



# *Introduction to Navigation using ROS*

Giorgio Grisetti

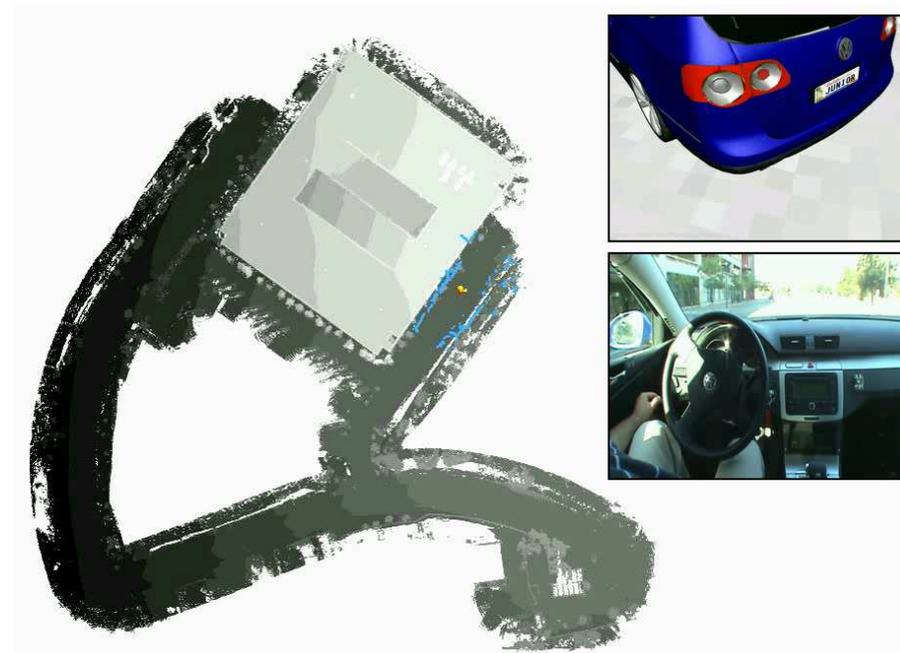
The material of this slides is taken from the Robotics 2 lectures given by G.Grisetti, W.Burgard, C.Stachniss, K.Arras, D. Tipaldi and M.Bennewitz

# Outline

- Navigation Concepts
  - Map, Robot Pose, and Path
  - Taxonomy of Navigation
    - Localization
    - Planning
    - Building a Map
- Building a Map

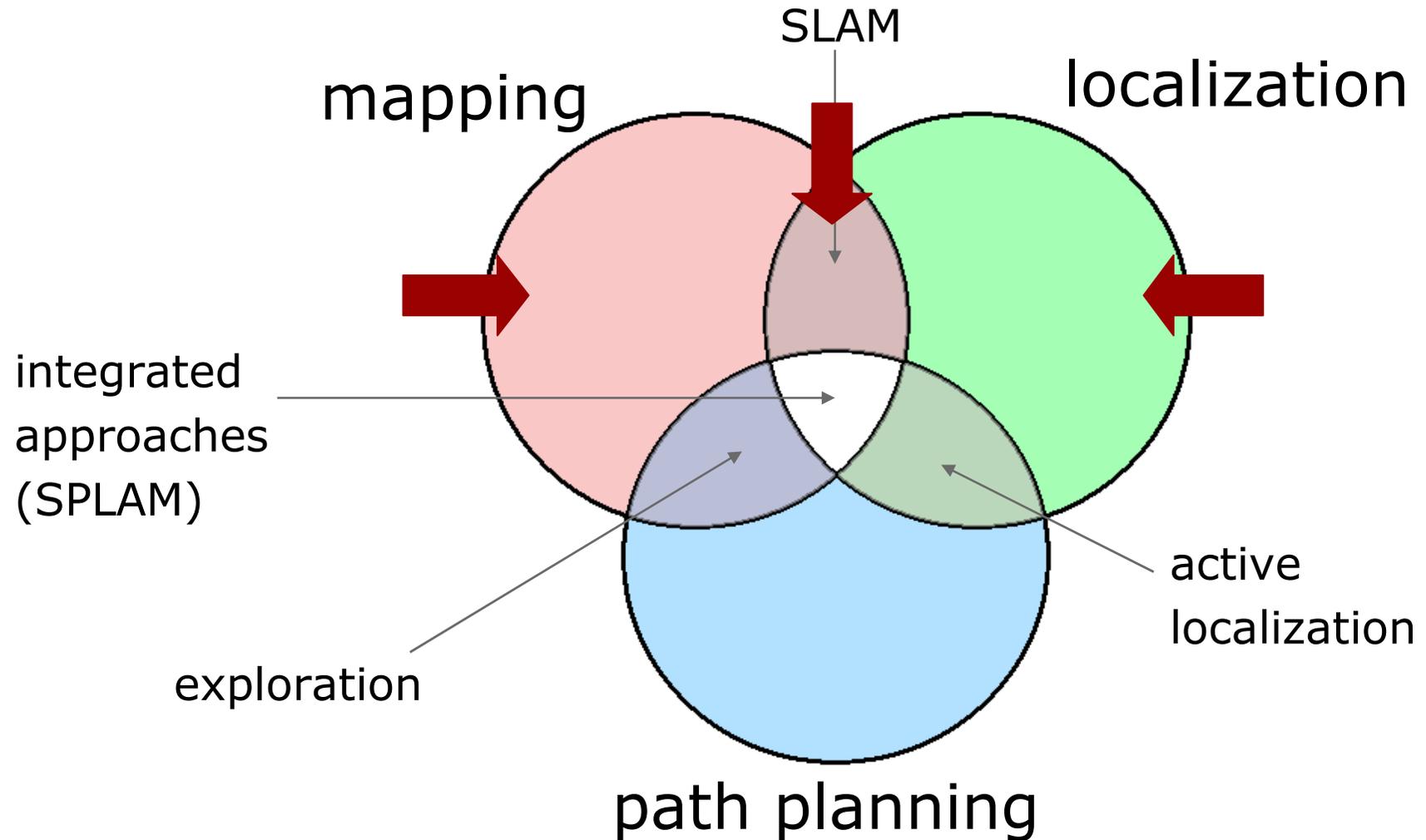
# Sense - Plan - Act

- To **accomplish a task**, a robot should “understand” the environment from its sensor measurements.
- **Understanding**: capturing the information at the right level of abstraction.
- An autonomously navigating robot should know
  - what the **environment** looks like and
  - **where** it is in the environment
  - ...



[courtesy of Rainer Kuemmerle and Dirk Heahnel]

# What is this Talk about?



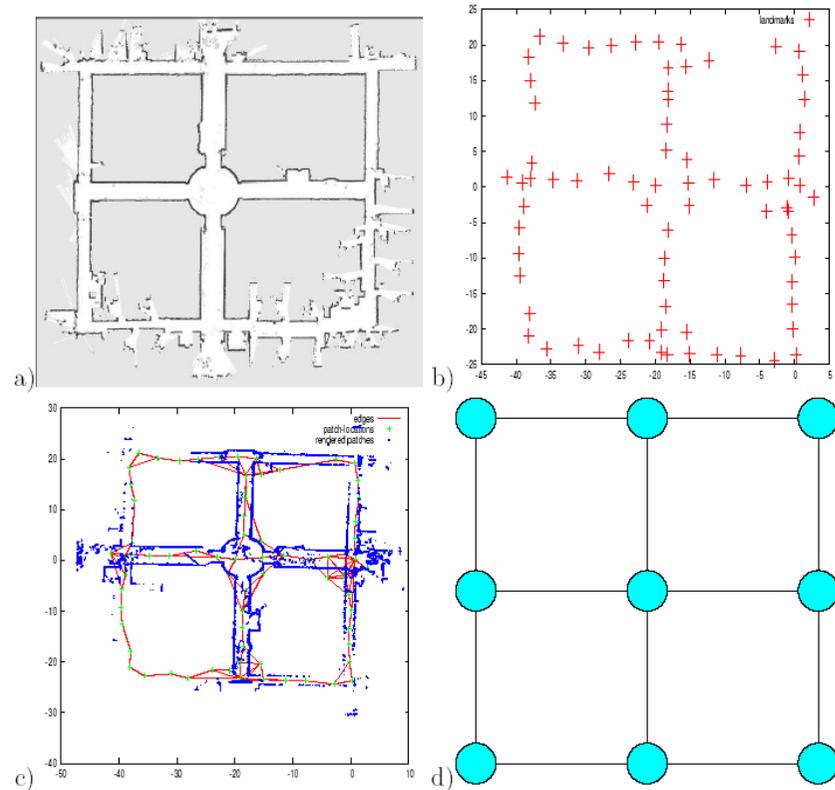
[courtesy of Cyrill and Wolfram]

# Map

- A map is a representation of the environment where the robot is operating.
- It should contain enough information to accomplish a task of interest.

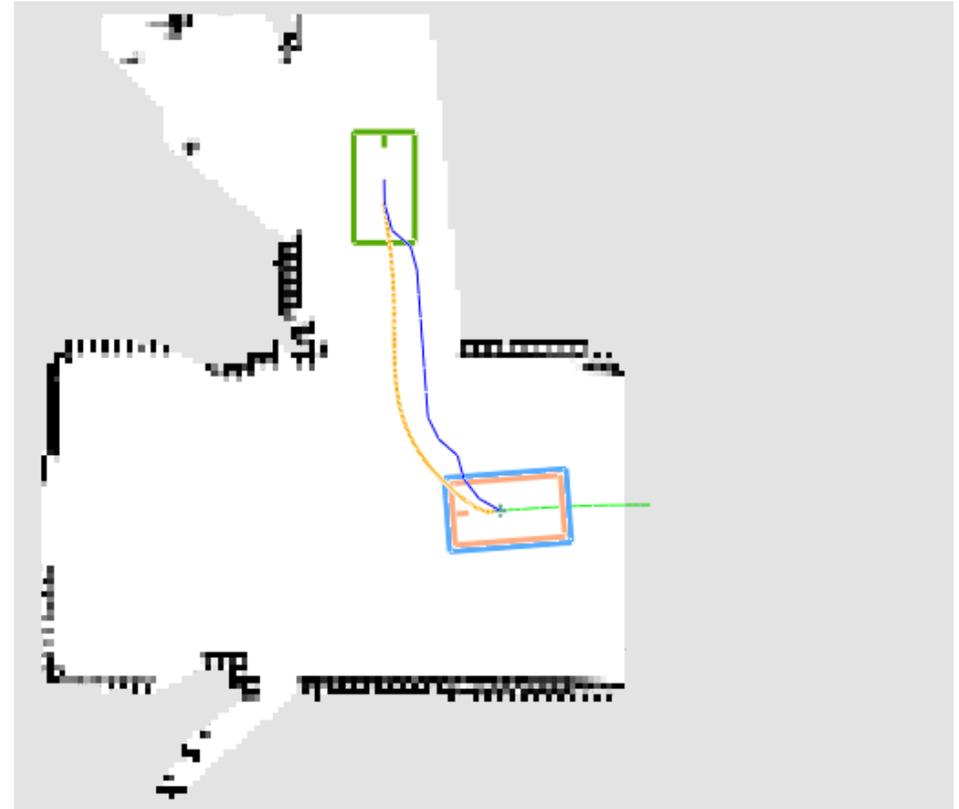
## Representations:

- Metric
  - Grid Based
  - Feature Based
  - Hybrid
- Topological
- Hybrid



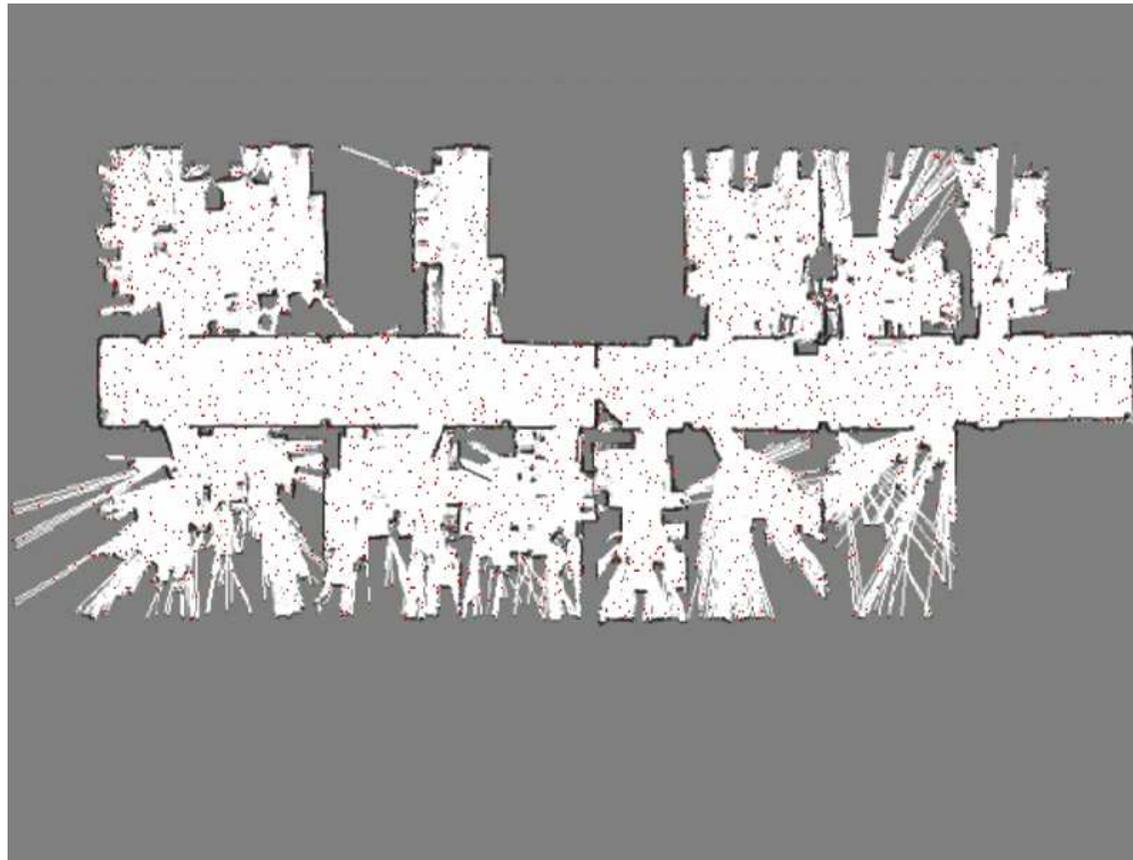
# Robot Pose and Path

- A metric map defines a reference frame.
- To operate in a map, a robot should know its position in that reference frame.
- A sequence of waypoints or of actions to reach a goal location in the map is a path.



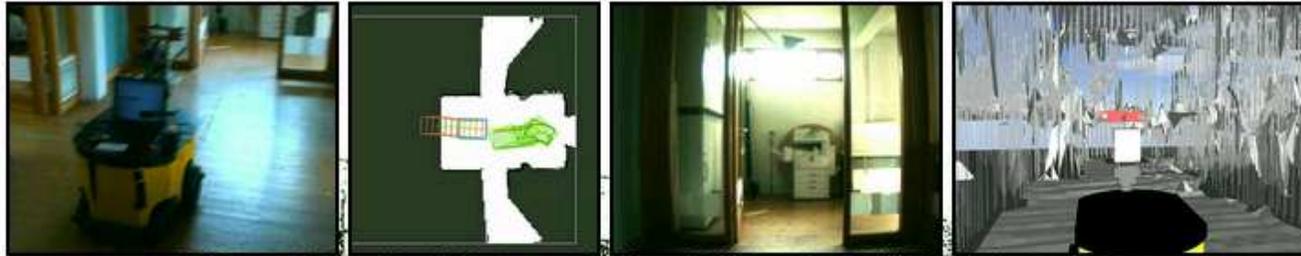
# Localization

- Determine the current robot position, the measurements up to the current instant and a map.



# Path Planning

- Determine (if it exists) a path to reach a given goal location given a localized robot and a map of traversable regions.



# Mapping

- Given a robot that has a perfect ego-estimate of the position, and a sequence of measurements, determine the map of the environment.
- A perfect estimate of the robot pose is usually not available.
- Instead we solve a more complex problem: Simultaneous Localization and Mapping (SLAM).

# SLAM

- SLAM= Simultaneous Localization and Mapping
  - Estimate:
    - the `map` of the environment
    - the `trajectory` of a moving device
- using a sequence of sensor measurements.

**these quantities  
are correlated**



# Putting Parts Together

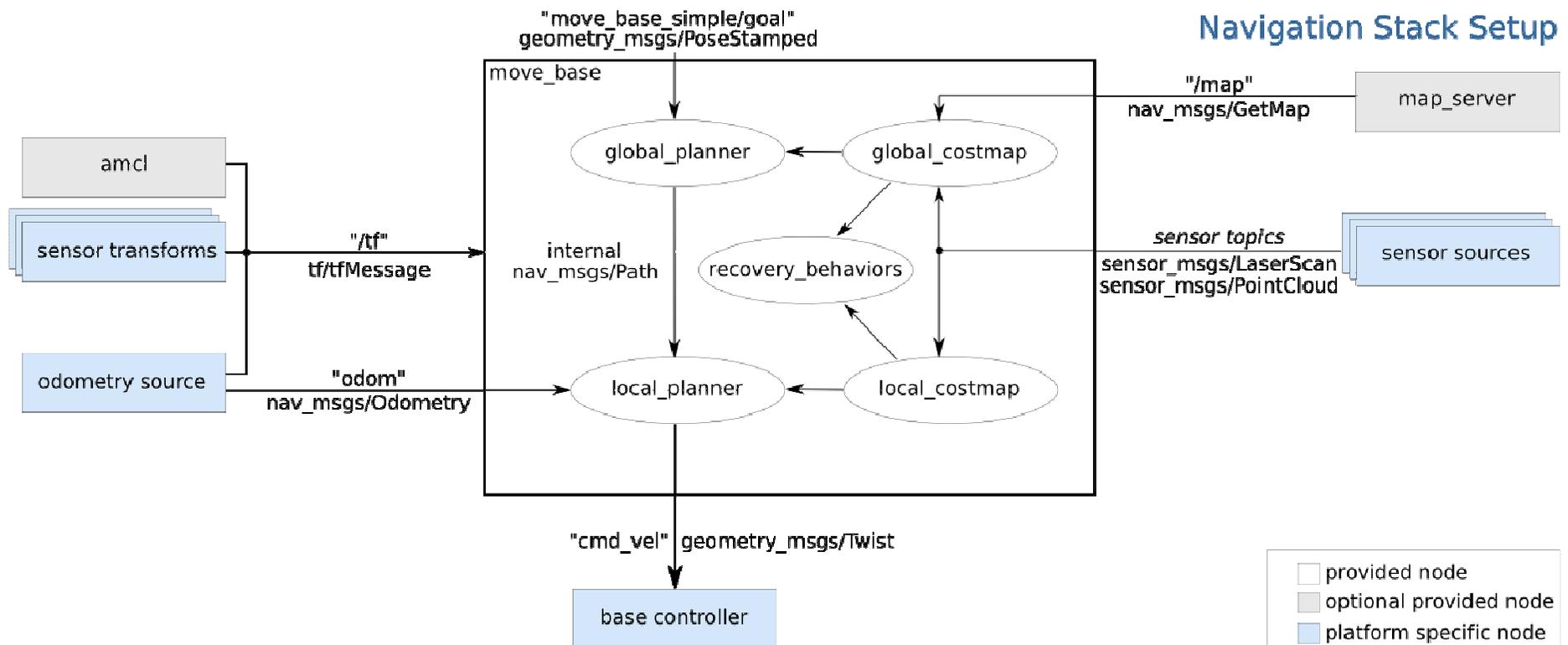
- To navigate a robot we need
  - A map
  - A localization module
  - A path planning module
- These components are sufficient if
  - The map fully reflects the environment
  - The environment is static
  - There are no errors in the estimate
- However
  - The environment changes (e.g. opening/closing doors)
  - It is dynamic (things might appear/disappear from the perception range of the robot)
  - The estimate is “noisy”

Thus we need to complement our ideal design with other components that address these issues, namely

- Obstacle-Detection/Avoidance
- Local Map Refinement, based on the most recent sensor reading.

# ROS Navigation Stack

- Map provided by a "Map Server"
- Each module is a node
- Planner has a layered architecture (local and global planner)
- Obstacle sensing refined on-line by appropriate modules (local and global costmap)



# Building a Map

- There are gozillions of SLAM algorithms around. ROS uses GMapping, which implements a particle filter to track the robot trajectories.
- To build a map you need to
  - Record a bag with /odom, /scan/ and /tf while driving the robot around in the environment it is going to operate in
  - Play the bag and the gmapping-node (see the ros wiki and the live demo), and then save it.
- The map is an occupancy map and it is represented as
  - An image showing the blueprint of the environment
  - A configuration file (yaml) that gives meta information about the map (origin, size of a pixel in real world)

# Localizing a Robot

- ROS implements the Adaptive Monte Carlo Localization algorithm
  - AMCL uses a particle filter to track the position of the robot (see paper of Fox et al.)
  - Each pose is represented by a particle.
  - Particles are
    - Moved according to (relative) movement measured by the odometry
    - Suppressed/replicated based on how well the laser scan fits the map, given the position of the particle.
- The localization is integrated in ROS by emitting a transform from a map-frame to the odom frame that “corrects” the odometry.
- To query the robot position according to the localization you should ask the transform of `base_footprint` in the map frame.

# Localization and MARRtino

- AMCL relies on a laser
- Unless you want to spend 5K euro, you will not get a laser, so your robot will not localize with this procedure
- However...
- You can get a kinect/xtion sensor, that provides data useful to simulate a laser scanner (how?)
- These data can then be plugged in AMCL et voila' you get your system running.

# Homework

- Bring up a working navigation stack by downloading the git repository on the web page
- Steer the robot through rviz by issuing target locations with the mouse
- Once you have all these components running,
- Write a node that whenever it receives a laser scan
  - computes the position of the laser scan (look in the header of the laser message)
  - Uses a tf listener to determine the position of the laser w.r.t. the **/map** frame
  - Writes a line in a file with the following format
  - `LASER_POSE <timestamp.sec>.<timestamp.usec> <x y z qx qy qz qw>`