Independent Study in Robotics:

ORB_SLAM2 integration with Robobulls Robot

Justin Rodney

Preface

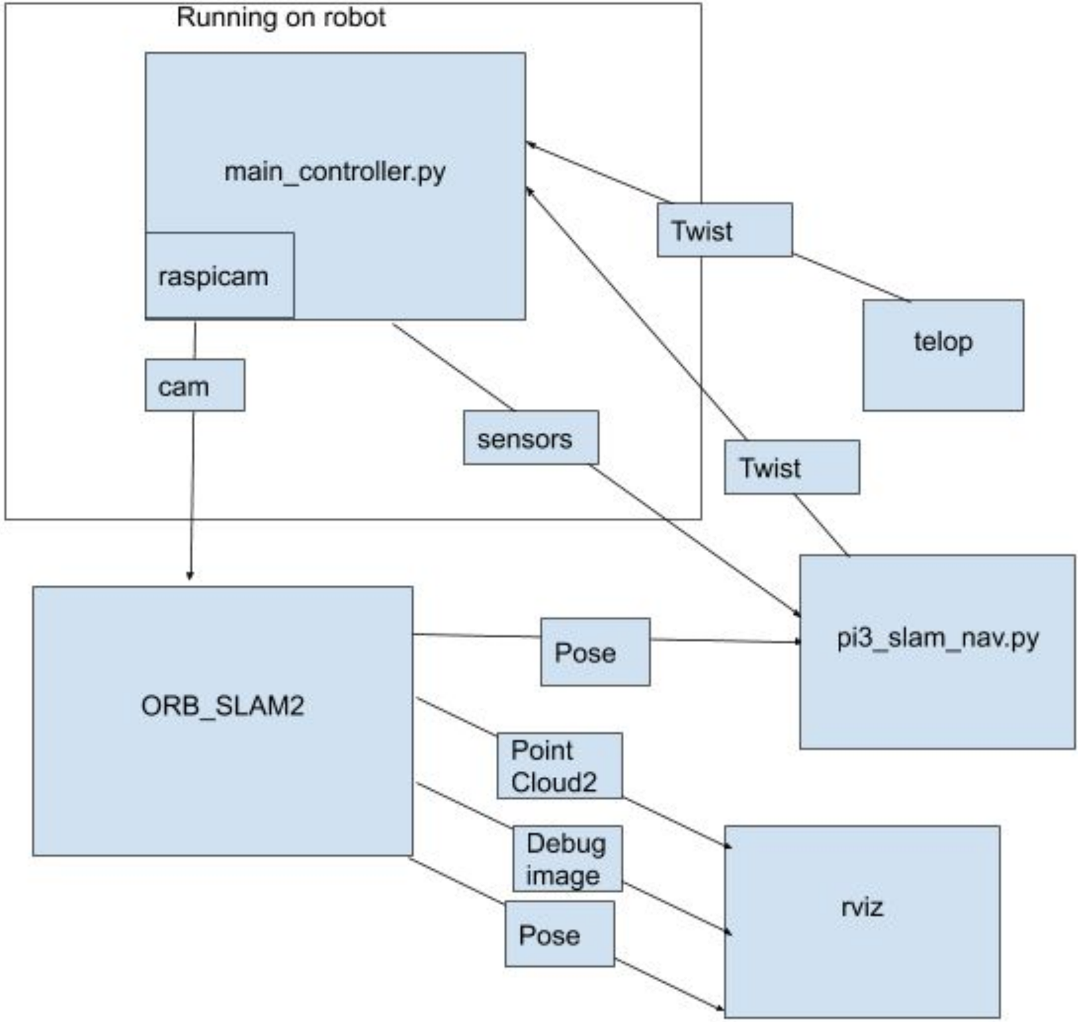
- After initially configuring ORB_SLAM2, how can we use the pose and PointCloud it publishes?
- Ideal applications
    - Autonomous Exploration in an unknown environment to create maps
    - Autonomous Navigation in a known environment via path planning
- ROS packages provide these features, but are incompatible with the current system because the lack of odometry information(*Navigation*, *Exploration_lite*).
- Current application with our robot: bug algorithm using published Pose information

Required Packages

- ❑ ROS Kinetic
- ❑ Existing ROS package on Robot and Client.
- ❑ Rviz*
- ❑ Telop*
- ❑ ORB_SLAM2(ROS implementation)

*included in full installation of ROS Kinetic

Software system



ORB_SLAM2

Is a simultaneous localization and mapping package that takes monocular, stereo, and RGBD camera's as input. Monocular is NOT true scale.

Publishes

- PointCloud2
  - All the identified keypoints in the map
- debug_image
  - Shows the image from the robots camera
  - Identified keypoints marked on the overlay
  - Shows whether the robot is currently localized
- Pose
  - Has the robots x,y, and z coordinates as well as quaternion

Setting up ORB_SLAM2

Use a checkerboard pattern with OpenCV's calibration program. Use output to set parameters in ORB_SLAM2's launch files

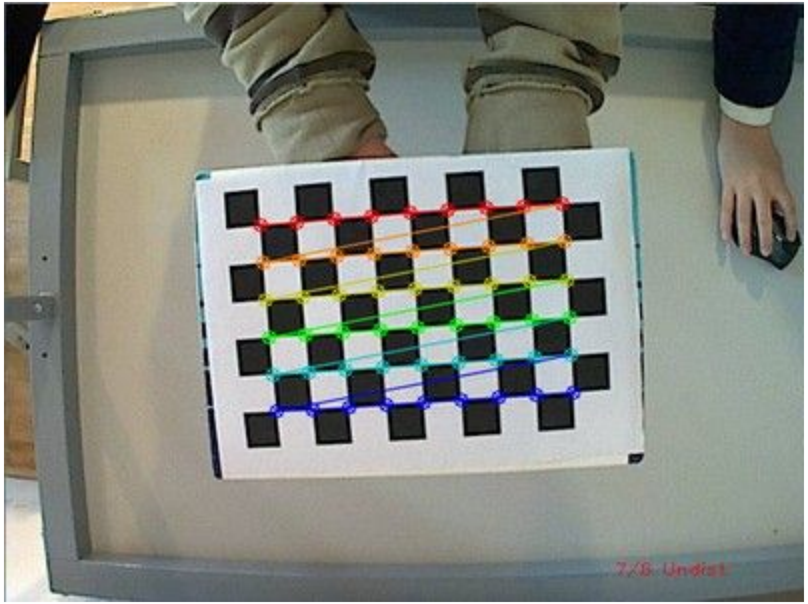
Corrects distortion

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$x_{corrected} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html



Launch File

Important parameters:

- Remap from=
- localize only
- load map/map_file
- Camera calibration parameters

```xml
<launch>
  <node name="orb_slam2_mono" pkg="orb_slam2_ros"
      type="orb_slam2_ros_mono" output="screen">

       <remap from="/cam/image_raw" to="/cam/image" />

       <param name="publish_pointcloud" type="bool" value="true" />
       <param name="publish_pose" type="bool" value="true" />
       <param name="localize_only" type="bool" value="false" />
       <param name="reset_map" type="bool" value="false" />

       <!-- static parameters -->
       <param name="load_map" type="bool" value="false" />
       <param name="map_file" type="string" value="map.bin" />
       <param name="voc_file" type="string" value="$(find orb_slam2_ros)/orb_slam2/Vocabulary/ORBvoc.txt" />

       <param name="pointcloud_frame_id" type="string" value="map" />
       <param name="camera_frame_id" type="string" value="camera_link" />
       <param name="min_num_kf_in_map" type="int" value="5" />

       <!-- ORB parameters -->
       <param name="/ORBextractor/nFeatures" type="int" value="2000" />
       <param name="/ORBextractor/scaleFactor" type="double" value="1.2" />
       <param name="/ORBextractor/nLevels" type="int" value="8" />
       <param name="/ORBextractor/iniThFAST" type="int" value="20" />
       <param name="/ORBextractor/minThFAST" type="int" value="7" />
       <!-- Camera parameters -->
       <!-- Camera frames per second -->
       <param name="camera_fps" type="int" value="30" />
       <!-- Color order of the images (0: BGR, 1: RGB. It is ignored if images are grayscale) -->
       <param name="camera_rgb_encoding" type="bool" value="true" />

        <!-- Camera calibration parameters -->
        <!--If the node should wait for a camera_info topic to take the camera calibration data-->
       <param name="load_calibration_from_cam" type="bool" value="false" />
       <!-- Camera calibration and distortion parameters (OpenCV) -->
      <param name="camera_fx" type="double" value="583.01740710559432" />
      <param name="camera_fy" type="double" value="583.01740710559432" />
      <param name="camera_cx" type="double" value="320" />
      <param name="camera_cy" type="double" value="240" />
       <!-- Camera calibration and distortion parameters (OpenCV) -->
      <param name="camera_k1" type="double" value="0.20483552665926258" />
      <param name="camera_k2" type="double" value="-0.42414204428032326" />
      <param name="camera_p1" type="double" value="0.0" />
      <param name="camera_p2" type="double" value="0.0" />
      <param name="camera_k3" type="double" value=".0095177708663656737" />
  </node>
</launch>
```

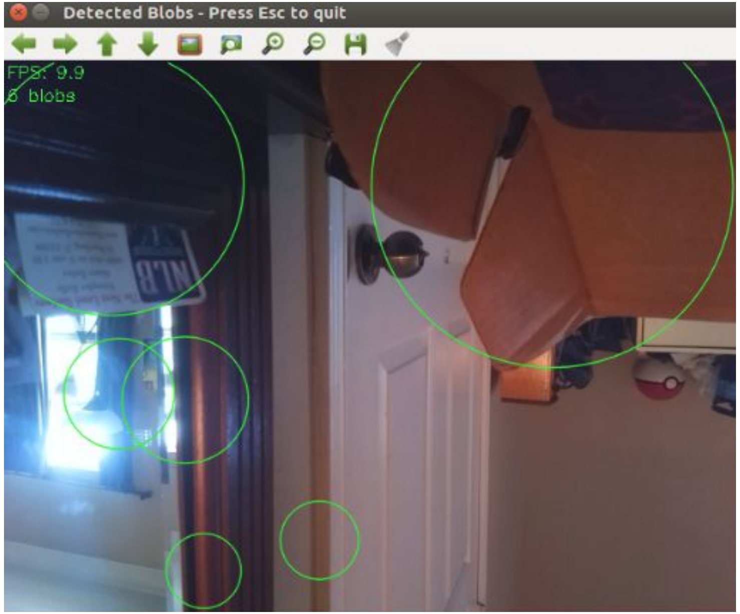
*from blob detection program, not ORB_SLAM2. ORB_SLAM2 is more demanding

FPS performance

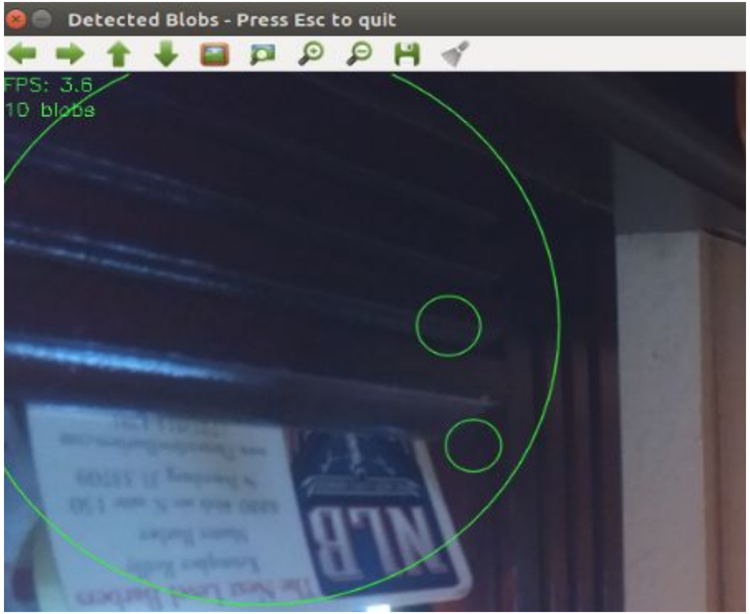
Tested performance:
On 3 wifi networks
With ROScore running on robot and client computer

- ❑ Lower FPS reduces localization

640x480

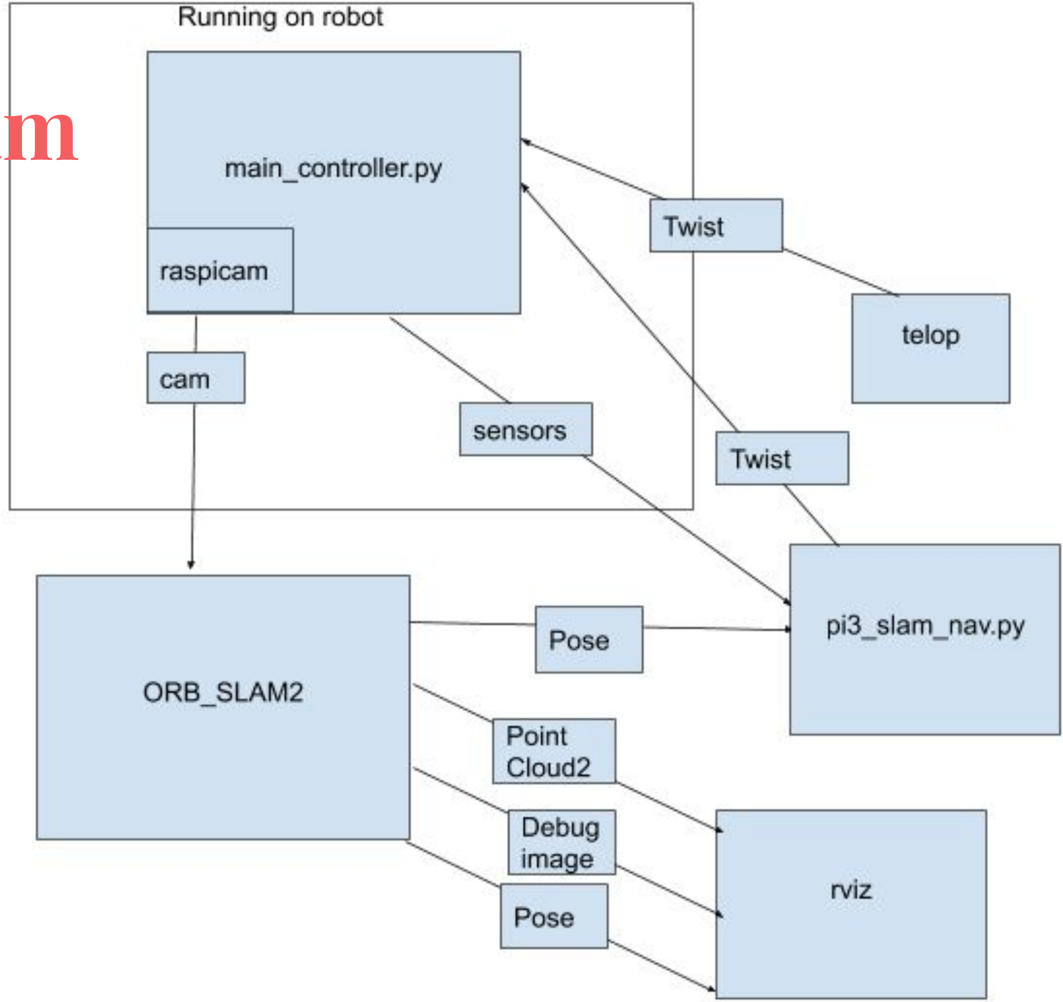


1280x960

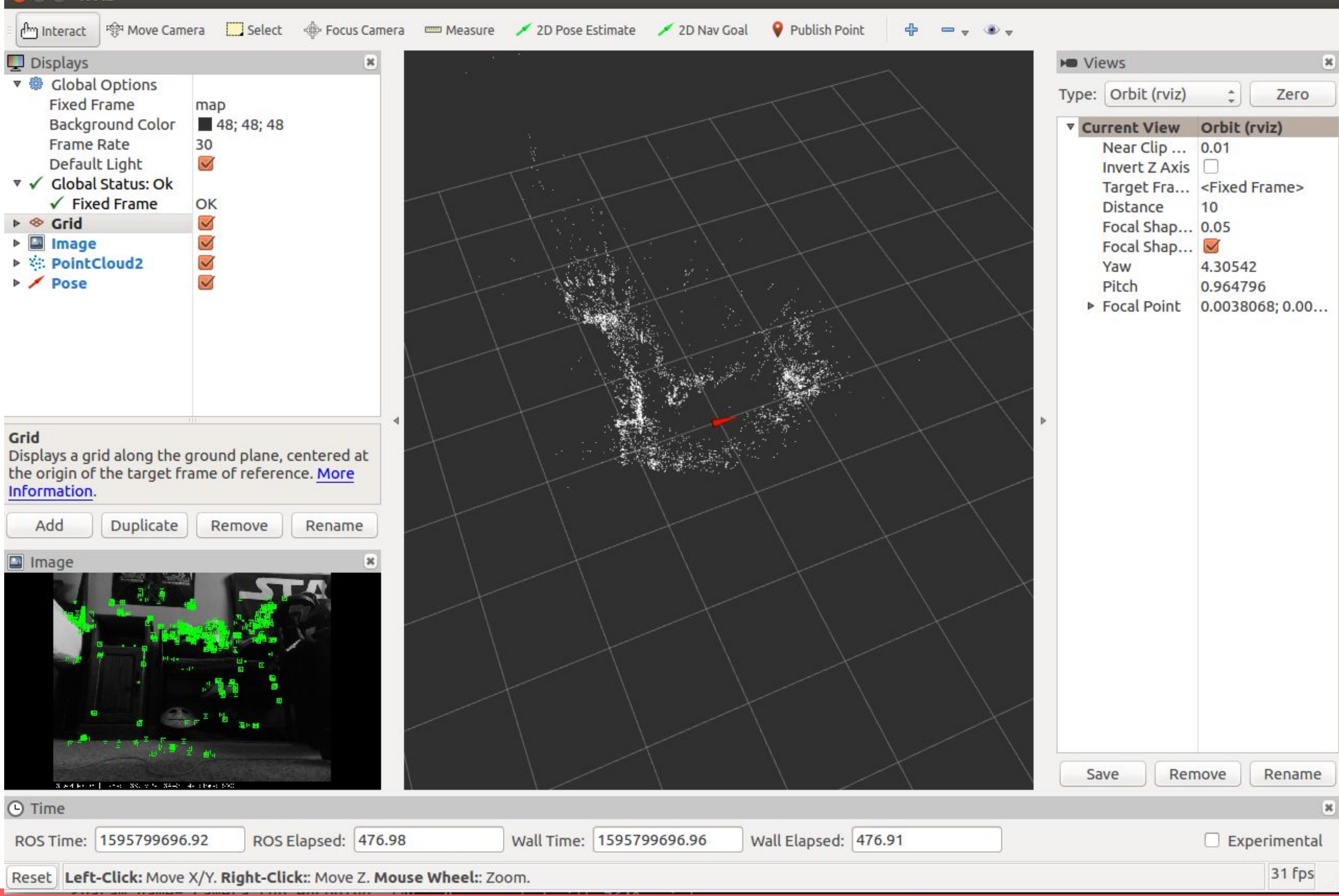


Running The Program

1. Start Rviz
2. Start ORB_SLAM2
3. Start telop to control robot to create map
4. Start pi3_slam_nav.py



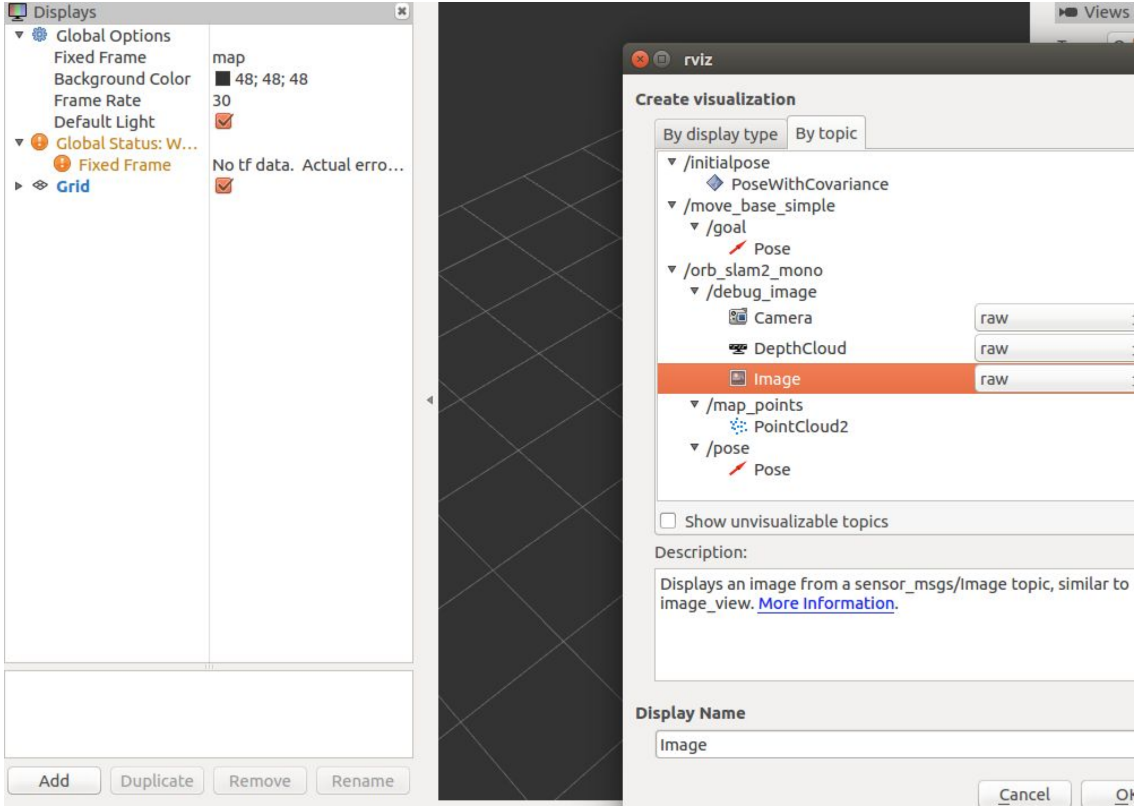
Rviz



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
Image
PointCloud2
Pose
Grid
Displays a grid along the ground plane, centered at the origin of the target frame of reference. More Information.
Add
Duplicate
Remove
Rename
Image
Views
Type:
Orbit (rviz)
Zero
Current View
Orbit (rviz)
Near Clip ...
0.01
Invert Z Axis
Target Fra...
<Fixed Frame>
Distance
10
Focal Shap...
0.05
Focal Shap...
Yaw
4.30542
Pitch
0.964796
Focal Point
0.0038068; 0.00...
Save
Remove
Rename
Time
ROS Time:
1595799696.92
ROS Elapsed:
476.98
Wall Time:
1595799696.96
Wall Elapsed:
476.91
Experimental
Reset
Left-Click: Move X/Y. Right-Click:: Move Z. Mouse Wheel:: Zoom.
31 fps

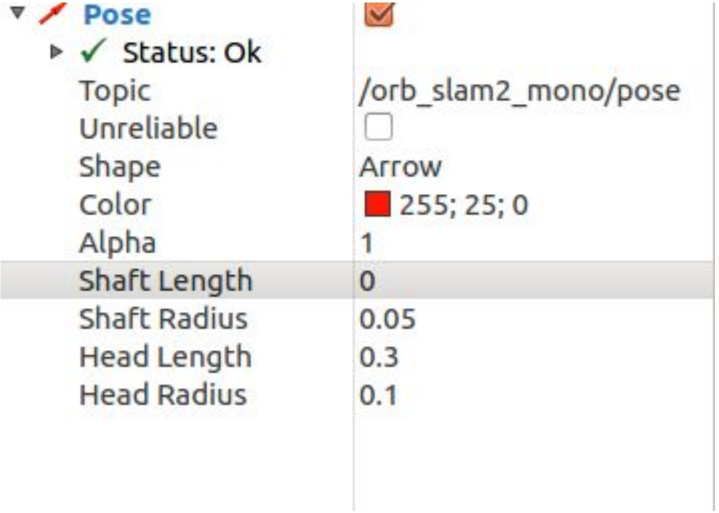
Setting Rviz Parameter #1

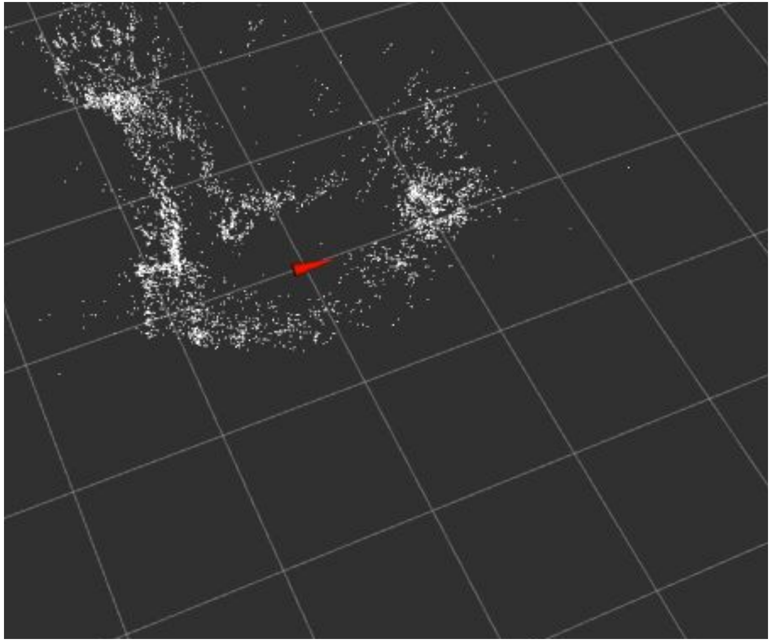
- Debug_image
- Map_points
- pose



Setting Rviz Parameter #2

- Set Shaft Length= 0





Telop

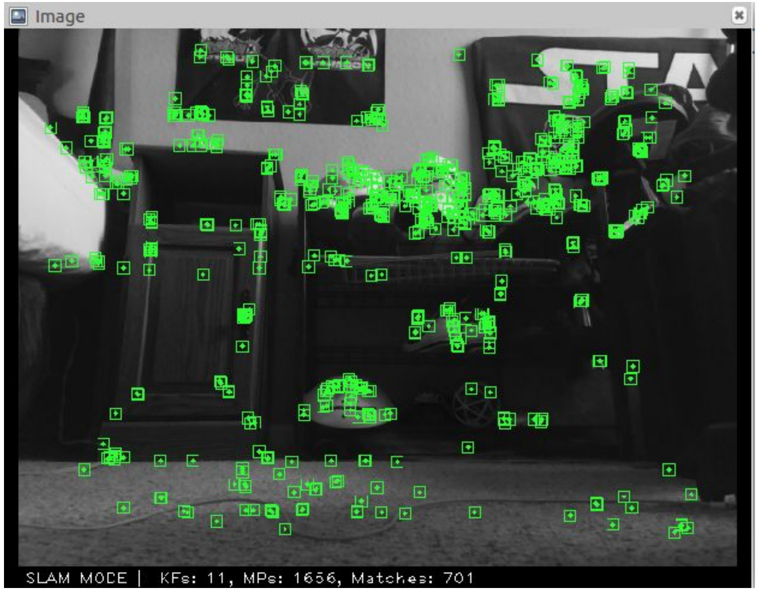
rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/pi3_robot_2019/r1/speed_vw

```
justinrodney@justinrodney-MacBookPro: ~/catkin_ws

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 2.29748649318     turn 4.59497298636
currently:      speed 2.5272351425      turn 5.05447028499
currently:      speed 2.77995865675     turn 5.55991731349
currently:      speed 3.05795452242     turn 6.11590904484
currently:      speed 3.36374997466     turn 6.72749994933
```

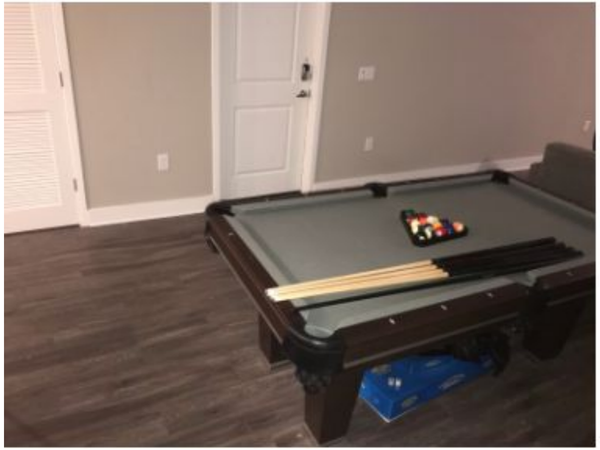
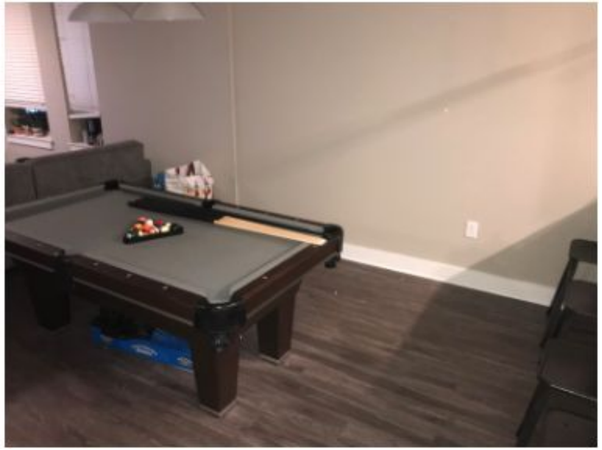
Point Generation

Does not work on objects that are too dark or too light

Also found in Yu-Ting Chung’s study on using raspberry pi with ORB_SLAM2

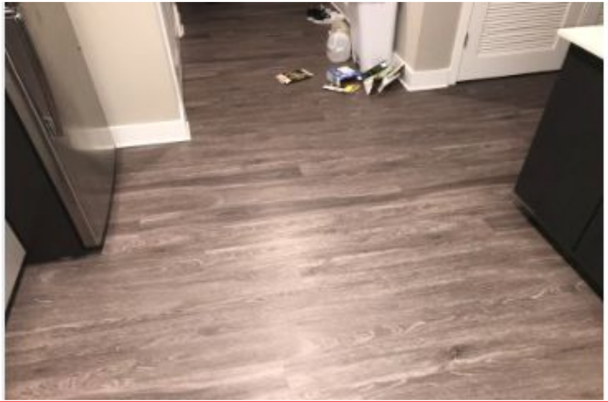


Environment 1 & 2

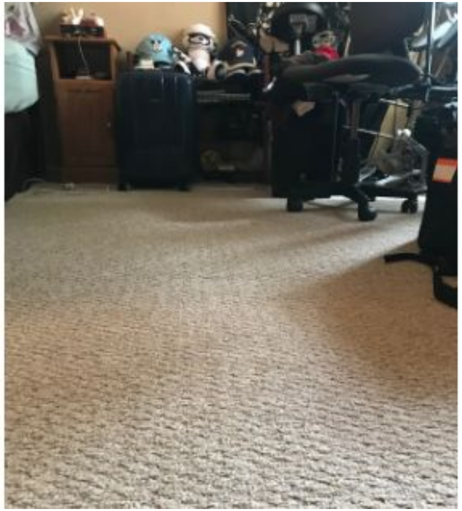
Living area 1

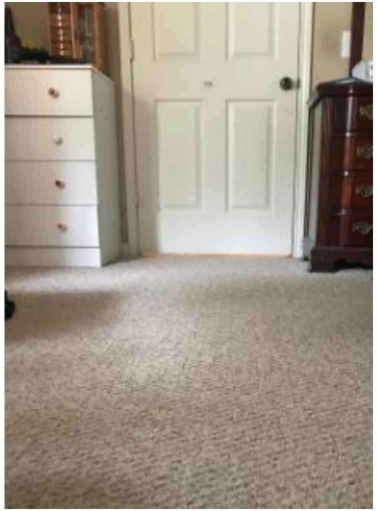
Kitchen

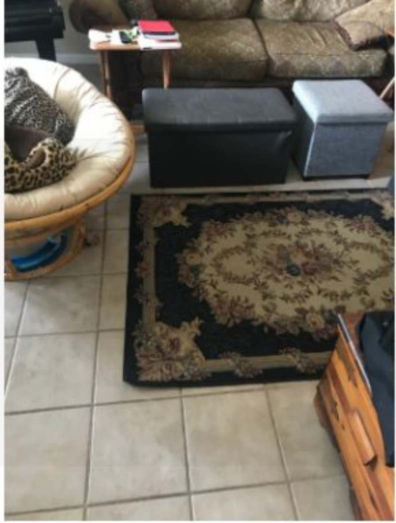



Environment 3 & 4

Bedroom







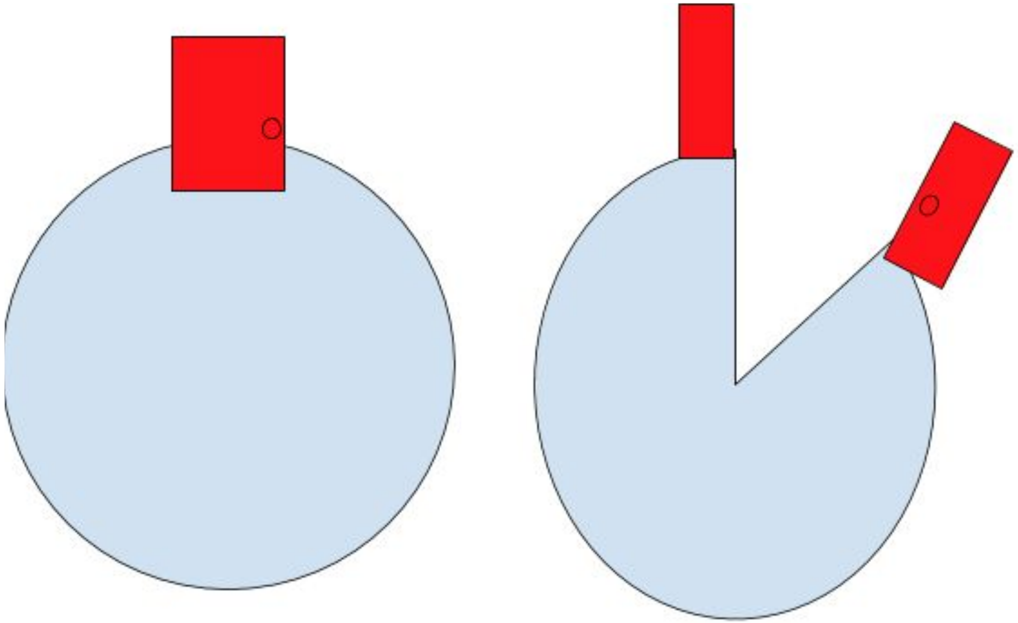
Living area 2

Mapping Methods tested

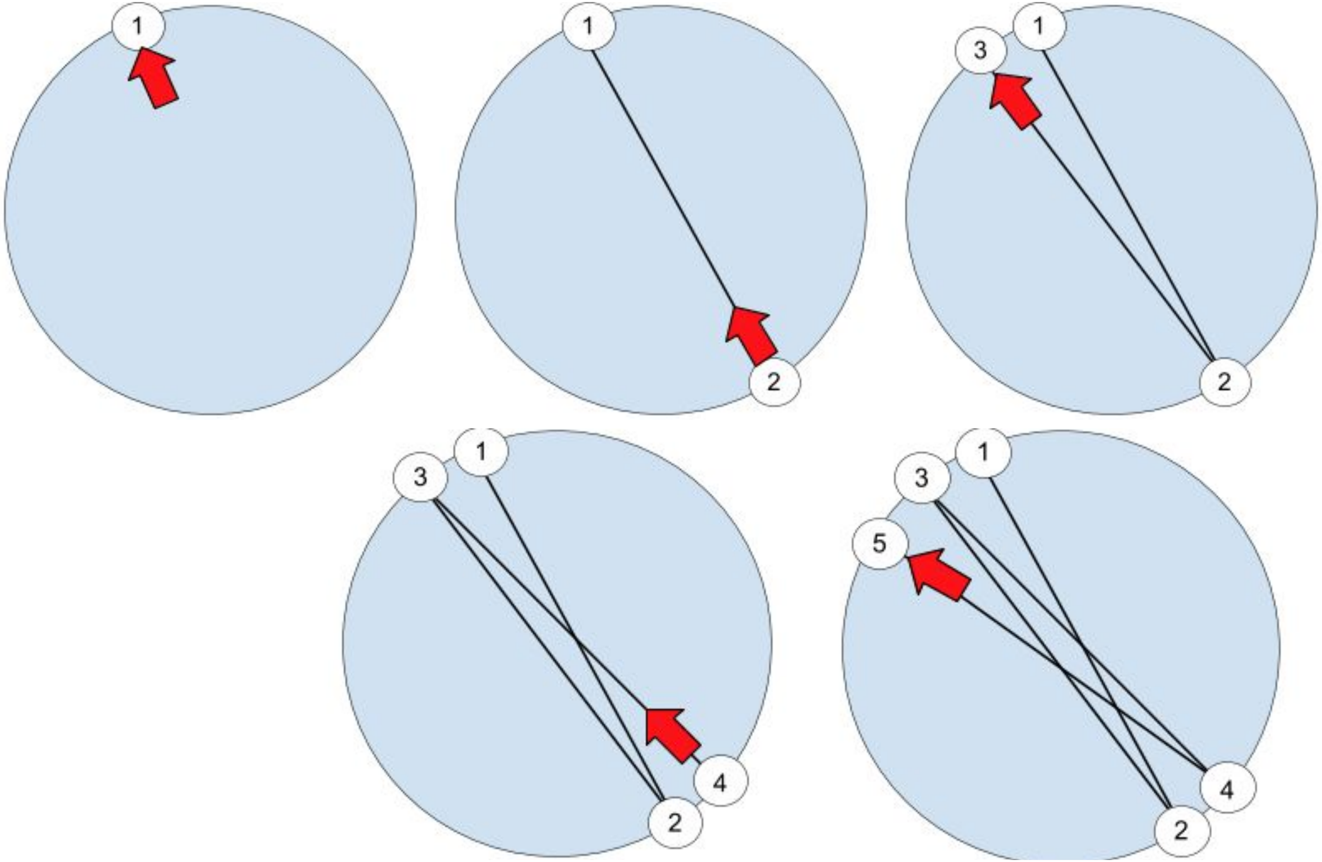
- Wall following
- Spinning the robot in a circle in the middle of the room
- Manual Mapping

Skewed map

If the robot was inside of a circle with a red door, instead of capturing the full 360 degrees of the environment, the map would have a portion of the map left blank with the rest of the environment skewed.



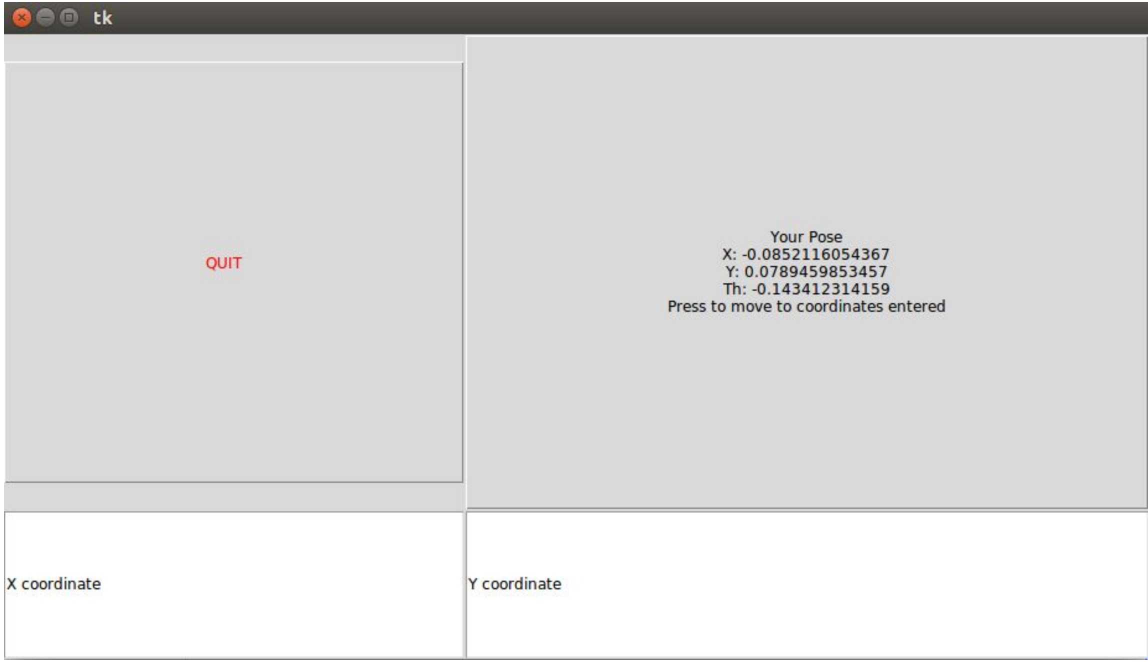
Mapping Motion



pi3_slam_nav

Uses the x, y, and yaw and the x and y coordinates of the goal to calculate the required yaw for the robot to be facing the goal.

Uses wall following for obstacle avoidance



Desired angle = atan2(goalY- currentY, goalX-currentX)
Distance = (goalX-currentX)^2 + (goalY-currentY)^2

The downfalls

- <10 FPS even when using the lowest usable resolution(640x480)
- Depth is approximated when the map is initialized. Cannot be converted to units of measurements.
- RGB processing cannot be used without RGBD capable cameras(stereo).
- Incompatible with Navigation(used with many ROS packages including autonomous navigation viapath planning) and Exploration_lite(autonomous exploration).
    - ORB_SLAM2 does not project proper transforms necessary for these packages

Why use ORB_SLAM2?

- ORB_SLAM2 is the only ROS package that is capable of SLAM using a mono-camera alone.
- Other packages require IMUs, laser scanners, and stereo or RBGD cameras.

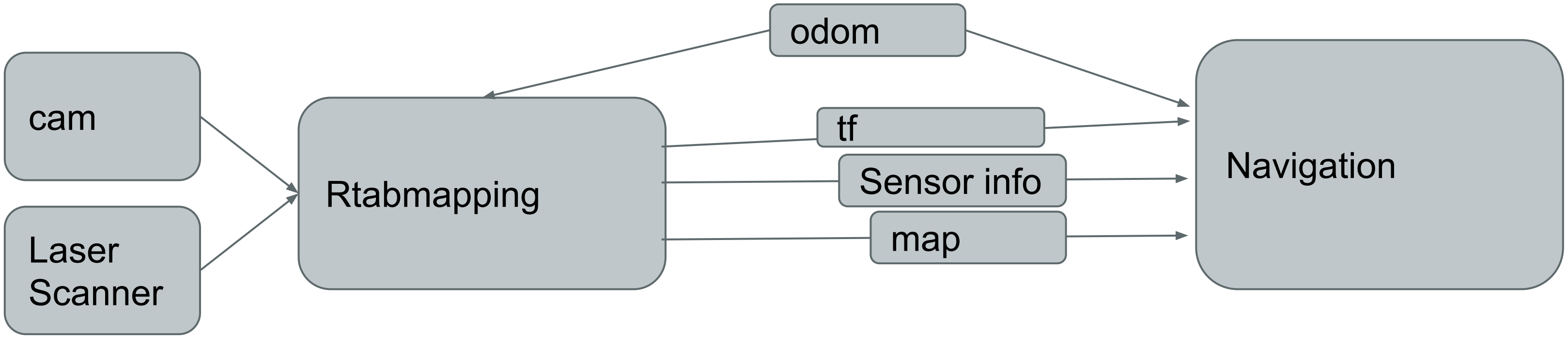
Octomap

- Creates an occupancy grid from PointCloud2 topic from ORB_SLAM2
- Unfortunately, unreliable in conjunction with ORB_SLAM2.
- Further refinement of parameters could create reliable maps for path_planning

Benefits of Upgrading the Robot

- Rtabmapping
  - Gives proper transform(tf) tree from map to odometry
  - Requires a laser scanner, but produces true to scale maps
  - Creates an occupancy grid
  - Can recognize specific objects in an environment

Allows the use of navigation and exploration_lite



Conclusion

Current system:

- ❏ Provides pose information inside a depth approximated grid
- ❏ Can be used for bug algorithms and other simple applications

Having better sensors would allow us to:

- ❏ Create true to scale maps
- ❏ Laser scanners/stereo cameras perform better
- ❏ Wider compatibility with other ROS packages

Works Referenced

Camera Calibration. (n.d.). Retrieved July 26, 2020, from https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

Chung, Y. (2018, July 18). Final Report. Retrieved July 30, 2020, from https://github.com/biorobaw/SLAM-S2018/blob/master/docs/Reports/final%20report.pdf

Gmapping. (n.d.). Retrieved July 26, 2020, from http://wiki.ros.org/geometry_msgs

Gmapping. (n.d.). Retrieved July 26, 2020, from http://wiki.ros.org/gmapping

Navigation. (n.d.). Retrieved July 26, 2020, from http://wiki.ros.org/navigation

Octomap. (n.d.). Retrieved July 26, 2020, from http://wiki.ros.org/octomap

Orb_slam2_ros. (n.d.). Retrieved July 26, 2020, from http://wiki.ros.org/orb_slam2_ros

Rtabmap_ros. (n.d.). Retrieved July 26, 2020, from http://wiki.ros.org/rtabmap_ros