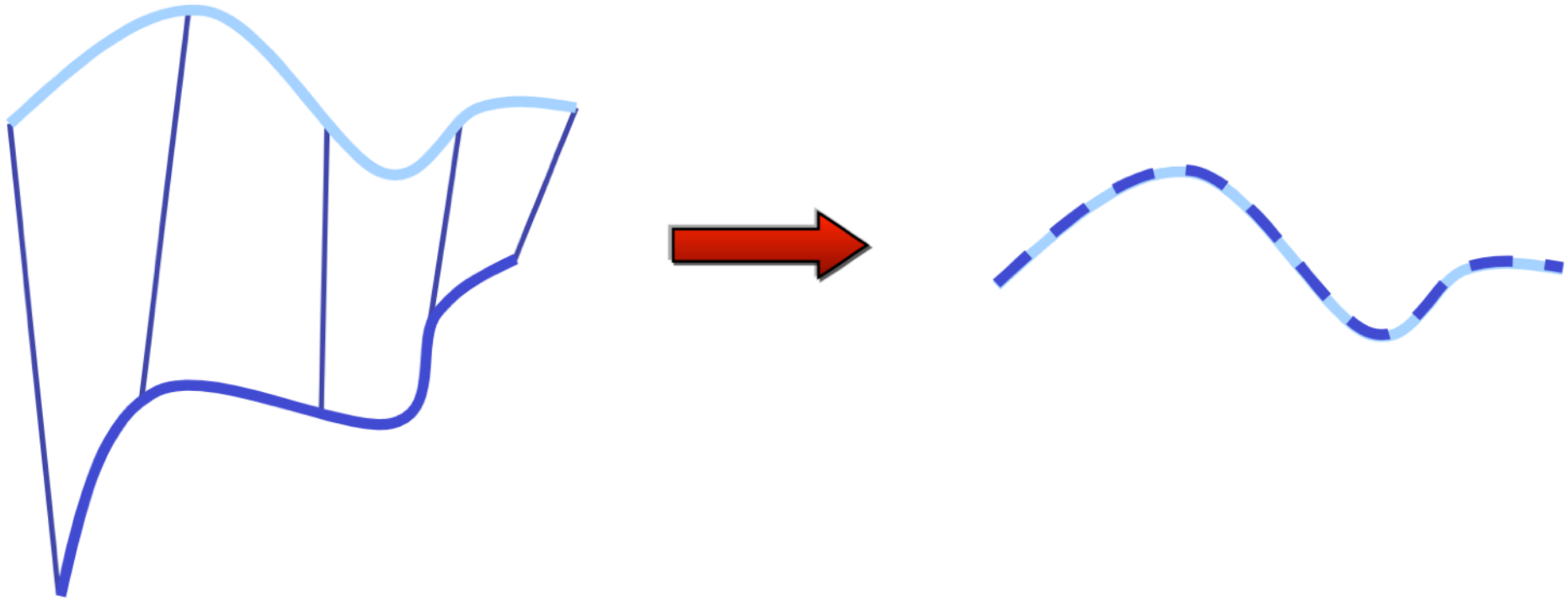
ROBOT INTELLIGENCE LAB

- Given two corresponding point sets:
  - $X = \{x_1, \ldots, x_n\}$ and $P = \{p_1, \ldots, p_n\}$
- The goal is to find translation t and rotation R that minimizes the sum of the squared error:
  - $E(R, t) = \frac{1}{N} \sum_{i=1}^{N_p} \|x_i - Rp_i - t\|^2$ where x_i and p_i are corresponding points.

Scan Matching (4. ICP)

ROBOT INTELLIGENCE LAB

- If the **correct correspondences** are known, the correct relative rotation/ translation can be calculated in a closed form (with SVD)



Scan Matching (4. ICP)

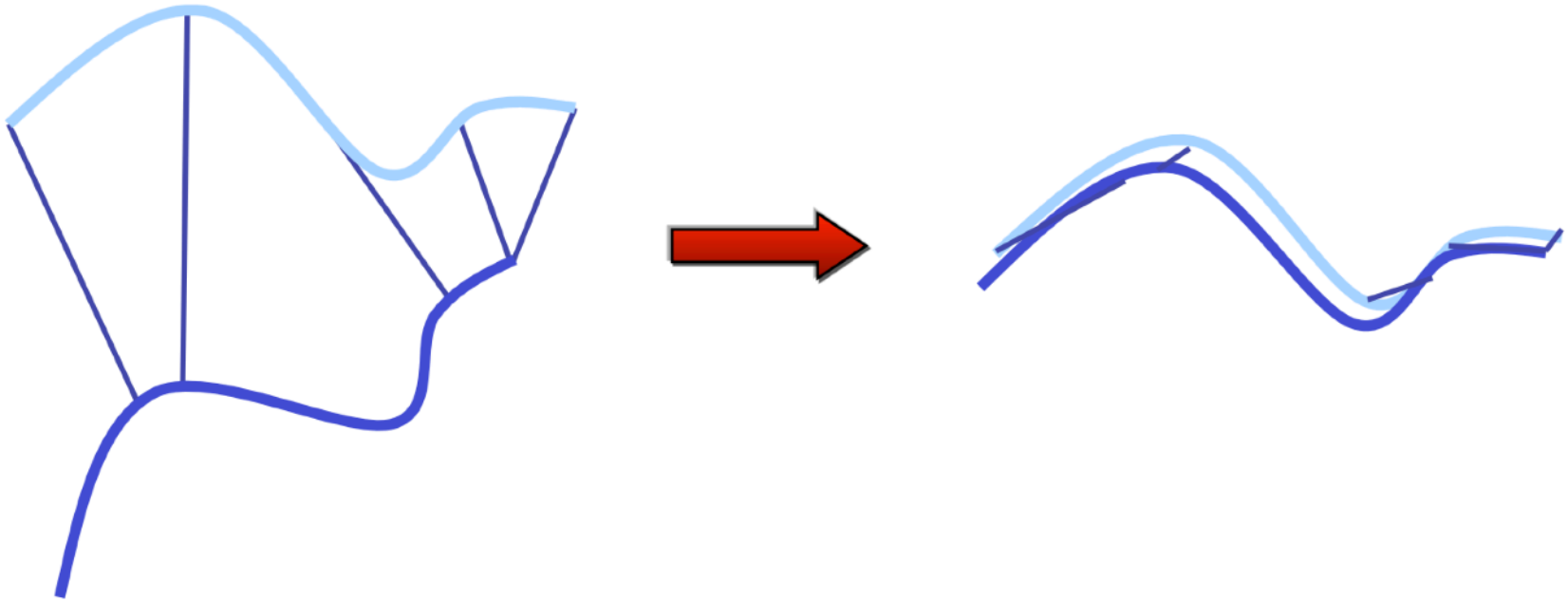
ROBOT INTELLIGENCE LAB

- If the **correct correspondences** are known, the correct relative rotation/ translation can be calculated in a closed form (with SVD)
  - First, find mean zero sets: $X' = \{x_i - \mu_x\} = \{x_i'\}$ and $P' = \{p_i - \mu_p\} = \{p_i'\}$
  - Let $W = \sum_{i=1}^{N} x_i' p_i'^T$ and denote the SVD of W by $W = U\Sigma V^T$. Then the optimal solution of $E(R, t)$ is given by
    - $R = UV^T$
    - $t = \mu_x - R\mu_p$

Scan Matching (4. ICP)

ROBOT INTELLIGENCE LAB

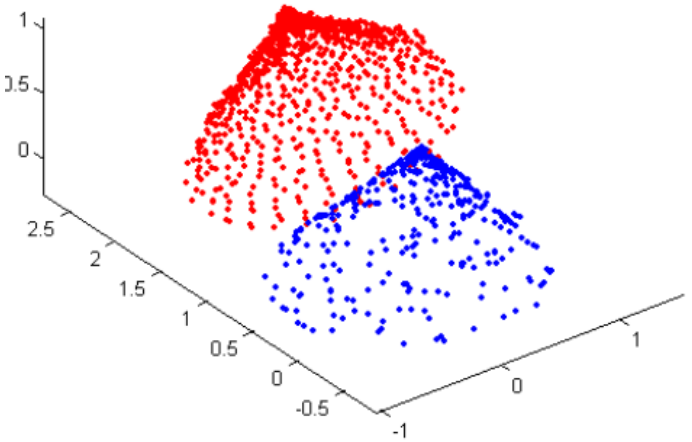
- **Unknown Data Association**
  - If correct correspondences are not known, it is generally impossible to determine the optimal relative rotation/translation in one step.



Scan Matching (4. ICP)

ROBOT INTELLIGENCE LAB

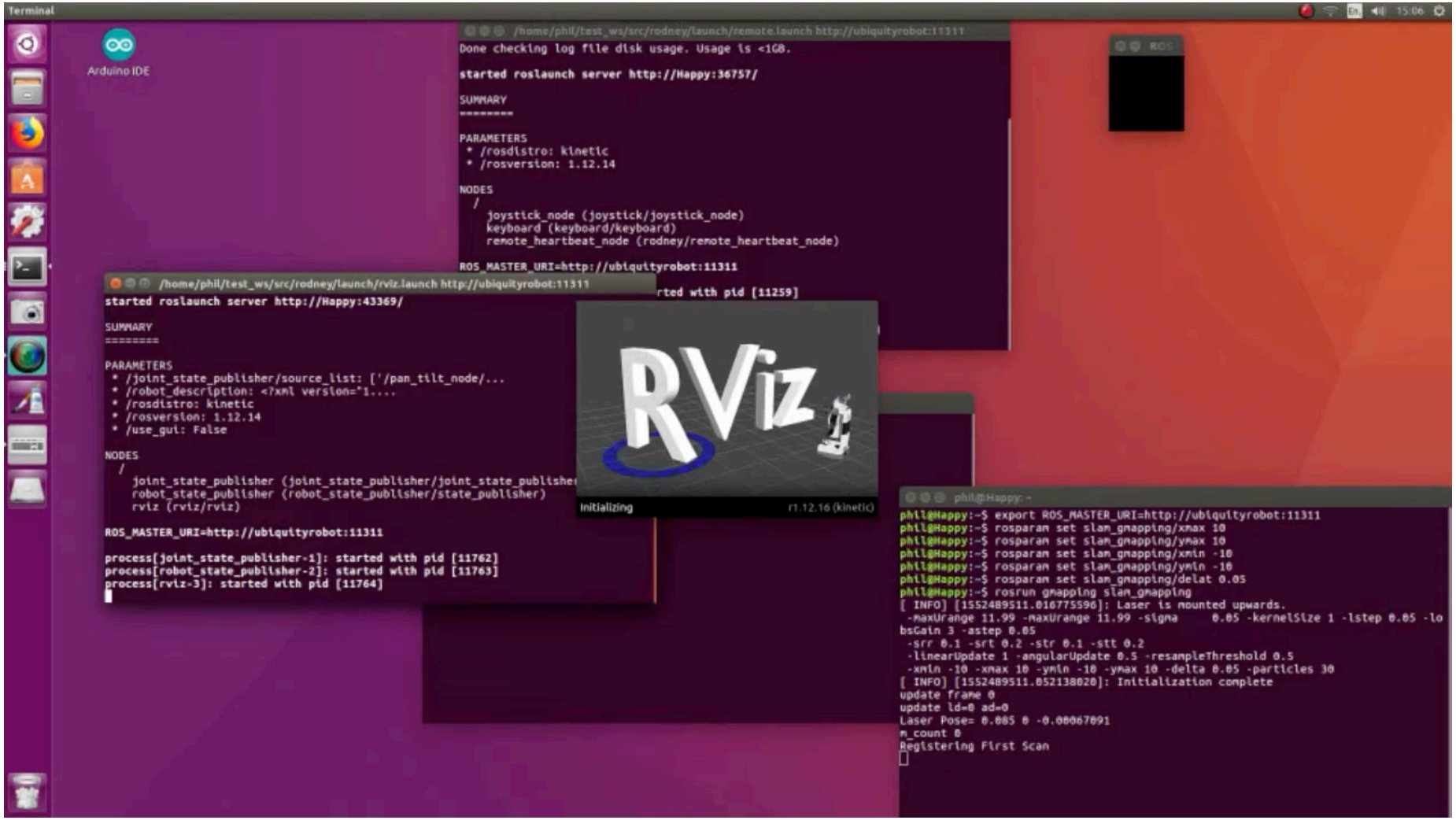
- **Iterated Closed Point (ICP)**
  - The idea is to iterate to make alignment.
    - **Keep pose fixed**, and for each (blue) point find closes match amongst (red) points.
    - **Keep matches fixed** (aka "known correspondences"), find the rigid transformation that minimizes some cost.
  - The major problem is to determine the **correct data associations**.
  - Often **RANSAC** is used for initialization.



Red: test point cloud
Blue: attempted match of model

gmapping

ROBOT INTELLIGENCE LAB



https://youtu.be/Hefxb2y1wjc

ROBOT INTELLIGENCE LAB

Cartographer

"Real-Time Loop Closure in 3D LIDAR SLAM," 2016

Cartographer

ROBOT INTELLIGENCE LAB

- "Real-Time Loop Closure in 3D LIDAR SLAM," 2016
  - Real-time mapping and **loop-closure** at a 5cm resolution.
  - Instead of using a **particle filter**, it generates **submap** by running a pose estimation which is assumed to be sufficiently accurate for short period of time.
  - When the **submap** genration is finished, it takes part in **scan matching** for **loop closure**.
    - If two **submaps** are close enough based on the current pose estimates, a **scan matcher** tries to find the **scan** in the **submap**.
    - If a sufficiently good match is found, it is added as a **loop closing constraint**.

Cartographer

ROBOT INTELLIGENCE LAB

- This is the method behind **cartographer** package in ROS.

cartographer

kinetic melodic *Show EOL distros:* ☐

Documentation Status

Package Summary

✔ Released ✔ Documented

Cartographer is a system that provides real-time simultaneous localization and mapping (SLAM) in 2D and 3D across multiple platforms and sensor configurations.

- Maintainer status: developed
- Maintainer: The Cartographer Authors <cartographer-owners AT googlegroups DOT com>
- Author: The Cartographer Authors <google-cartographer AT googlegroups DOT com>
- License: Apache 2.0
- External website: https://github.com/googlecartographer/cartographer
- Source: git https://github.com/googlecartographer/cartographer.git (branch: 1.0.0)

Package Links

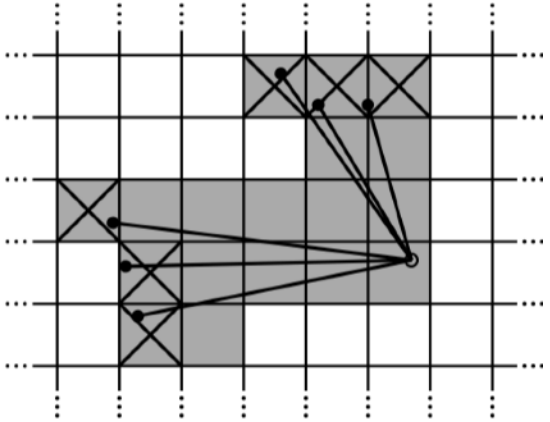
Code API
cartographer website
FAQ
Changelog
Change List
Reviews

Dependencies (1)
Used by (2)
Jenkins jobs (8)

Cartographer

ROBOT INTELLIGENCE LAB

- Cartographer combines **local** and **global** approaches to 2D SLAM.
- **Local approach**
  - Each individual **scan** (translation and rotation of LIDAR observations) is matches against a **submap** (a small chunk of the world) using **scan matching** (a nonlinear optimization that aligns the scan with the submap).
  - **Submap**
    - A few consecutive **scans** are used to build a **submap**.



A **scan** (solid lines) and pixels (in a **submap**) associated with hits (shaded and crossed out) and misses (shade only).

Cartographer

ROBOT INTELLIGENCE LAB

- ***Ceres*** **scan matching**
  - The scan pose ξ is optimized relative to the current **local submap** using Ceres-based **scan matcher** (nonlinear least squares problem).

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

    - $M_{\text{smooth}} : \mathbb{R}^2 \to \mathbb{R}$ is a smooth version of the probability values in the **local submap**.

Cartographer

ROBOT INTELLIGENCE LAB

- **Closing Loops**
  - As **scans** are only matched against a **submap** (containing a few recent **scans**), the local approach slowly accumulates error.
  - Larger spaces are handle by many small **submaps**.
  - **Sparse Pose Adjustment**
    - The relative poses where scans are inserted are stored for loop closure.

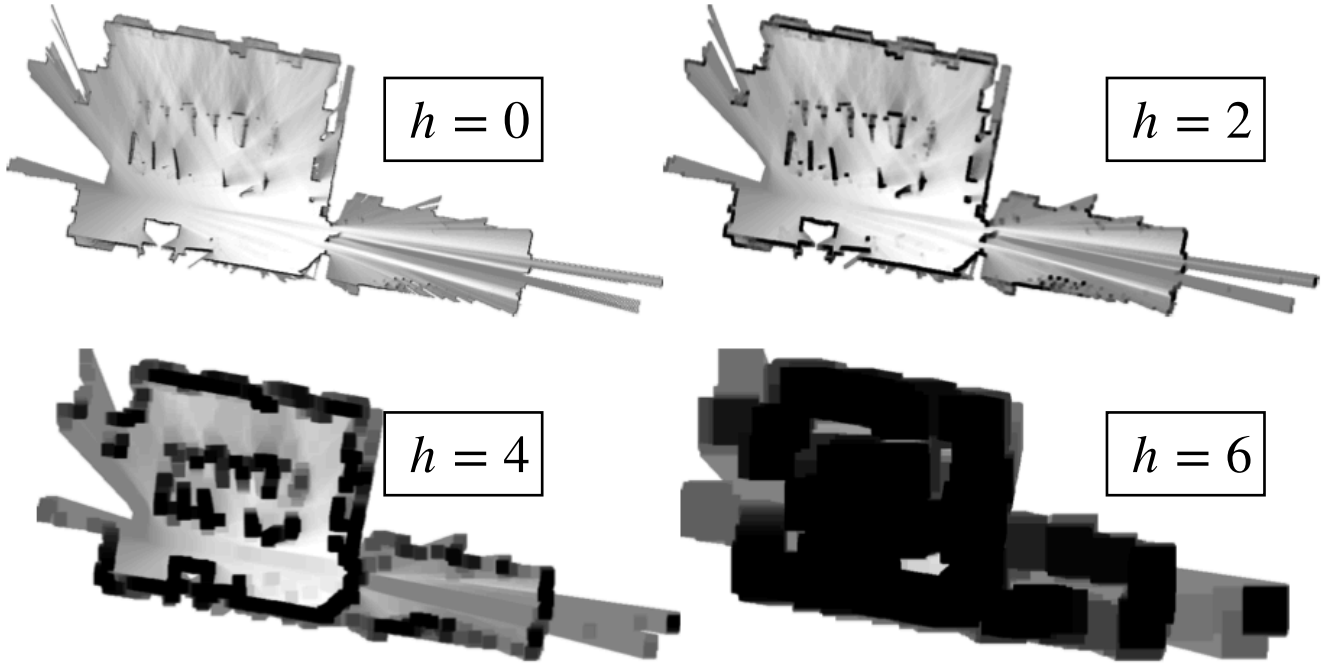
$$\xi^{\star} = \operatorname*{argmax}_{\xi \in \mathcal{W}} \sum_{k=1}^{K} M_{\text{nearest}}(T_{\xi} h_k)$$

  - For efficient scan matching, a **branch and bound approach** is used.
    - The main idea is to represent subsets of possibilities as nodes in a **tree** where the root node represents all possible solutions, $\mathcal{W}$.

Cartographer

ROBOT INTELLIGENCE LAB

- **Precomputed grids** per different height c_h
  - Occupancy maps with different heights are precomputed for computational efficiency (maximum of the values of $2^h \times 2^h$ box of pixels).



Cartographer

ROBOT INTELLIGENCE LAB

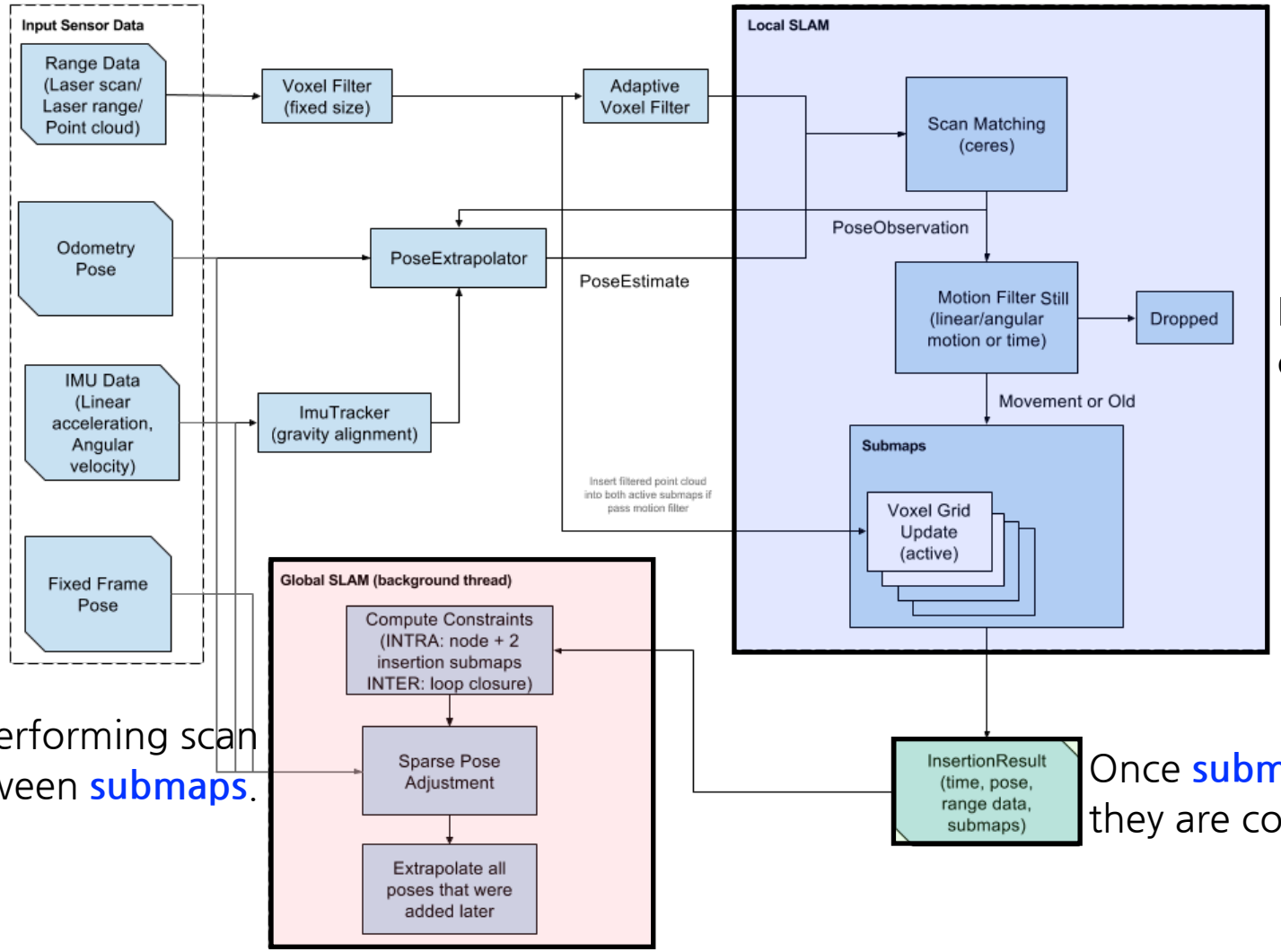
Algorithm 3 DFS branch and bound scan matcher for (BBS)

$best\_score \leftarrow score\_threshold$
Compute and memorize a score for each element in $\mathcal{C}_0$.
Initialize a stack $\mathcal{C}$ with $\mathcal{C}_0$ sorted by score, the maximum score at the top.
while $\mathcal{C}$ is not empty **do**
  Pop c from the stack $\mathcal{C}$.
  if $score(c) > best\_score$ **then**
    if c is a leaf node **then**
      $match \leftarrow \xi_c$
      $best\_score \leftarrow score(c)$
    else
      Branch: Split c into nodes $\mathcal{C}_c$.
      Compute and memorize a score for each element in $\mathcal{C}_c$.
      Push $\mathcal{C}_c$ onto the stack $\mathcal{C}$, sorted by score, the maximum score last.
    end if
  end if
end while
return $best\_score$ and $match$ when set.

Branch step

Cartographer

ROBOT INTELLIGENCE LAB



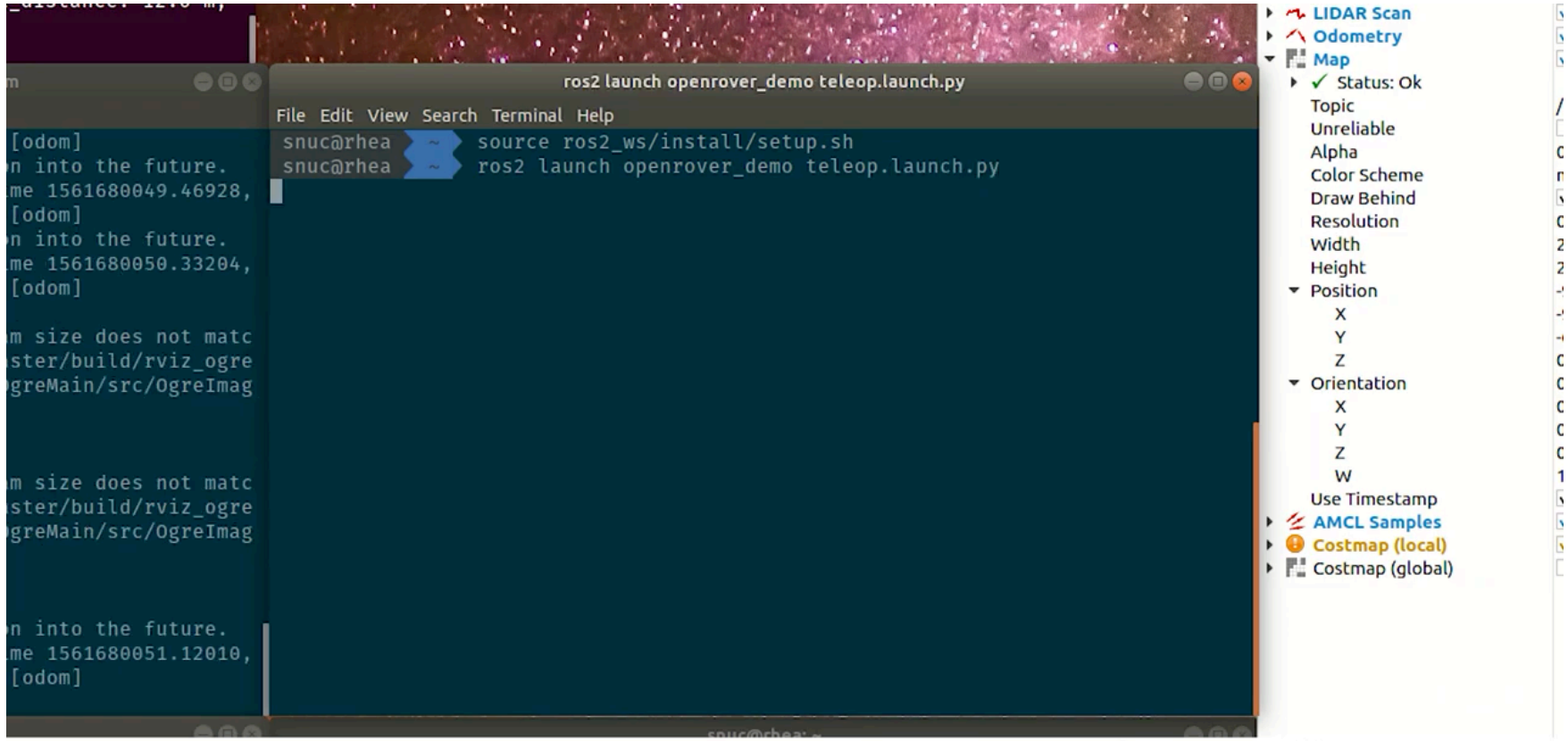
Local SLAM of constructing **Submaps**

Global SLAM of performing scan matching between **submaps**.

Once **submaps** no longer changes, they are considered for loop closing.

Cartographer

ROBOT INTELLIGENCE LAB



ROVER ROBOTICS

Run teleop scripts

Sponsored by:

open robotics

AWS RoboMaker

https://youtu.be/Vgbh60XiMnw

Thank You