Module D: SLAM & Pure Pursuit

Part 1. Introduction to SLAM

Part 2: Particle Filter SLAM

Part 3. Cartographer & Pure Pursuit

References:

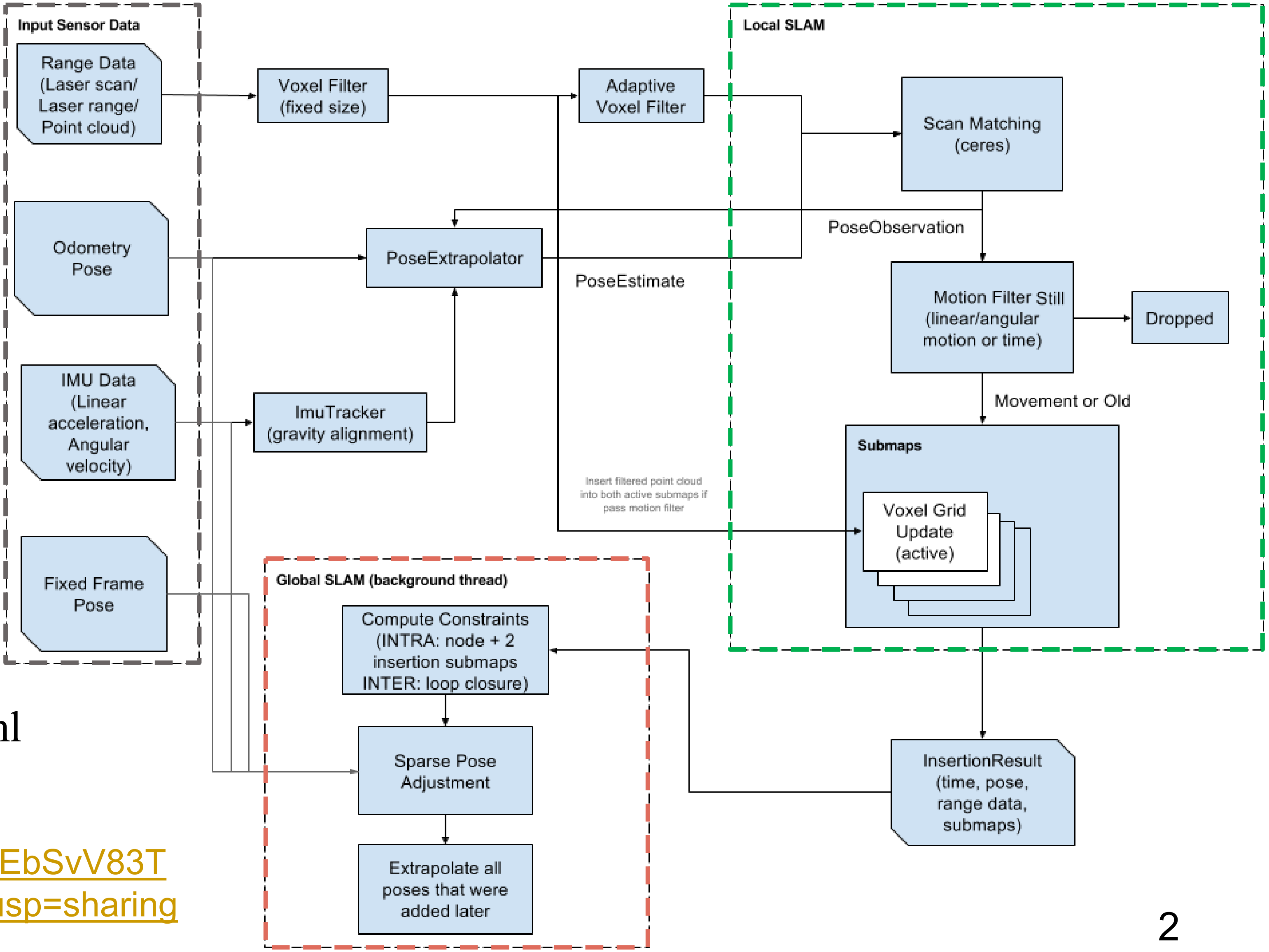
https://f1tenth.org and UPenn ESE 680 Slides

https://google-cartographer.readthedocs.io/en/latest/

Google Cartographer

- Cartographer is a system that provides real-time simultaneous localization and mapping (SLAM) in 2D and 3D across multiple platforms and sensor configurations
- Three major parts:
  - Input Sensor Data Processing
  - Local SLAM (front-end)
  - Global SLAM (back-end)
- Cartographer ROS Integration:

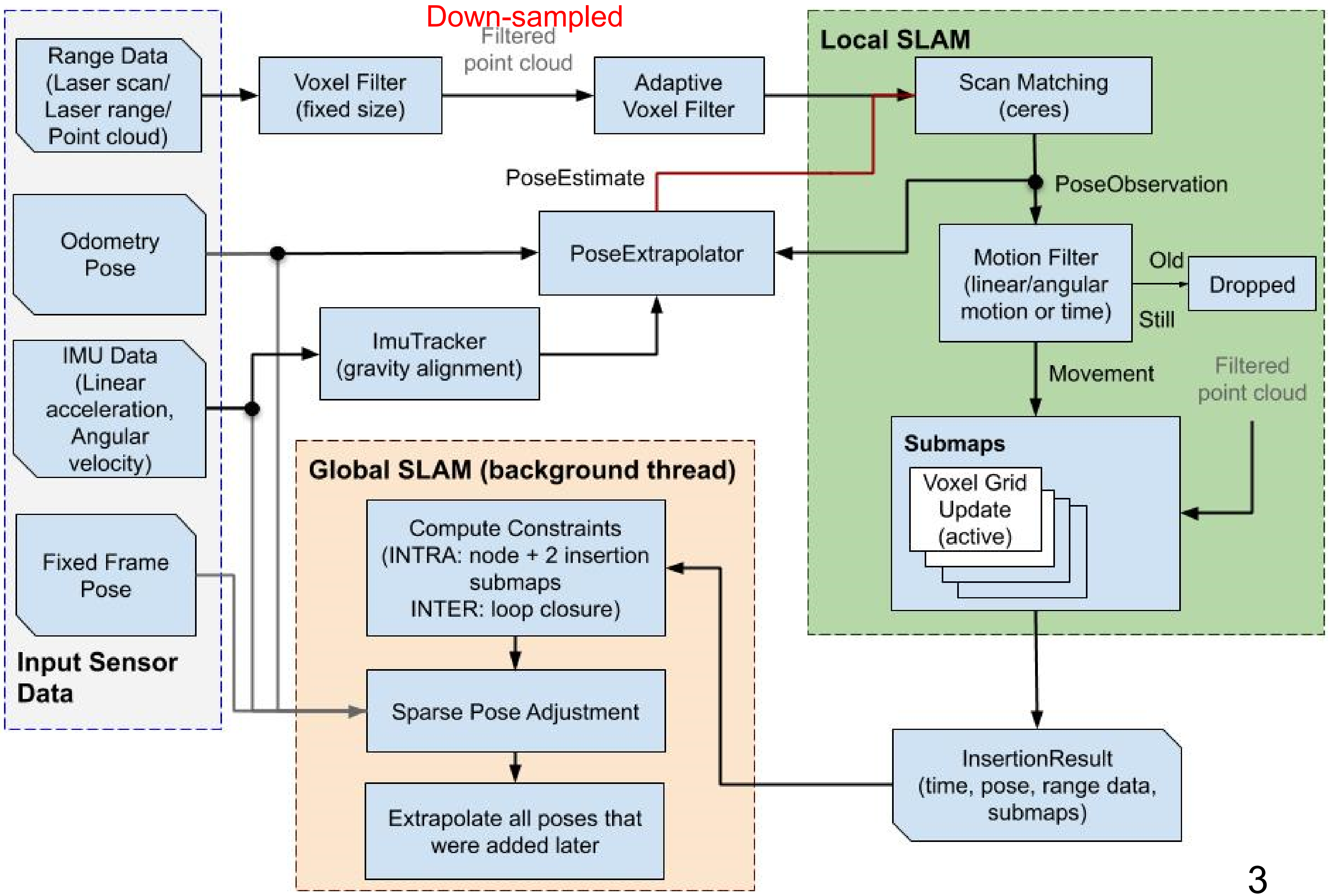
  https://google-cartographer-ros.readthedocs.io/en/latest/index.html



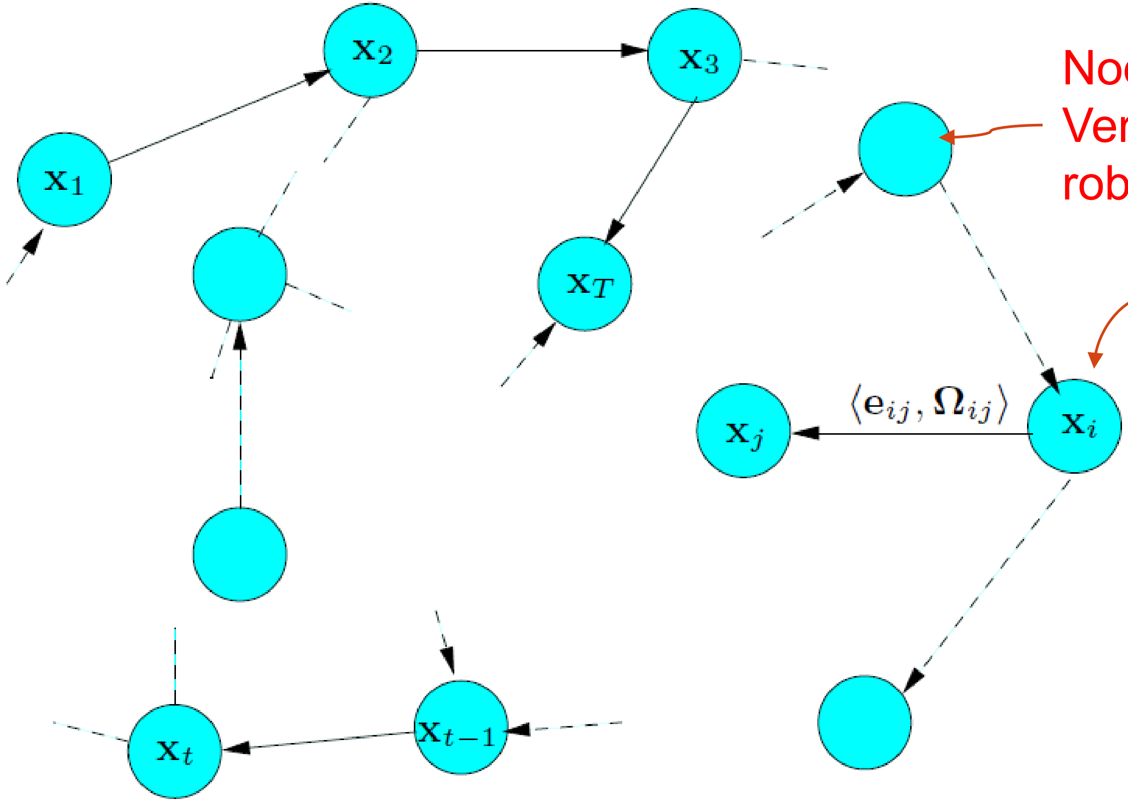
Original figure: https://docs.google.com/drawings/d/1kCJ_dEbSvV83THCUfMikCPw7xFrTkrvRw5r6Ji8C90c/edit?usp=sharing

Google Cartographer- Modified Figure

- Sensor Data Processing:
  - Fixed-frame pose by tf
  - PointCloud2 msg: x-y-z
  - Voxel filter (fixed/adaptive sizes): downsampling to reduce the density of grid;
- Local SLAM (front end):
  - Scan matching by ceres – a Least Squares method
  - Motion filter: detect robot movement; Add point cloud to submaps if moved;
- Global SLAM (background):
  - Scan matching for submaps
  - Loop closure;
  - Pose/trajectory adjustment.

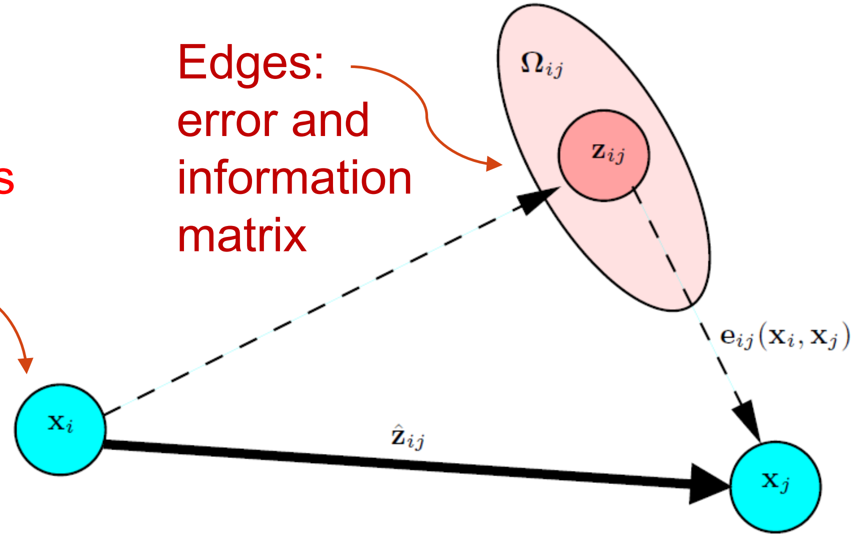


Graph-based SLAM



Nodes (or Vertices): robot poses

Edges: error and information matrix



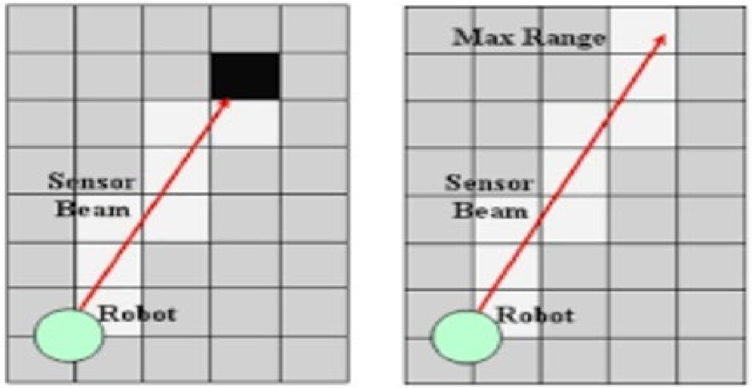
- Error $\boldsymbol{e}_{ij}(\mathbf{x_i}, \mathbf{x}_j)$:
  - Difference between the predicted pose by odometry and the estimated pose from observation (Lidar or camera) – a highly nonlinear function;
  - Linearize the error by Taylor expansion;
- Information matrix $\mathbf{\Omega}_{ij}$:
  - Derived from the linearized errors
- Scan matching the sub maps by ceres – a Least Squares method: $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} \sum_{ij} \boldsymbol{e}_{ij}^T \mathbf{\Omega}_{ij} \boldsymbol{e}_{ij}$

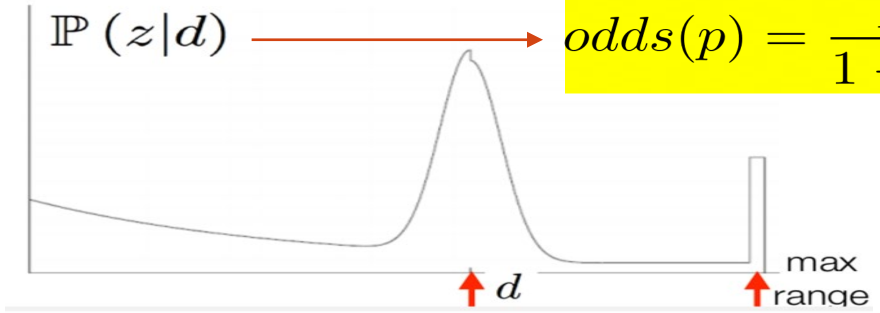
Figures from G. Grisetti, R. Kümmerle, C. Stachniss and W. Burgard, "A Tutorial on Graph-Based SLAM," in IEEE Intelligent Transportation Systems Magazine, vol. 2, no. 4, pp. 31-43, winter 2010.

Construct a Pose graph and minimize the error norm for localization
Again, landmarks are assumed independent from each other and are updated by EKF separately, as in the particle filter SLAM.

What is a submap?

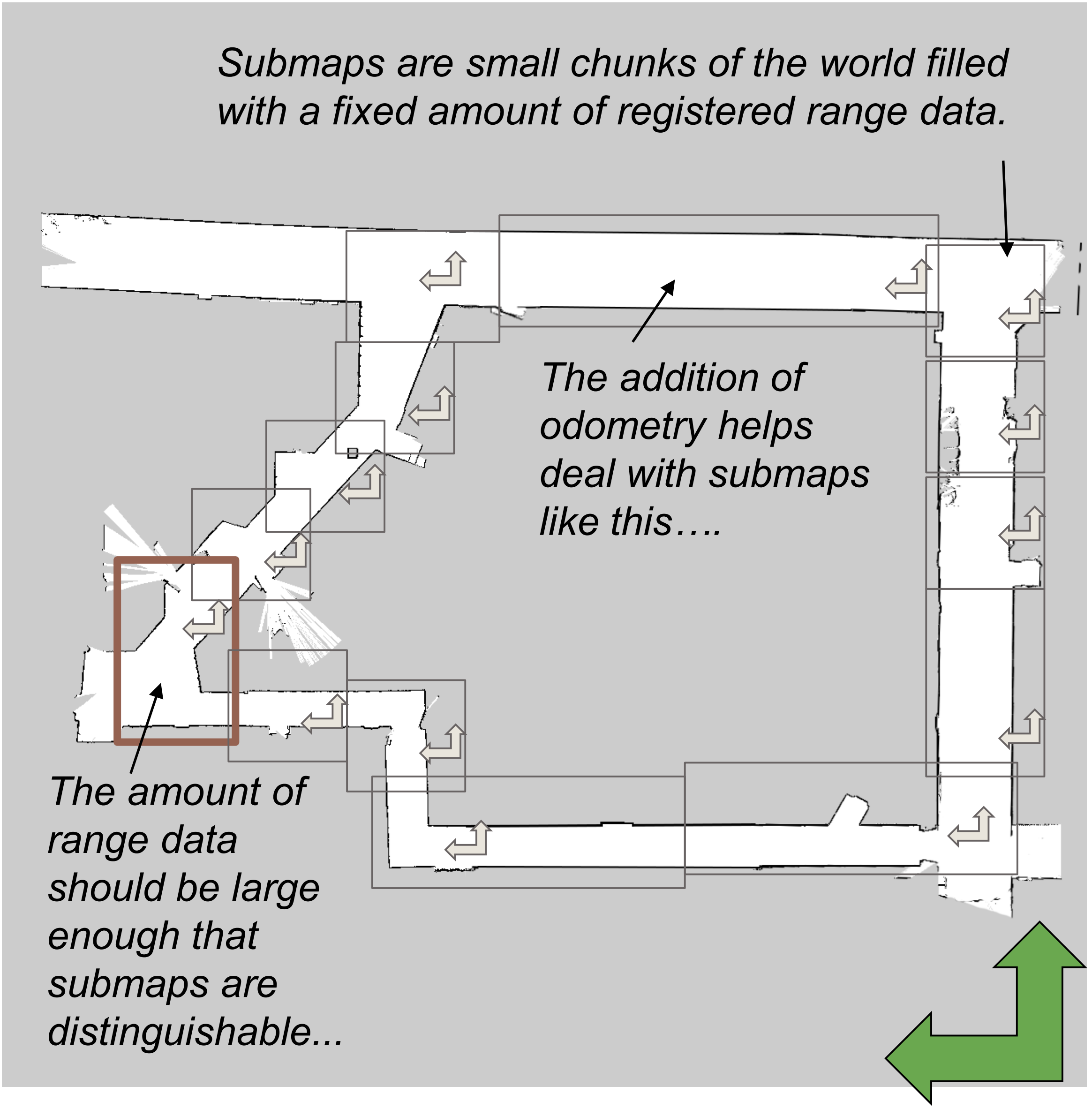
- A submap is a probability grid array where each cell of the map has a fixed width and height. The array contains the odds of the cell being obstructed (occupied)



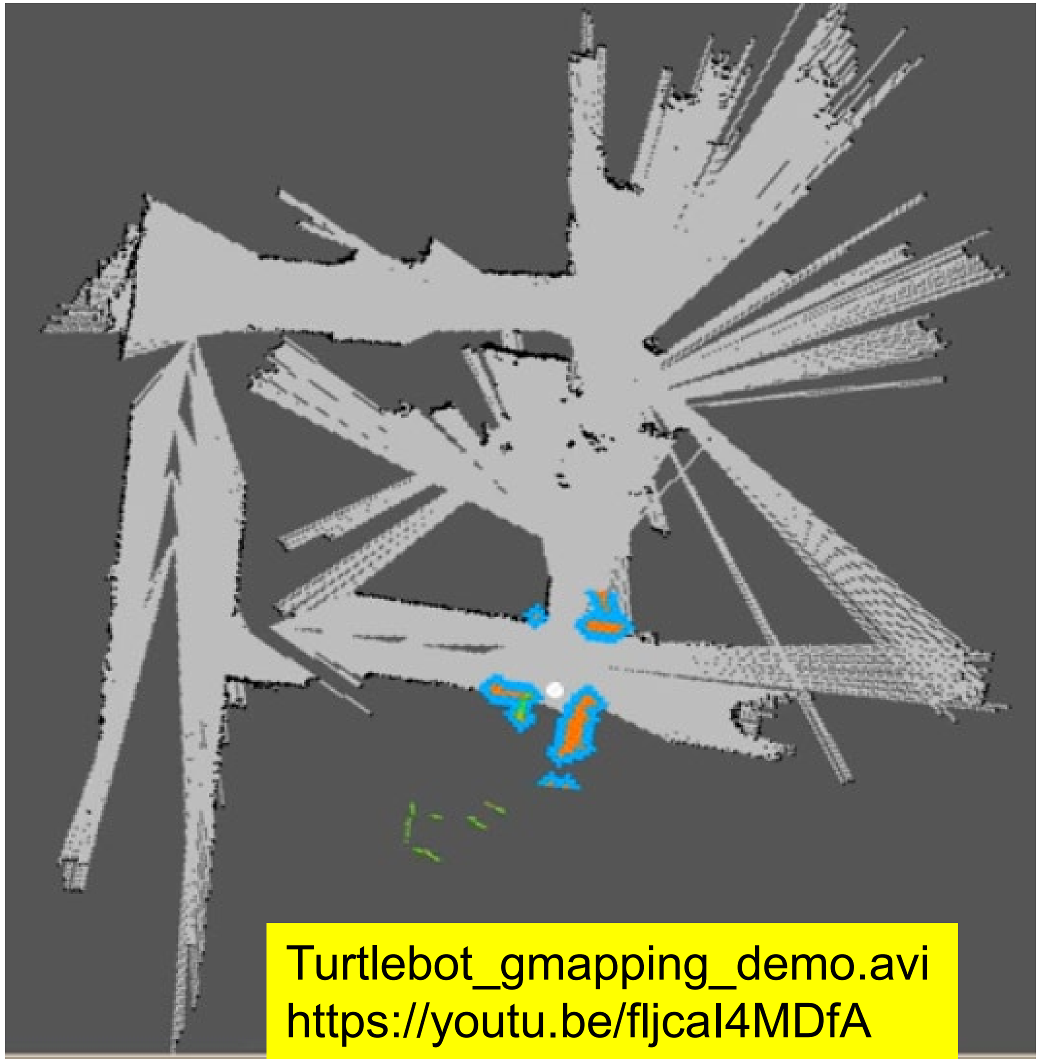


$$odds(p) = \frac{p}{1-p}$$

Ground truth



Scan Matching in Cartographer



Turtlebot_gmapping_demo.avi
https://youtu.be/fljcaI4MDfA

- Methods of Scan Matching:
  - PL-ICP: Point-to-Line Iterative Closest /Corresponding Point algorithm (Censi 2008 ICRA) see UPenn Lab 5 Scan Matching. Use range/angle, high complexity, run slow
  - Scan Matching in Cartographer: use down-sampled point clouds, run fast. Also use Ceres Solver (large-scale nonlinear optimization library): ceres-solver.org, Bundle Adjustment problems → correlative matching
  - Bundle adjustment: Given a set of measured image feature locations and correspondences, the goal is to find 3D point positions and camera parameters that minimize the projection error – a nonlinear least square optimization.
  - RANSAC for outlier rejection

Cartographer: Loop Closure

- Recognize when the robot has returned to a previously mapped region of the world.
- Essentially, two regions in the map are found to be the same region in the world even though their position is incompatible
- Transform submaps to match them by correlation- minimizing the residual error squared.

Sum over each scan, submap combination

Residual

Covariance matrix

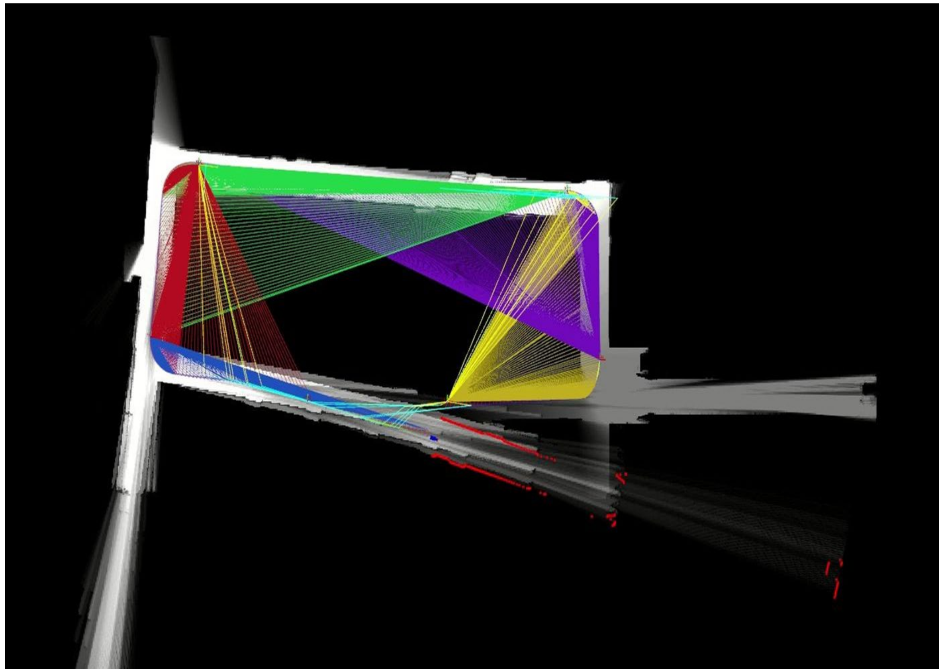
$$\mathbf{argmin}_{\Xi^m, \Xi^s} \frac{1}{2} \sum_{i,j} \rho \left(E^2 \left(\xi_i^m, \xi_j^s; \Sigma_{i,j}, \xi_{ij} \right) \right)$$

Submap poses

Scan poses

Huber loss for outlier rejection

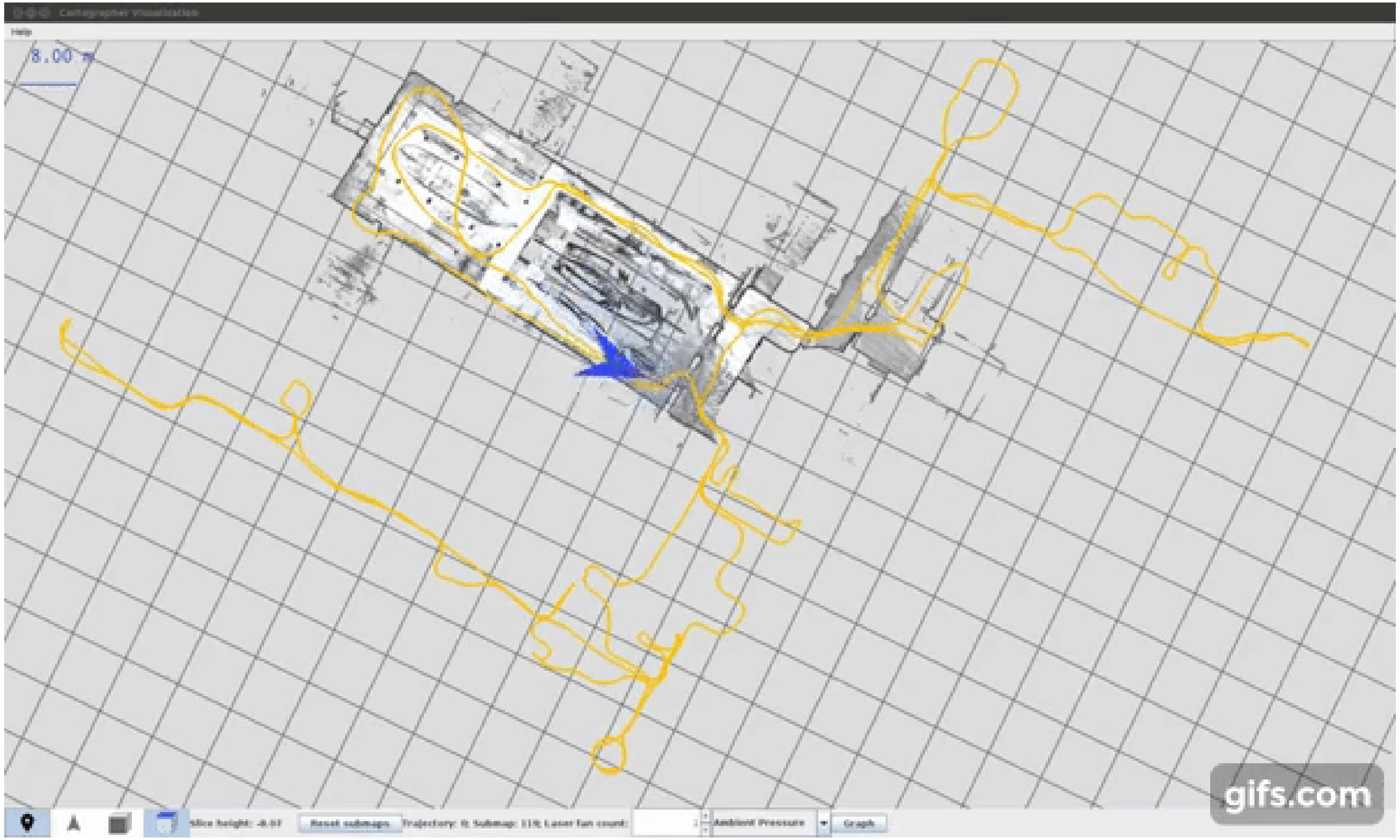
Relative pose between submap & scan (constraint)

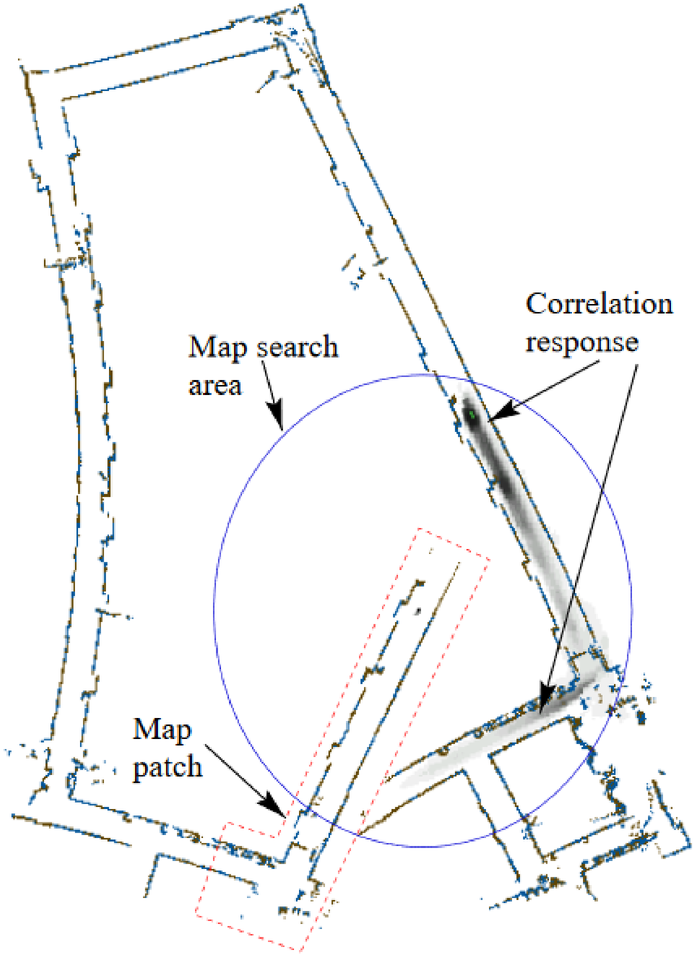


https://drive.google.com/file/d/1nkw8ICFoNe1U44aPM9KJsYYr3HRff6N6/view
Very long video, run at x2 speed

williams_etal_ras2009-SLAM-loopclosure

Cartographer: Loop Closure





Gutmann-CIRA 1999 - Incremental mapping

Using Cartographer: Installation

- We recommend creating a new workspace for cartographer.
- Has some unusual dependencies (e.g. the absolute latest version of protobuf which can mess up other packages).
- Follow the instructions to the letter: https://google-cartographer-ros.readthedocs.io/en/latest/
- When you run the build command, it creates copies of the launch and configuration files in a build directory! From then on, it will use those copies.
- If you modify the originals without re-running the build command, your changes will not take effect!

Warning: Cartographer is regularly updated, follow the online instructions

```
# Install wstool and rosdep.
sudo apt-get update
sudo apt-get install -y python-wstool python-rosdep ninja-build

# Create a new workspace in 'catkin_ws'.
mkdir catkin_ws
cd catkin_ws
wstool init src

# Merge the cartographer_ros.rosinstall file and fetch code for dependencies.
wstool merge -t src https://raw.githubusercontent.com/googlecartographer/cartographer_ros/m
wstool update -t src

# Install proto3.
src/cartographer/scripts/install_proto3.sh

# Install deb dependencies.
# The command 'sudo rosdep init' will print an error if you have already
# executed it since installing ROS. This error can be ignored.
sudo rosdep init
rosdep update
rosdep install --from-paths src --ignore-src --rosdistro=${ROS_DISTRO} -y

# Build and install.
catkin_make_isolated --install --use-ninja
source install_isolated/setup.bash
```

Using Cartographer: F1/10 Model

- Copy the f110_description files to your workspace...
- `cp -r racecar_description ~/cartographer_ws/src/`
- Lets have a look at what's in here…

- Good trick: you can also directly edit the copies for faster tuning, see:
- ~/cartographer_ws/install_isolated/share/cartographer_ros/configuration_files/F110_2d.lua
- ~/cartographer_ws/install_isolated/share/cartographer_ros/launch/F110_2d.launch
- BE CAREFUL, if you rebuild you will lose your changes!

- Move F110-2d.lua to: `~/cartographer_ws/src/cartographer_ros/cartographer_ros/configuration_files/F110_2d.lua`
- This file contains the parameters of the optimization problem. For example, it includes whether or not to utilize odometry (and if so what the name of the topic is).
- Let's have a quick look at the file...

Using Cartographer: Launch File

```
<launch>
  <param name="robot_description" command="$(find xacro)/xacro '$(find
f110_description)/urdf/racecar.xacro'" />
  <node name="robot_state_publisher" pkg="robot_state_publisher"
    type="robot_state_publisher" />

  <node name="cartographer_node" pkg="cartographer_ros"
      type="cartographer_node" args="
          -configuration_directory $(find cartographer_ros)/configuration_files
          -configuration_basename F110_2d.lua"
      output="screen">
      <remap from="odom" to="/vesc/odom" />
      <remap from="imu" to="/imu/data" />
  </node>

  <node name="cartographer_occupancy_grid_node" pkg="cartographer_ros"
      type="cartographer_occupancy_grid_node" args="-resolution 0.05" />
</launch>
```

Using Cartographer: Making a Map with Robot

LEHIGH UNIVERSITY

Terminal 1
- Step 1: `cd & sudo ./jetson_clocks.sh`
- Step 2: `cd ~/your_workspace`
- Step 3: `source devel/setup.bash`
- Step 4: `roslaunch racecar telop.launch`

Terminal 2
- Step 1: `cd ~/cartographer_ws`
- Step 2: `source devel/build_isolated/setup.bash`
- Step 3: `roslaunch cartographer_ros F110_2d.launch`

Terminal 3
- cd ~/your_workspace
- mkdir maps
- cd maps
- rosrun map_server map_saver –o name_of_map
- convert name_of_map.pgm -fuzz 34% -fill black -opaque gray converted_map.pgm

Tips
- Loop closures are very important! Make sure you drive the robot through loops when possible.
- Take a look at F110_2d.lua: you can optimize various parameters for better results!
- http://google-cartographer-ros.readthedocs.io/en/latest/tuning.html