Navigation Tutorial

Eitan Marder-Eppstein

Willow Garage

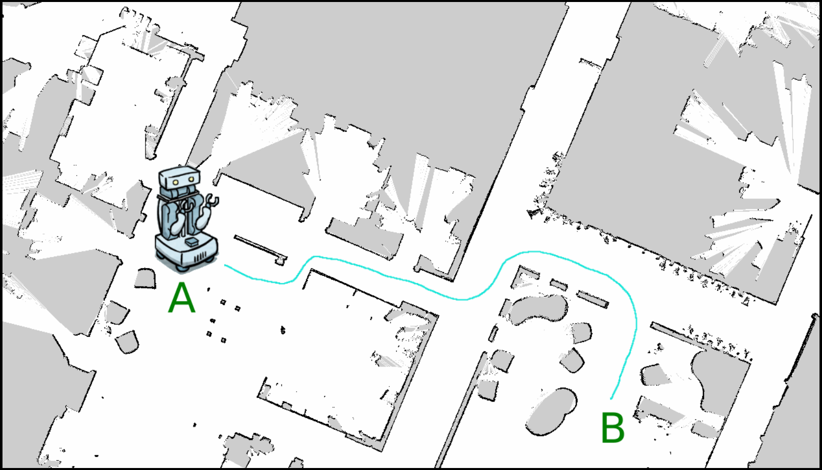
http://www.ros.org/wiki/navigation

October 25, 2010

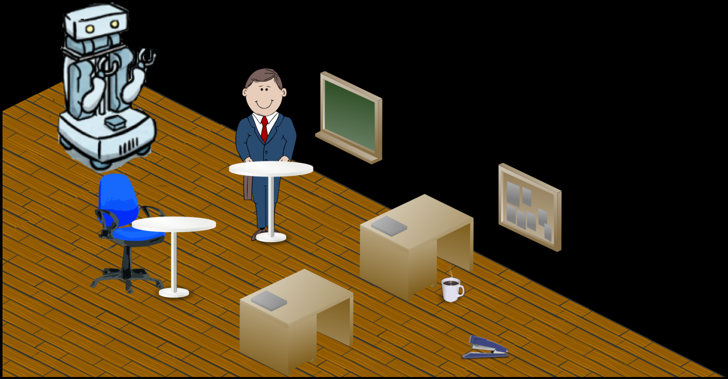
Outline

- Brief overview of navigation
- Run navigation with SLAM to build a map
- Send goals to the navigation stack through code
- Learn how to save a map and use it for navigation later

Overview



A Typical Office Environment

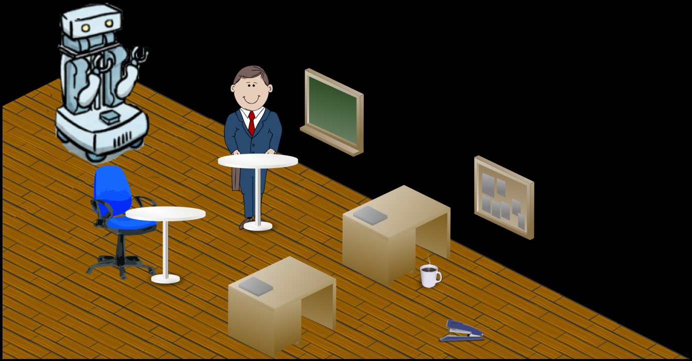


The PR2 Robot

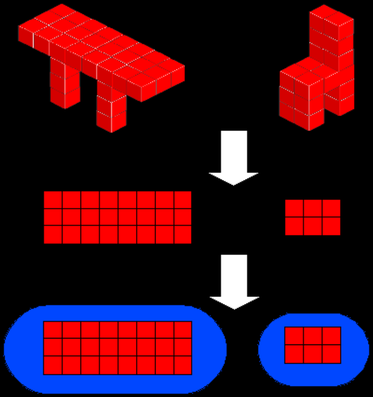
- Holonomic base
- Planar Hokuyo laser on base
- Actuated Hokuyo laser just below head - takes 2 seconds to produce a full 3D scan of the environment

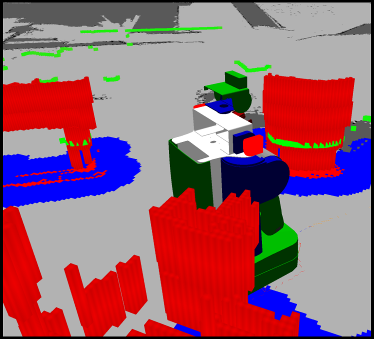
Obstacle Avoidance

- Use a 3D Voxel Grid to store information about known free, known occupied, and unknown space



Voxel Grid

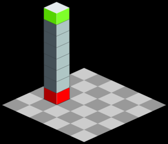




Voxel Grid Implementation

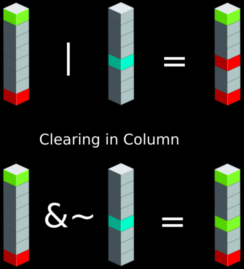
- 3D raytracing at 2D speed
- Allows for tracking of unknown space

2D Grid of
32-bit Integers



Every 2 bits of each
integer represents a cell
at a different height

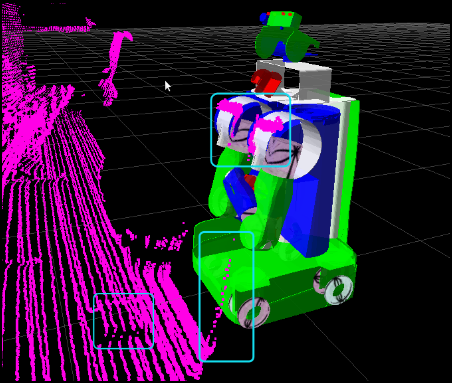
Merging in Column



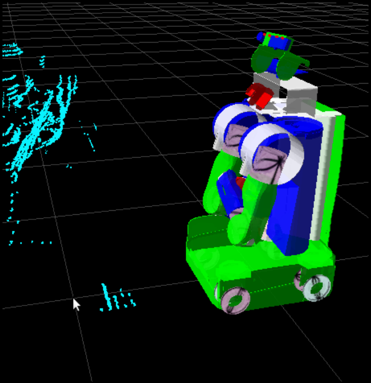
Clearing in Column

Sensor Processing Pipeline

Raw Sensor Data

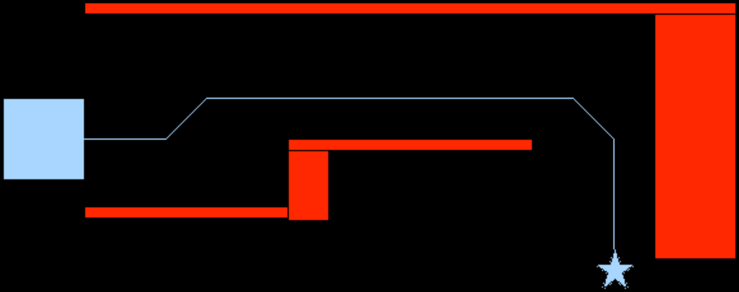


Processed Sensor Data



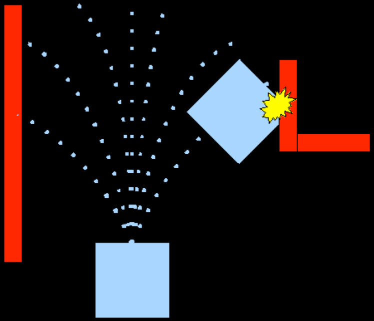
Global Planner

- Fast, grid based planner that uses an A* heuristic.
- Optimistic, uses the inscribed circle of the robot for planning.

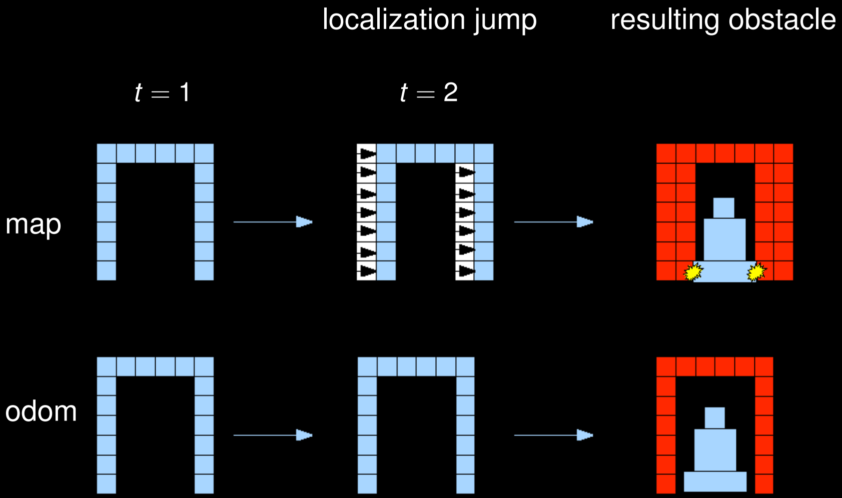


Local Planner

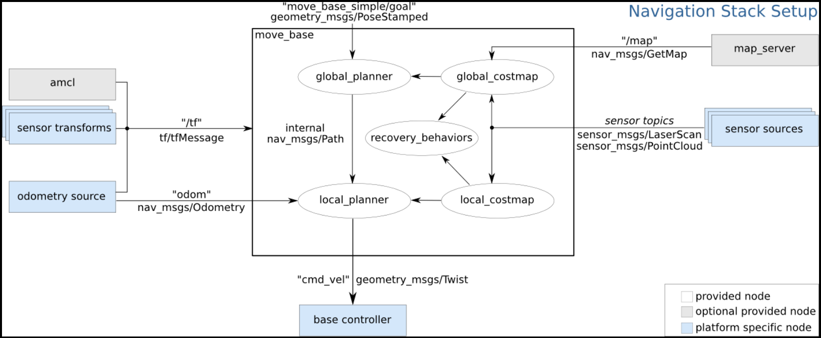
- Forward simulates a number of possible velocity commands using the Dynamic Window Approach.
- Checks for collisions using the footprint of the robot.



Frames Matter



The Navigation Stack



PR2 Navigation Flavors

- **pr2_2dnav_local:** Navigation in the odometric frame. Does not use any localization or a user-provided map.
- **pr2_2dnav_slam:** Navigation with SLAM, builds a map as you go.
- **pr2_2dnav:** Navigation with a user-provided map. Requires that the user initialize localization in that map using rviz.

Task 1: Make a Map

- Get things up and running on the robot:
  - `http://www.ros.org/wiki/pr2_2dnav_slam`
  - The joystick will be active, so you can drive the robot around
  - You can also send goals to the navigation stack using the "2D Nav Goal" button in rviz
- If you have extra time
  - Play around with rviz, give the robot a goal and jump in front of it, put an object in the robot's path and see if it can avoid it, etc.

Task 2: Goals With Code

- Complete the following tutorial
  - `http://www.ros.org/wiki/navigation/Tutorials/SimpleNavigationGoals`
- If you have extra time
  - Try to send a goal to the navigation stack in the "map" frame instead of the "base_link" frame
  - Try to send a goal to the navigation stack in python instead of C++ (`http://www.ros.org/wiki/actionlib`)

Task 3: Save and Use a Map

- Follow instructions on using the "map_saver" tool
  - `http://www.ros.org/wiki/map_server`
- Bring down pr2_2dnav_slam
- Follow instructions on using the "map_server" tool
  - `http://www.ros.org/wiki/map_server`
- Run pr2_2dnav
  - `http://www.ros.org/wiki/pr2_2dnav`
  - Send a goal using rviz or your code
- If you have extra time
  - Ask any questions you might have. Try to come up with something on your own. Take a break.

```
#wait for the action server to be available
move_base_client = actionlib.
   SimpleActionClient(’move_base_local’,
   MoveBaseAction)
move_base_client.wait_for_server()

#construct a simple goal in the base_link
   frame
goal = MoveBaseGoal()
goal.target_pose.header.frame_id = ’base_link’
goal.target_pose.pose.position.x = 1.0
goal.target_pose.pose.orientation.w = 1.0

#send the goal and wait for the base to get
   there
move_base_client.send_goal_and_wait(goal)
```

```
#Get the pose of the 3x4 checkerboard
get_checkerboard_pose = rospy.ServiceProxy('
   wide_get_checkerboard_pose',
   GetCheckerboardPose)
board_pose = get_checkerboard_pose(3, 4, .108,
    .108).board_pose
```

```
#given the pose of the checkerboard, get a
   good pose to approach it from
get_approach_pose = rospy.ServiceProxy(’
   get_approach_pose’, GetApproachPose)
nav_pose = get_approach_pose(board_pose).
   nav_pose
```

```
#OK... our nav_pose is now ready to be sent to
    the navigation stack as a
goal
move_base_client = actionlib.
   SimpleActionClient('move_base_local',
   MoveBaseAction)
move_base_client.wait_for_server()
goal = MoveBaseGoal()
goal.target_pose = nav_pose

#send the goal and wait for the base to get
   there
move_base_client.send_goal_and_wait(goal)
```