RBE 3002 CDR

Team 13

Path Planner

We use the path planning node that we developed in lab3.

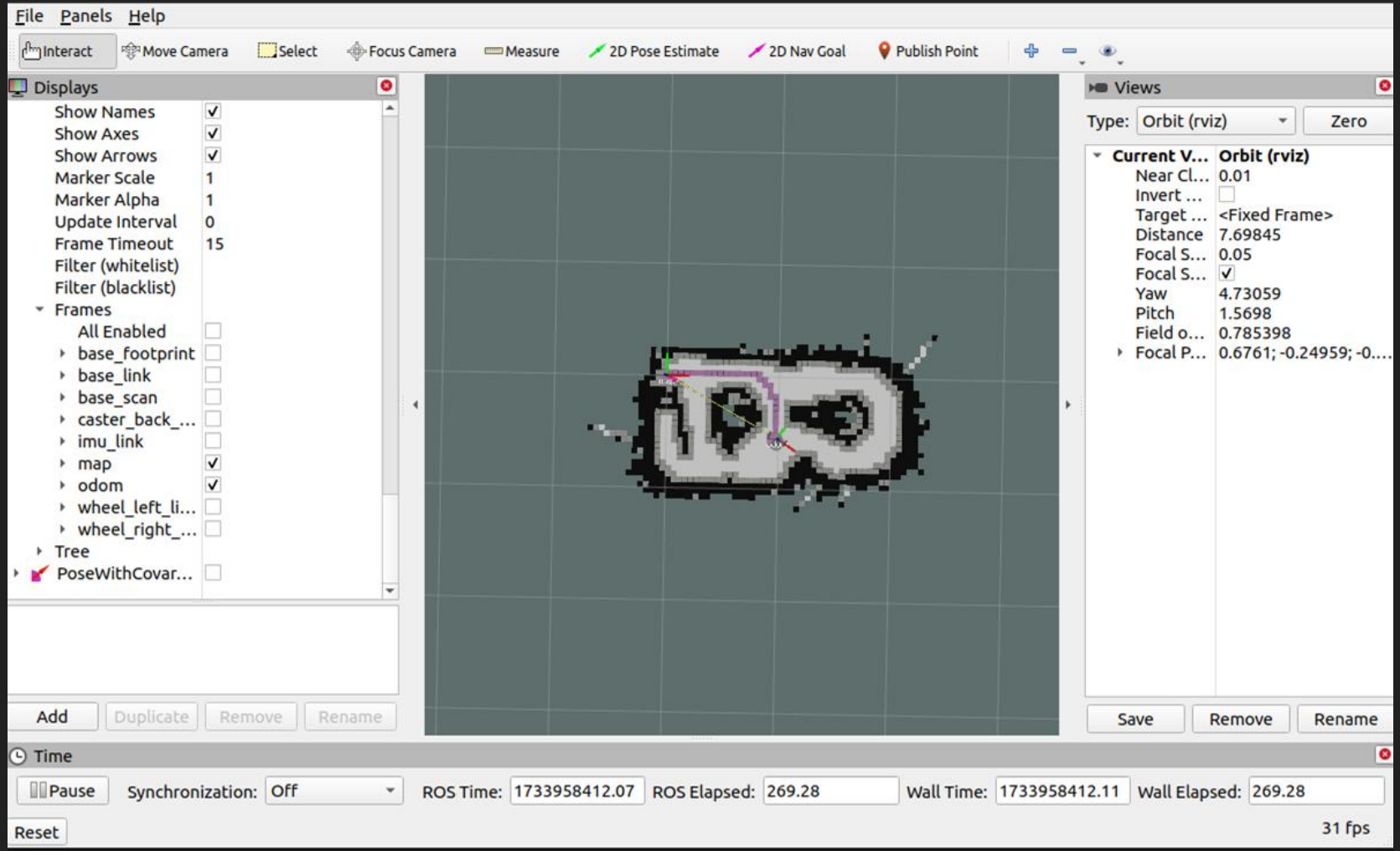
The pathplanner node uses A* to find the path

- Special Heuristics
  - If a cell in the path is close to the cspace, extra cost is added to ensure the path tends toward the middle of the aisle

- Checks if starting point is in cspace, and tries to find a walkable neighbor

- Gets list of centroids from Lab4.
  - If it cannot create a path to the target centroid, it removes that centroid from the list and tries finding a path to the next centroid.
  - If it cannot find a path to all of the centroids in the list, it creates a path to go home.

Used: Phase 1, Phase 2, Phase 3


File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Show Names
Show Axes
Show Arrows
Marker Scale 1
Marker Alpha 1
Update Interval 0
Frame Timeout 15
Filter (whitelist)
Filter (blacklist)
Frames
All Enabled
base_footprint
base_link
base_scan
caster_back_...
imu_link
map
odom
wheel_left_li...
wheel_right_...
Tree
PoseWithCovar...
Add
Duplicate
Remove
Rename
Views
Type: Orbit (rviz)
Zero
Current V... Orbit (rviz)
Near Cl... 0.01
Invert ...
Target ... <Fixed Frame>
Distance 7.69845
Focal S... 0.05
Focal S...
Yaw 4.73059
Pitch 1.5698
Field o... 0.785398
Focal P... 0.6761; -0.24959; -0....
Save
Remove
Rename
Time
Pause
Synchronization: Off
ROS Time: 1733958412.07
ROS Elapsed: 269.28
Wall Time: 1733958412.11
Wall Elapsed: 269.28
Reset
31 fps

Pure Pursuit

Set a variable to the optimal distance for the robot to “follow the carrot” from–this is done through tuning via a trial and error approach.

Add error handling for the case when the robot loses the path: increase the distance variable until the value is within epsilon, set that goal, once it’s reached, set the distance variable back and repeat the process.

Add the final conditional for when the goal is reached. Through trial and error, it was discovered that the optimal way to determine if the goal was reached was by using the distance variable multiplied by a coefficient, and calculating the difference between that value and the final cell in the path list.

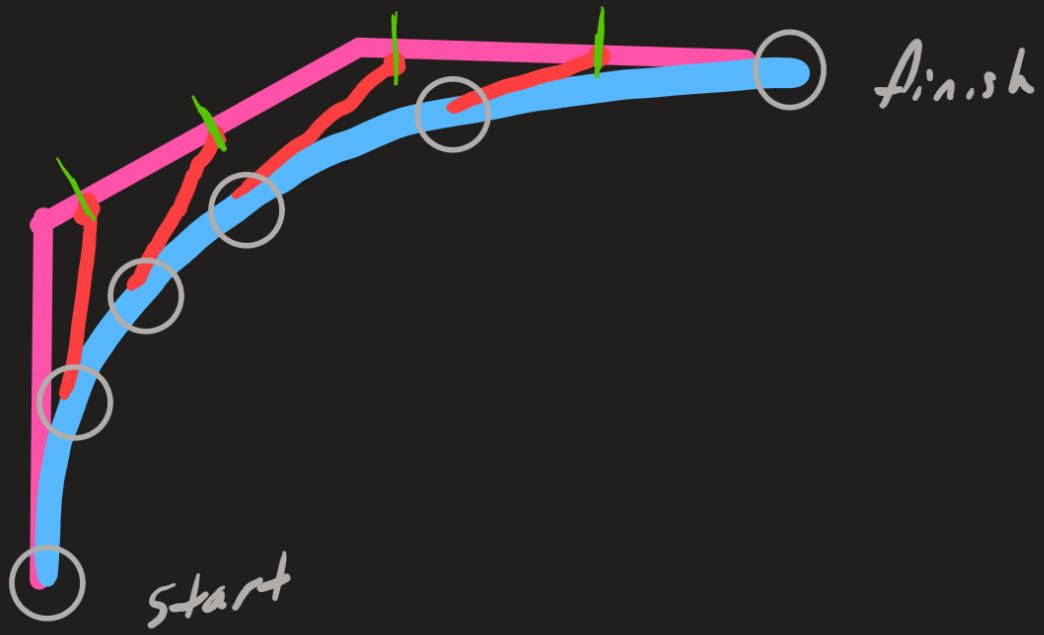
If there are no more centroids, there is a new message published which uses the (0,0) odom coordinate to send the robot to its starting position.

Once the goal-reached message is received, the next centroid in the list of frontier centroids will be used to generate a new path and repeat pure pursuit.

If there are no more centroids, there is a new message published which uses the (0,0) odom coordinate to send the robot to its starting position.

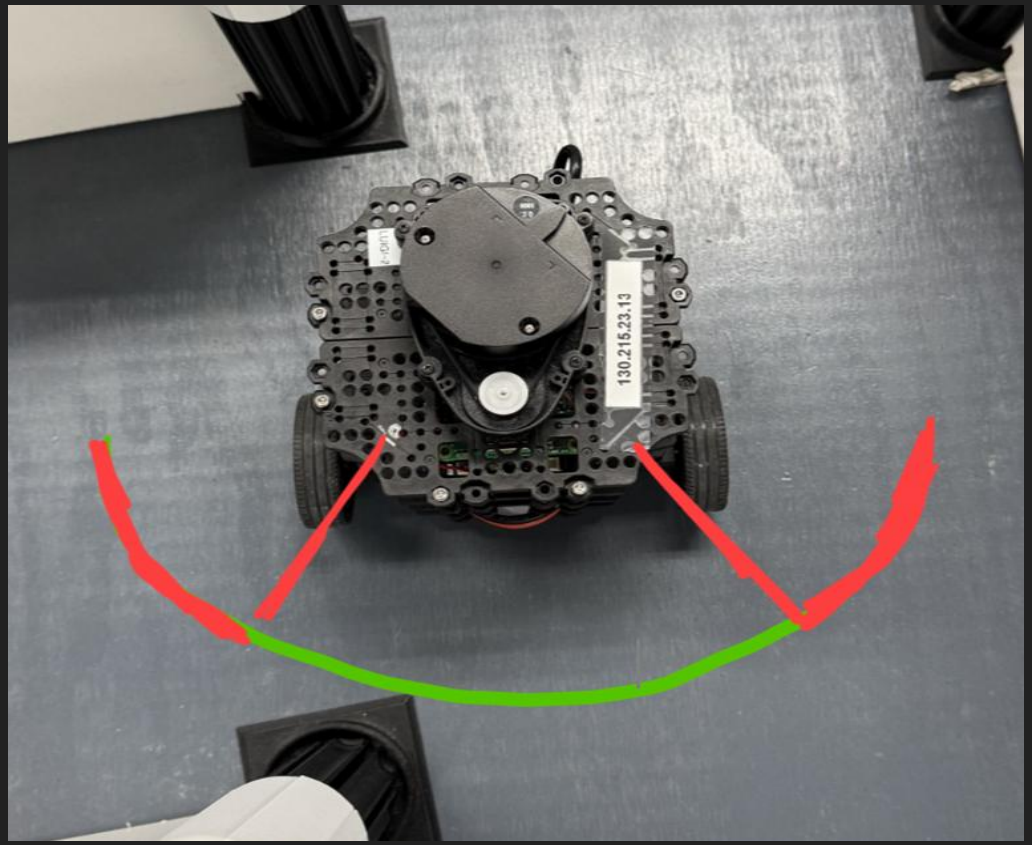
Used: Phase 1, Phase 2, and Phase 3

Pure Pursuit “Follow the carrot”



Pure Pursuit Optimization

To overcome the issue of hitting the beams with the tires, wall sensing was implemented. If the robot detected a wall in front of it in three sections: front right, front, and front left.

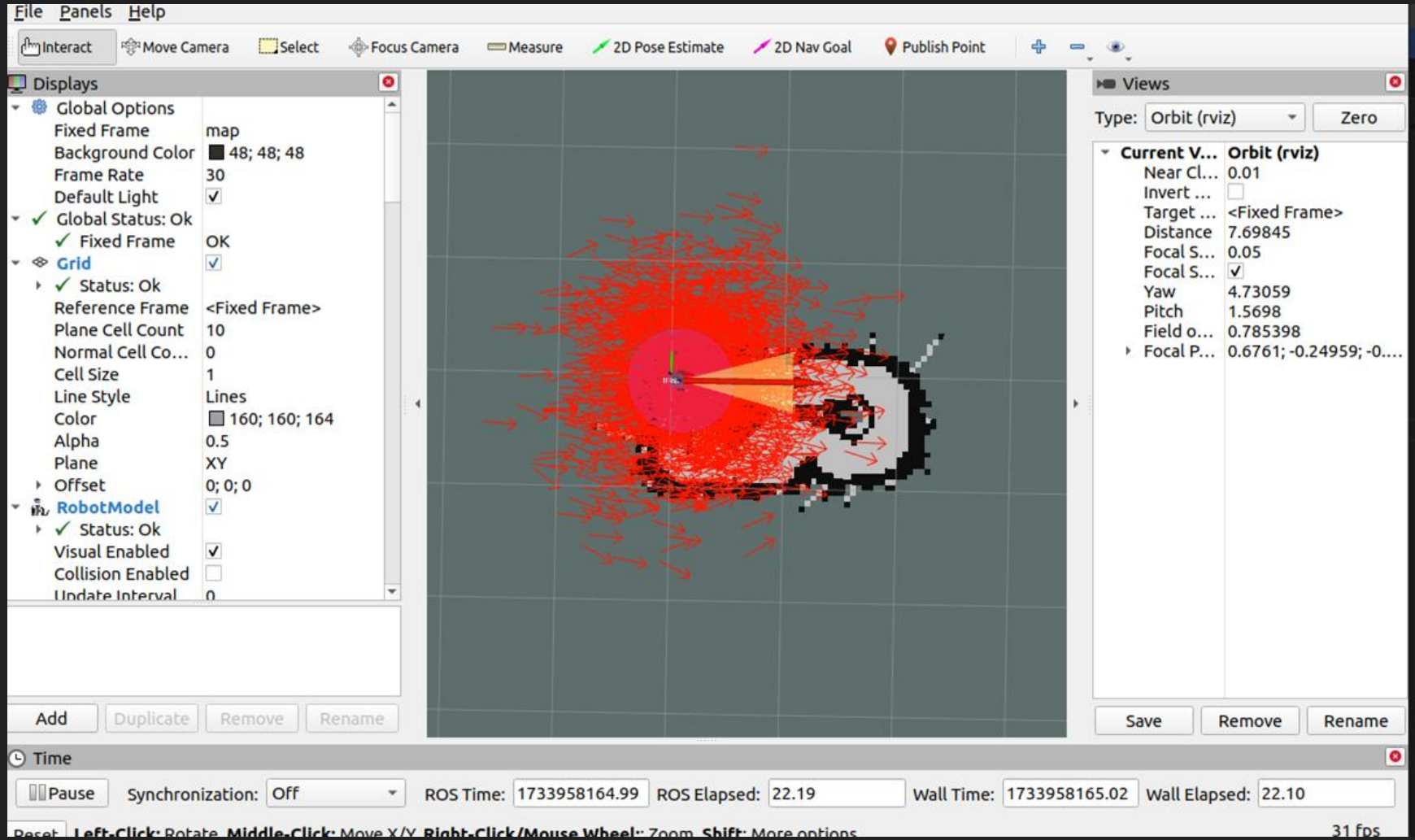


Centroids

- Frontier cells are found be checking the 4 neighbors of a cell
  - If a neighboring cell is part of the unknown area, the cell is a frontier cell
- Frontier cells are grouped by proximity
  - A centroid is calculated for each group
- The centroid is not set as the target
  - The frontier cell closest to the calculated centroid becomes the target
- Centroids are weighted by distance from the robot
  - Closer centroids are better
- Best centroid is sent to path planner, other centroids are sent to path planner as a Path message

AMCL

- ROS Global Localization Service
  - Used to evenly distribute the particles across the maze before localization starts
- Turning while localizing
  - Speeds up localization
- Localized
  - Uses the calculated determinant from amcl pose to determine if it is localized
  - Stop spinning
  - Publish goal (received from Rviz)
  - Publish amcl pose (used instead of odometry for driving)
- Navigation
  - Uses pure pursuit
    - If a wall is seen, the path is recalculated to the same point that was given in Rviz


File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30
Default Light
Global Status: Ok
Fixed Frame OK
Grid
Status: Ok
Reference Frame <Fixed Frame>
Plane Cell Count 10
Normal Cell Co... 0
Cell Size 1
Line Style Lines
Color 160; 160; 164
Alpha 0.5
Plane XY
Offset 0; 0; 0
RobotModel
Status: Ok
Visual Enabled
Collision Enabled
Update Interval 0
Add
Duplicate
Remove
Rename
Views
Type: Orbit (rviz)
Zero
Current V... Orbit (rviz)
Near Cl... 0.01
Invert ...
Target ... <Fixed Frame>
Distance 7.69845
Focal S... 0.05
Focal S...
Yaw 4.73059
Pitch 1.5698
Field o... 0.785398
Focal P... 0.6761; -0.24959; -0....
Save
Remove
Rename
Time
Pause
Synchronization: Off
ROS Time: 1733958164.99
ROS Elapsed: 22.19
Wall Time: 1733958165.02
Wall Elapsed: 22.10
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
31 fps

What could have been done differently?

- Our code is not organized. Copies of functions exist in multiple files.
- Wall sensing prevented the robot from crashing into the beams; however, it caused the undesired consequence of “wiggling” in tight areas. A potential fix for this would be to create an incremental variable that gradually reduces the speed of the robot if the wall is sensed at a further distance, and having the robot only stop and recalculate a path if the wall is sensed within a minimum distance.

Changes since PDR

- Path Planner
  - If the path starts in the c-space it can still find a path
  - If no path is found, creates a new path using a different centroid
  - Changed heuristic weighting
- Pure pursuit
  - Tuning
  - Wall avoidance
  - Path Overshoot
- Centroids
  - Snap to nearest point
- AMCL
  - Ros service to spread particles
  - Direction of turning
  - Version of pure pursuit that handles paths differently

How did the project go?

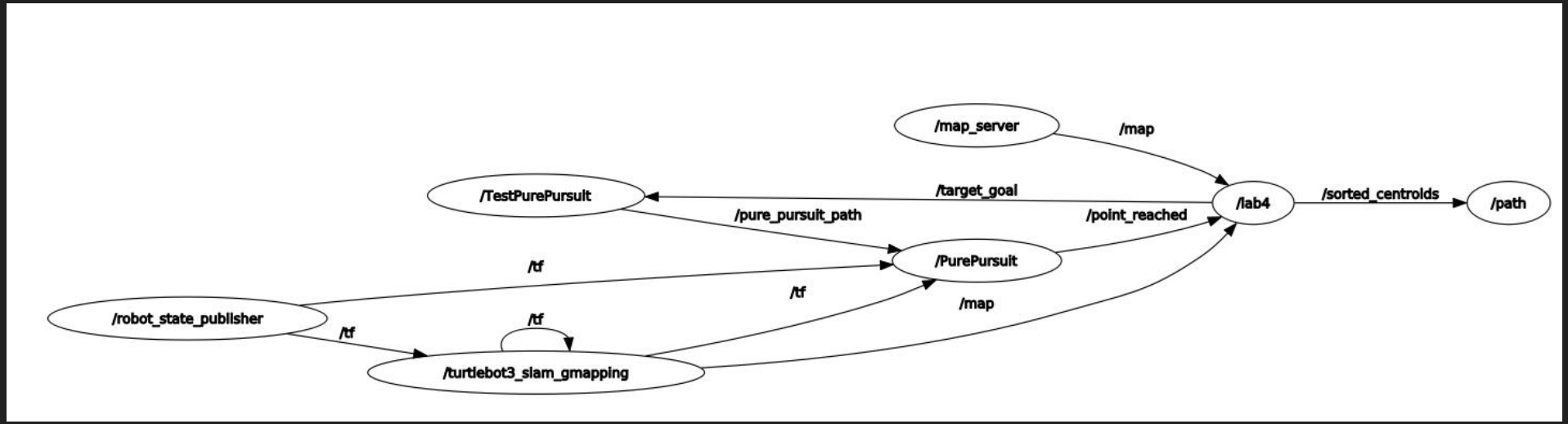
- What went well
  - Team member’s schedules were similar
  - Got full credit for demo and demoed early
  - Well-informed team
  - **SA’s were very helpful**

- What didn’t
  - Too much time spent in the lab
  - Ran into a lot of code issues
  - Reduced to team of 3

Challenges

- AMCL particles didn’t spread out and it remembered its previously calculated pose
- Tuning of pure pursuit
  - The pure pursuit crashed into some of the poles
  - Pure pursuit was slow and got stuck in a loop depending on the path created
  - The robot could get stuck and make no progress in the right spot
- Laser scan data was computationally heavy
- Time commitment

Phase 1/2 - Node Graph



Phase 3 - Node Graph