

ROS

Overview & Simulation

Dr. Juan Rojas

Sun Yat Sen University

School of Software

中山大学

软件学院

Motivation

Scalability

Debugging

Encapsulation

Simulation

Fault Tolerance

Open Source

State Machine



History



STAIR: STanford Artificial Intelligence Robot

Artificial Intelligence Laboratory, Computer Science Department, Stanford University

 **ROS.org**

Applications Overview

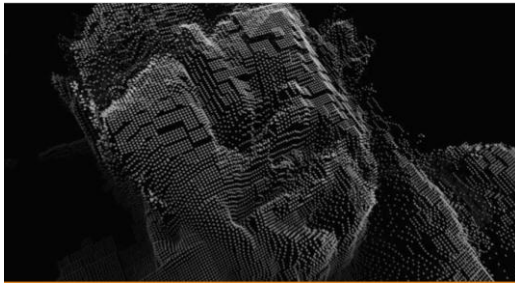
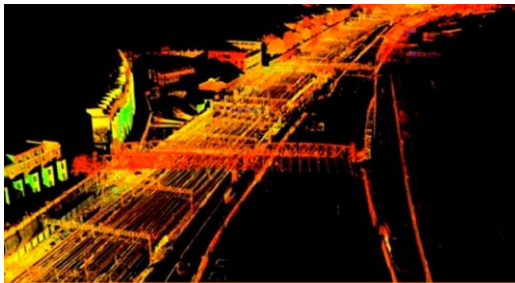


Image Processing



Controlling a Mobile Robot



SLAM and Navigation



Combining Vision and Navigation



Manipulation

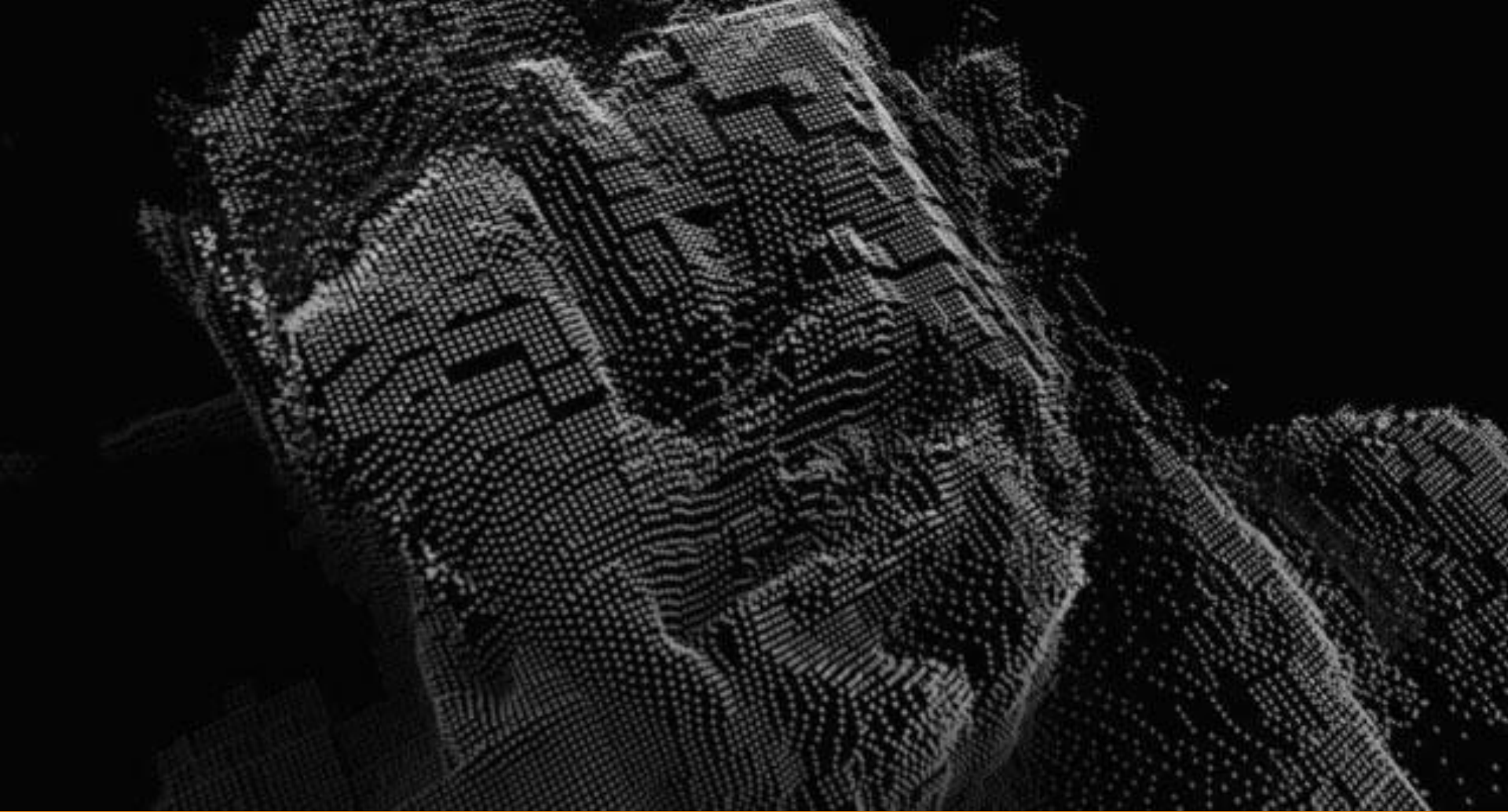


Image Processing

OpenCV Overview

OpenCV Overview: > 500 functions

opencv.willowgarage.com

Robot support



General Image Processing Functions



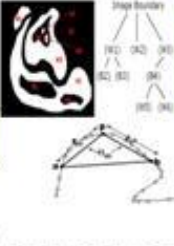
Segmentation



Transforms



Machine Learning: • Detection, • Recognition



Geometric descriptors



Features



Tracking



Matrix Math

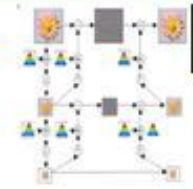
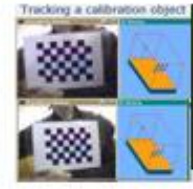
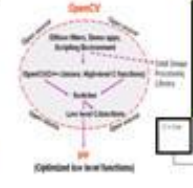


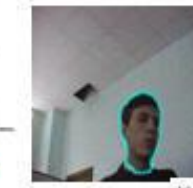
Image Pyramids



Camera calibration, Stereo, 3D



Utilities and Data Structures



Fitting

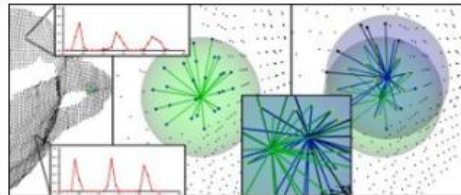


OpenPCL

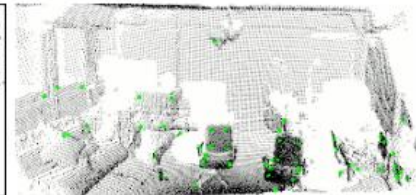
filters



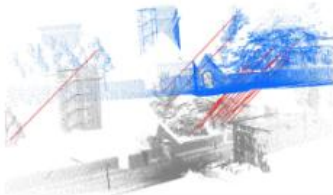
features



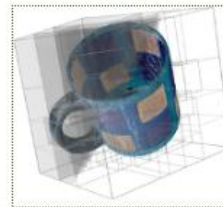
keypoints



registration



kdtree



octree



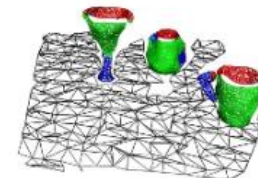
segmentation



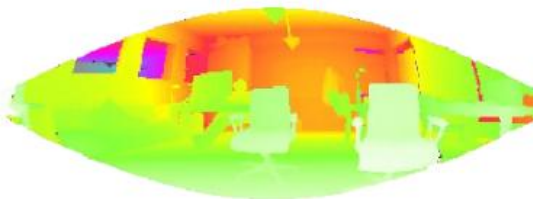
sample_consensus



surface



range_image



io



visualization



Edge Detection



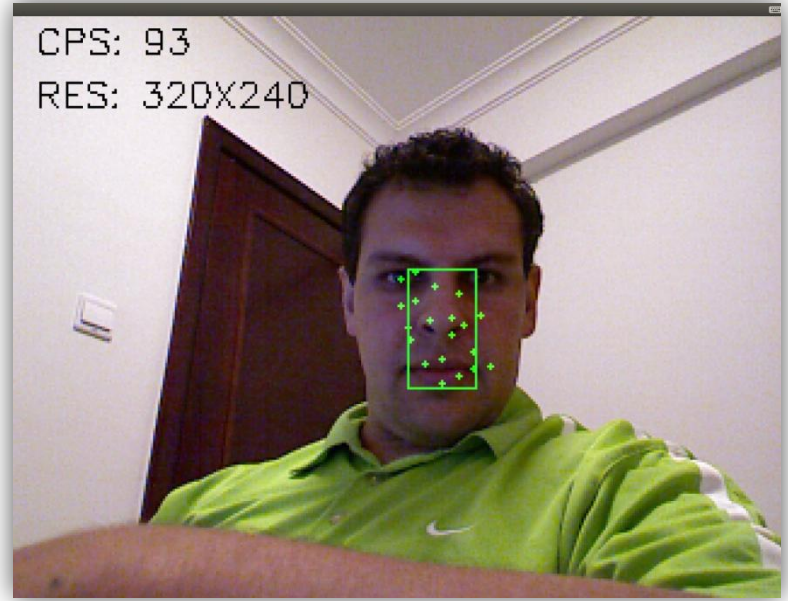
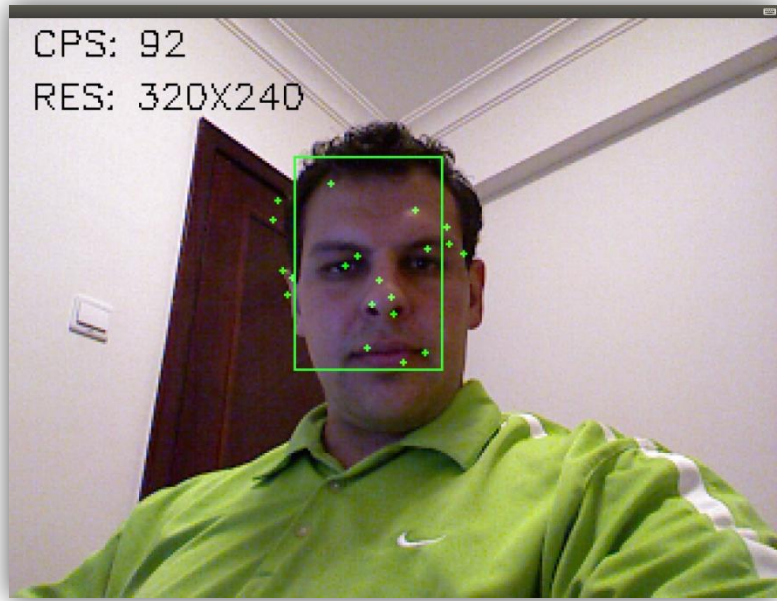
- Kinect Drivers:

```
roslaunch rbx1_vision openni_node.launch
```

- Edge Detection:

```
roslaunch rbx1_vision cv_bridge_demo.py
```


Advanced Image Processing



Face Features & Key-Point Detection & Optical Flow

- Kinect: launch the OpenNI Driver:

```
roslaunch rbx1_vision openni_node.launch
```

- Run the face tracker:

```
roslaunch rbx1_vision face_tracker2.launch
```

Face Velocity Tracking

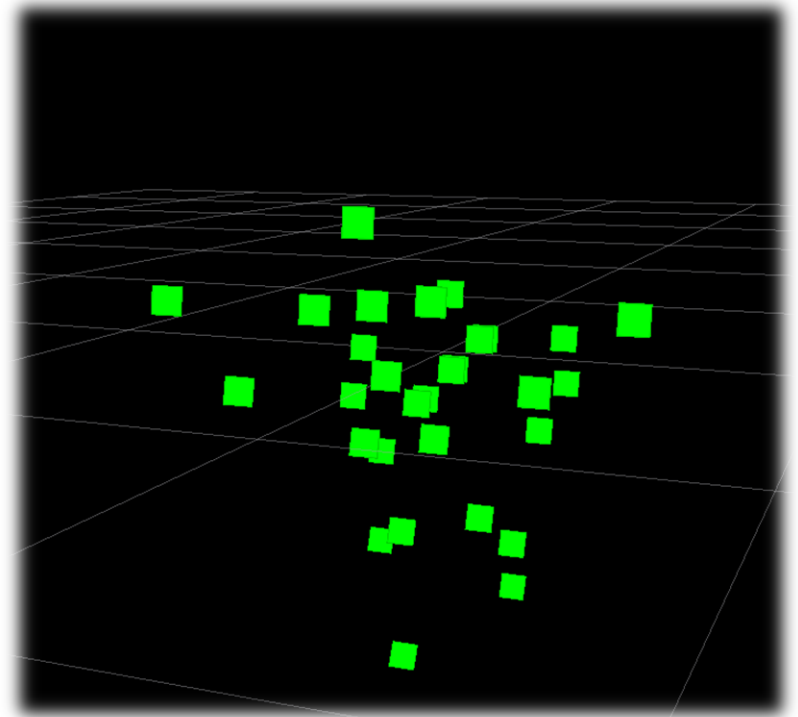
- Online Face Tracker Analysis
 1. Launch: Face tracker.
 2. Launch: ROI tracker
 3. Plot: Results in rxplot

```
roslaunch rbx1_apps object_tracker.launch
```

```
rxplot -p 15 /cmd_vel/angular/z
```

Skeleton Markers

- Follow Body Motion.
- Useful:
Learning from Demonstration.



```
roslaunch rbx1_vision openni_node.launch
```

```
roslaunch skeleton_markers markers_from_tf_groovy.launch
```

```
roslaunch rviz rviz -d `rospack find \skeleton_markers`/markers_from_tf_groovy.rviz
```

Face Velocity Tracking

Online Face Tracker Analysis

1. Launch: Face tracker.
2. Launch: ROI tracker
3. Plot: Results in rxplot

```
roslaunch rbx1_apps object_tracker.launch
```

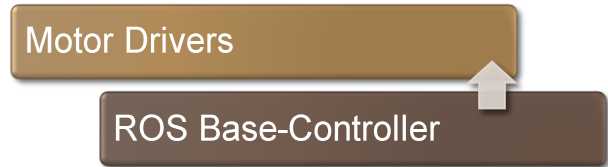
```
rxplot -p 15 /cmd_vel/angular/z
```



Controlling a Mobile Robot

Teleoperating your Robot

- ROS move-base controller subscribes to `/cmd_vel` and tells drivers how to move.



```
roslaunch rbx1_bringup fake_pi_robot.launch
```

```
roslaunch rbx1_nav sim.launch
```

- Teleoperate with keyboard, joystick, GUI.

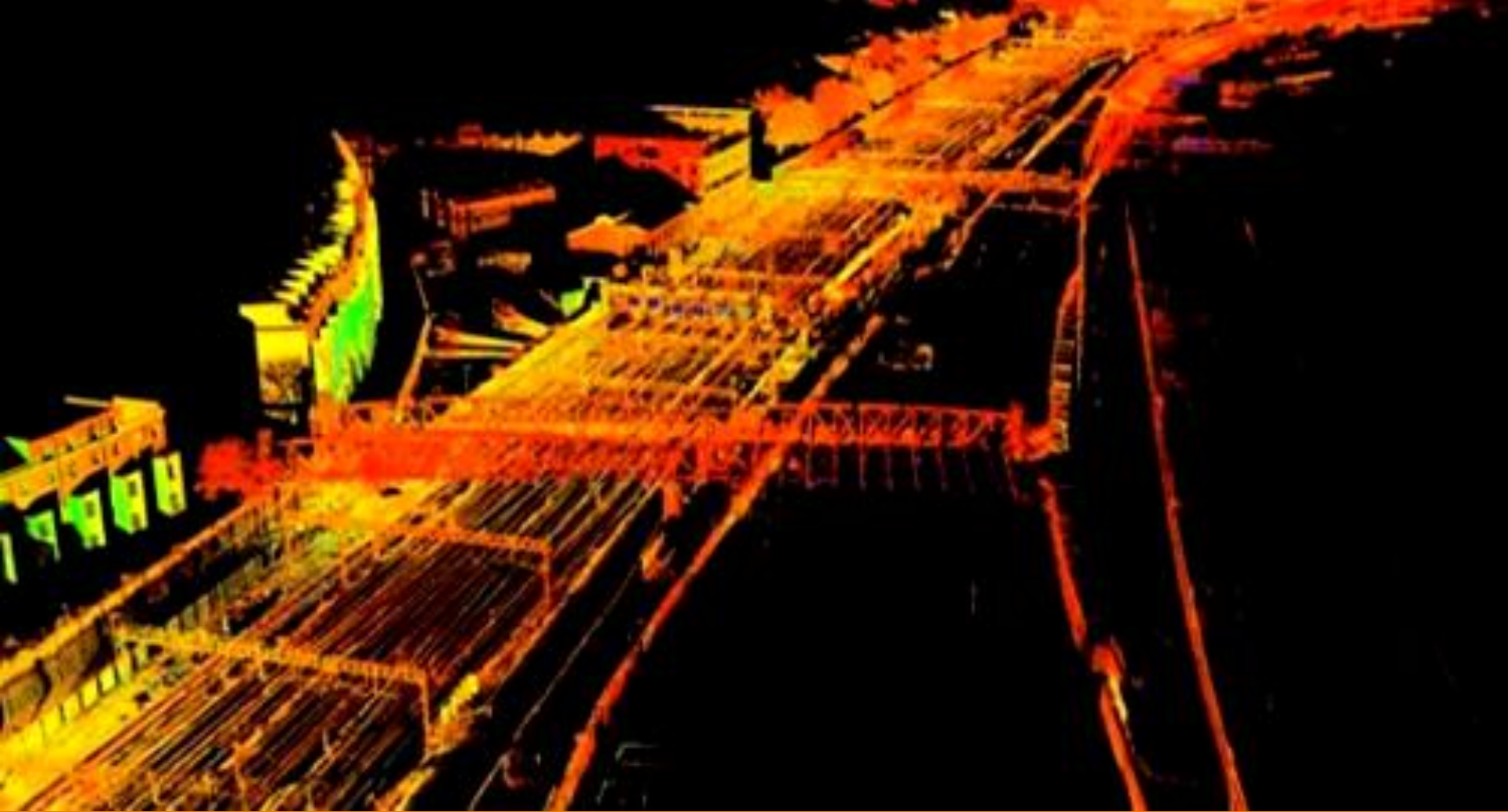
```
Keyboard:    roslaunch rbx1_nav keyboard_teleop.launch
```

```
GUI:         arbotix_gui
```

Analysis

The screenshot displays the ArbotIX Controller GUI with three main components:

- Terminal (Left):** Shows a sequence of position and orientation data points for a trajectory. The position (x, y, z) and orientation (x, y, z, w) values change across several steps, indicating a path in a 2D plane.
- RXPlot (Center):** A graph showing a red trajectory line. The vertical axis represents height (z) from 1.1 to 1.5, and the horizontal axis represents time from 42 to 56. The trajectory starts at a height of 1.5, drops to approximately 1.27, rises to 1.42, drops to 1.21, rises to 1.31, and finally drops to 1.11.
- Move Base (Top Right):** A window showing a top-down view of the robot's base with a red dot indicating its current position.
- Move Servos (Right):** A list of servos with checkboxes and sliders for control:
 - arm_elbow_flex: slider at 0
 - arm_shoulder_lift: slider at 0
 - arm_shoulder_pan: slider at 0
 - arm_wrist_flex: slider at 0
 - gripper: slider at 0
 - head_pan: slider at 0
 - head_tilt: slider at 0
- Robot View (Bottom Right):** A top-down view of the robot's head and gripper, with a yellow arrow pointing downwards from the gripper.



SLAM and Navigation

Navigation

1. Move to goal with empty map

```
roslaunch rbx1_bringup fake_turtlebot.launch
```

2. Move-Base Controller.

```
roslaunch rbx1_nav fake_move_base_blank_map.launch
```

3. See Simulation

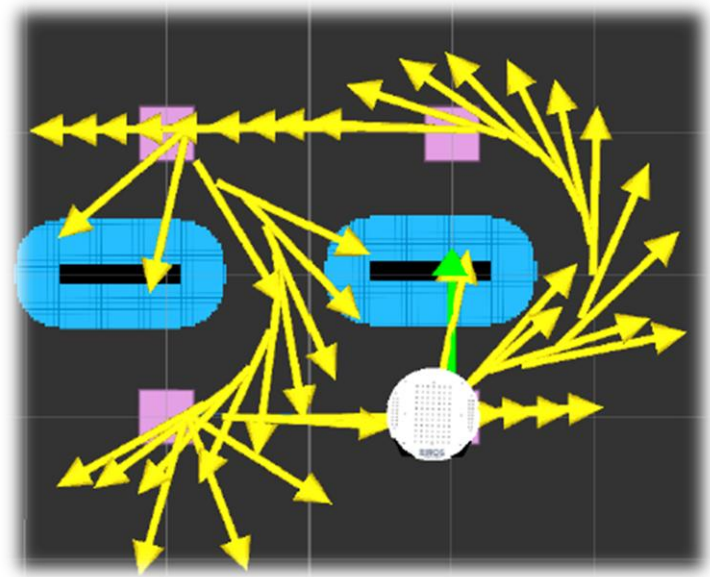
```
roslaunch rbx1_nav nav.rviz
```

4. Pass Commands

```
rostopic pub -1 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.2, y: 0, z: 0},  
angular: {x: 0, y: 0, z: 0}}'; rostopic pub -r 10 /cmd_vel geometry_msgs/Twist  
'{linear: {x: 0.2, y: 0, z: 0}, angular: {x: 0, y: 0, z: 0.5}}'
```

Collision Avoidance

- Costmaps recomputed to find optimal path.



```
roslaunch rbx1_bringup fake_turtlebot.launch
```

```
roslaunch rbx1_nav fake_move_base_obstacle.launch
```

```
roslaunch rbx1_nav fake_move_base_obstacle.launch
```

```
roslaunch rbx1_nav fake_move_base_obstacle.launch
```


8.2 Varying Navigation Parameters

- Online Parameter Adjustment

The screenshot shows a web browser window with a title bar at the top displaying system icons and the time 8:14 AM. The main content area is titled "more_user_trajectory_parameters" and contains a list of 20 parameters. Each parameter is represented by a row with a name, a slider control, a numerical range, and a text input field. The sliders are positioned to show the current value of each parameter. The text input fields contain the current values, some of which are highlighted in blue.

Parameter Name	Slider Range	Current Value
acc_lim_x	0.0 - 20.0	2.5
acc_lim_y	0.0 - 20.0	2.5
acc_lim_theta	0.0 - 20.0	3.2
max_vel_x	0.0 - 20.0	0.5
min_vel_x	0.0 - 20.0	0.1
max_vel_theta	0.0 - 20.0	1.0
min_vel_theta	-20.0 - 0.0	-1.0
min_in_place_vel_theta	0.0 - 20.0	0.4
sim_time	0.0 - 10.0	1.0
sim_granularity	0.0 - 5.0	0.025
angular_sim_granularity	0.0 - 1.57079632679	0.025
pdist_scale	0.0 - 5.0	0.8
gdist_scale	0.0 - 5.0	0.6
occdist_scale	0.0 - 5.0	0.1
oscillation_reset_dist	0.0 - 5.0	0.05
escape_reset_dist	0.0 - 5.0	0.1
escape_reset_theta	0.0 - 5.0	079632679
vx_samples	1 - 300	20
vtheta_samples	1 - 300	20

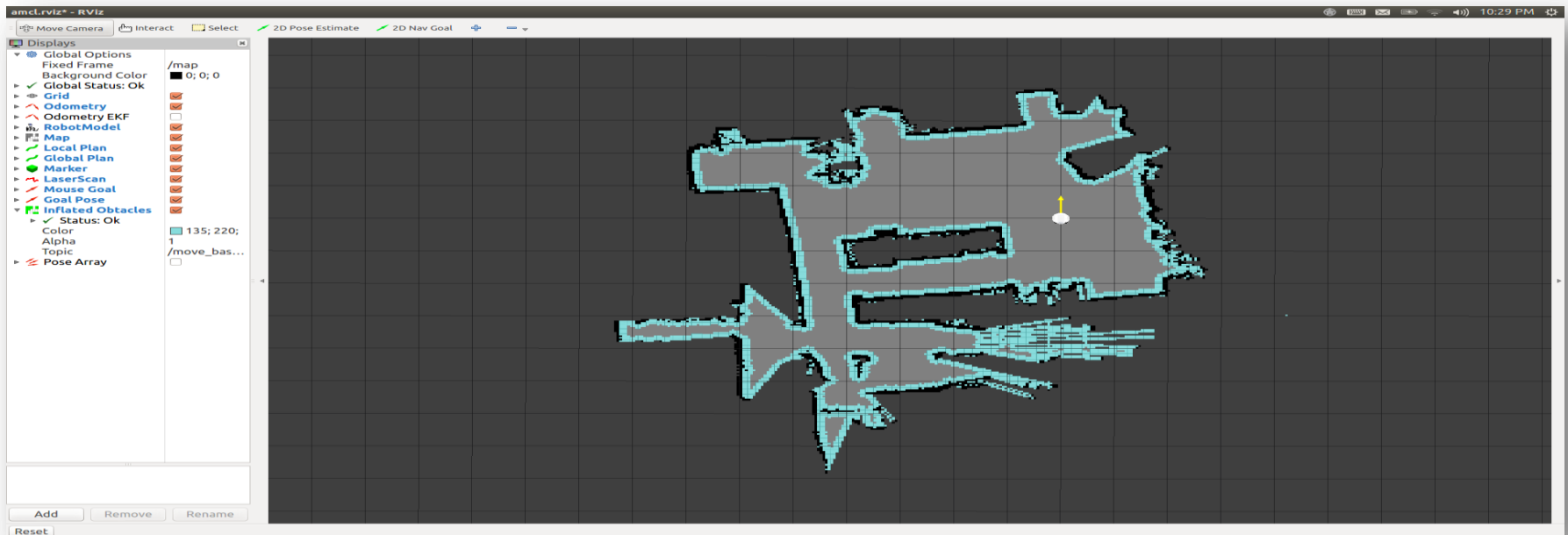
SLAM & Navigation

- Map. Localize. Motion Planning.

```
roslaunch rbx1_bringup fake_turtlebot.launch
```

```
roslaunch rbx1_nav fake_amcl.launch map:=test_map.yaml
```

```
roslaunch rviz rviz -d `rospack find rbx1_nav`/amcl.rviz
```





Combining Vision and Navigation

Person Follower

- Turtlebot_follower node
 - Does not recognize person
 - Looks for moving objects within a certain distance
 - Tells robot how close to follow
 - Program commands robot to move so as to follow blob

```
roslaunch rbx1_vision openni_node.launch
```

```
roslaunch rbx1_apps follower.launch
```

```
roslaunch rbx1_bringup fake_turtlebot.launch
```

```
roslaunch rviz rviz -d `rospack find rbx1_nav`/sim.rviz
```



Manipulation

PR2 Manipulation Simulation

- Solve Kinematics and Dynamics Easily
- Avoid Collisions
- Easy GUI
- Planning & Executing Motions
- Goal Acquisition through Sensor Integration

- Launch the PR2 in Gazebo

```
roslaunch pr2_gazebo pr2_empty_world.launch
```

- Launch the move it plan and execute file:

```
roslaunch pr2_moveit_config moveit_planning_execution.launch
```