RoboND Where Am I Project

Kevin James
September 2018

Abstract

An important function of any robot operating in a structured environment is localizing itself within the environment. ROS provides packages for *Adaptive Monte Carlo Localization* with odometry, and navigation. These functions enable a robot equipped with lidar, and provided with a map, to determine probabilistically where it is on the map, to find a path to a new location, and to track its position as it moves. This project demonstrates the above functions on two robots, encompassing *global localization* and routing.

Introduction

This project evaluates ROS packages for localization and navigation in a mapped environment. Two robots are tested on the *jackal_race* map, which consists of a small course built from construction barriers. A robot starts in a corridor and must navigate out of the corridor to a goal on the other side of a barrier. Only one end of the corridor provides a route to the goal. To accomplish the navigation task, the robot must localize itself on the map. *Adaptive Monte Carlo Localization (AMCL)* uses a particle filter approach to probabilistically determining the robot's location over a sequence of {move, sense, resample} updates.

Background

The task of localization is a primary requirement for a robot operating in a structured environment. Unless a robot has a known starting point, even given a map the robot will be unable to navigate to a specific location until it has determined its own location. In the simulated world used for this project, the robot is required to move to a specified position and orientation. The robot starts on the map without its location and then uses laser range measurements and a particle filter to form a probabilistic estimate of its location. Each particle in the filter represents a hypothesis about the robot's pose. Particles are initialized randomly. The robot moves, and measures the locations of features in its environment. Robot motion causes each particle to be updated by the motion command. For example if the particle starts at position {x1, y1} and the command moves the robot by {2, 3} then the particle position becomes {x1 + 2 + noise, y1 + 3 + noise}. Measurements are then correlated against expectation for each particle. This process results in a product of Gaussian probabilities forming the weight for each particle. The *resampling* step creates a new set of particles distributed according to the normalized weights of

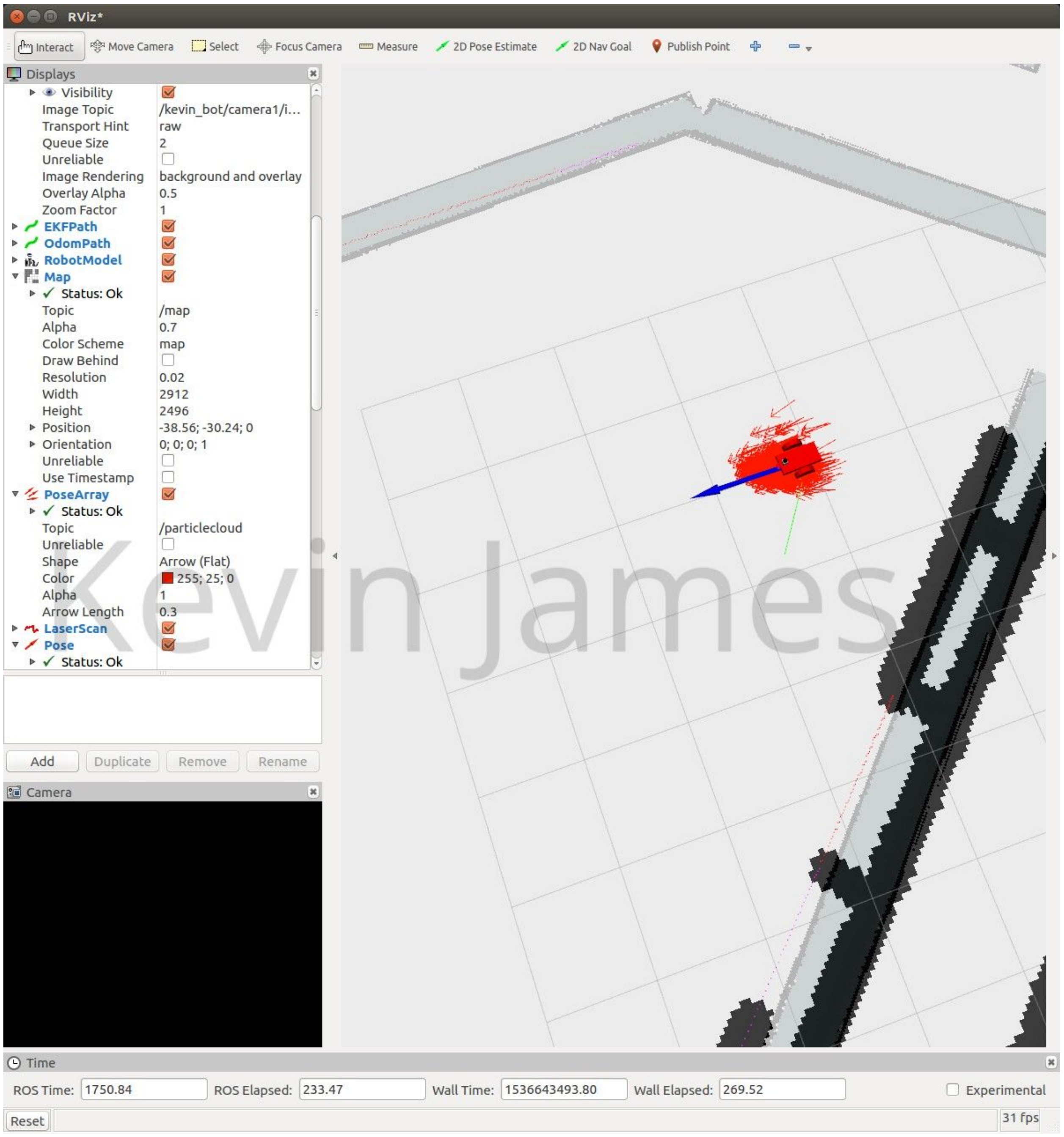
the current particles. Iterating this process focuses the particle distribution in the area of highest likelihood.
Unlike the unimodal Kalman family of filters, the particle filter is multimodal and maintains many hypotheses, enabling it to discover one or more plausible estimates of pose. For example, if the map contains four identical regions, and the robot is located in one such region, the particle filter should predict four likely poses until additional measurements enable the precise location to be established. The Extended Kalman Filter is good for determining the actual position of the robot as it moves locally using noisy actuators, but is not a good fit for the *global localization* problem that is the subject of this project.

Results

Initial localization results were unimpressive. As shown first for *udacitybot*, while the robot arrived at the goal, the particle clustering was not tight. The most effective adjustment was to the the *odom_alpha* parameters.

Udacitybot

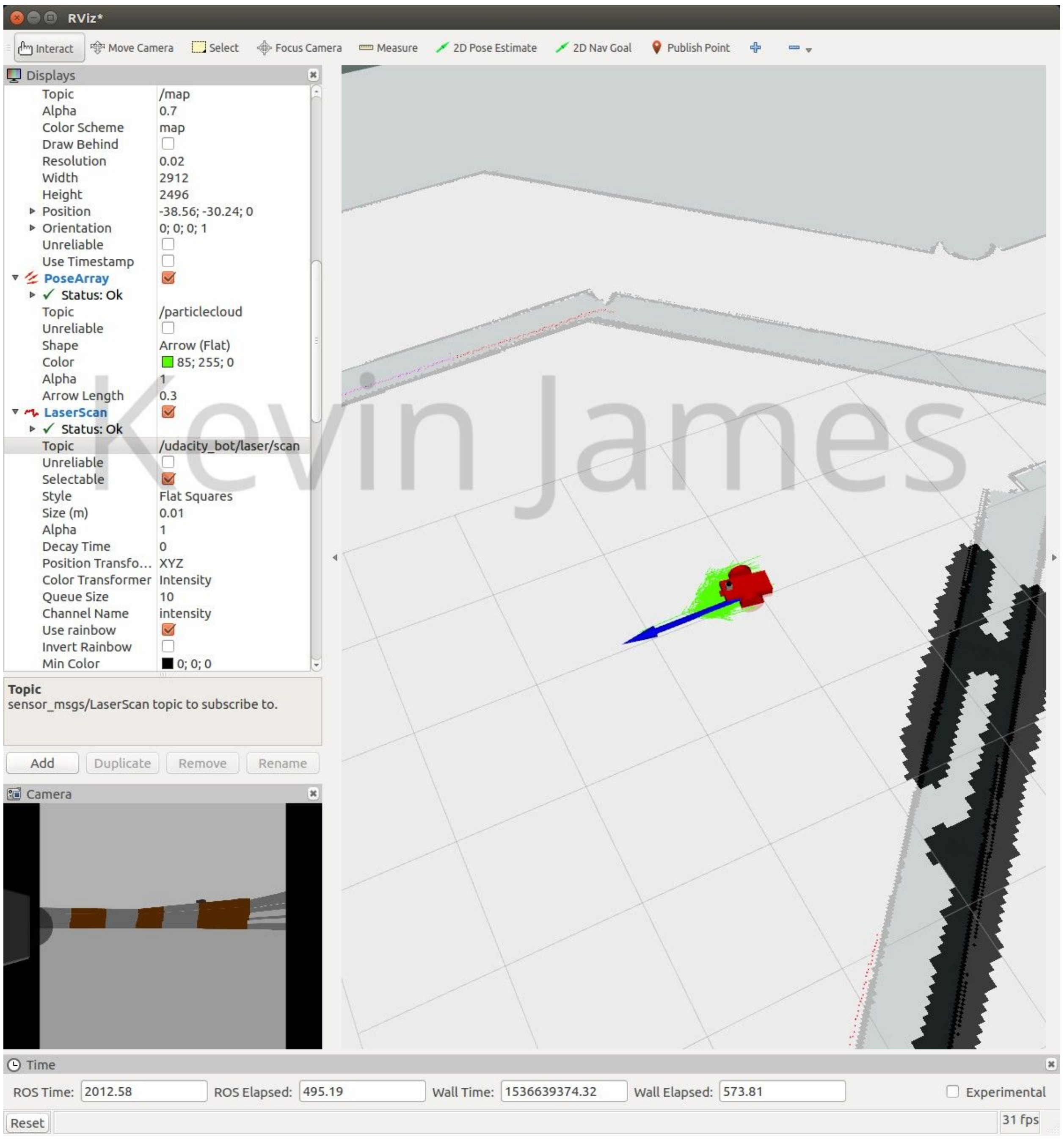
The following image shows the localization results with the listed parameters before changing *odom_alpha** from defaults.



Visibility	✓
Image Topic	/kevin_bot/camera1/i…
Transport Hint	raw
Queue Size	2
Unreliable	☐
Image Rendering	background and overlay
Overlay Alpha	0.5
Zoom Factor	1
EKFPath	✓
OdomPath	✓
RobotModel	✓
Map	✓
Status: Ok	
Topic	/map
Alpha	0.7
Color Scheme	map
Draw Behind	☐
Resolution	0.02
Width	2912
Height	2496
Position	-38.56; -30.24; 0
Orientation	0; 0; 0; 1
Unreliable	☐
Use Timestamp	☐
PoseArray	✓
Status: Ok	
Topic	/particlecloud
Unreliable	☐
Shape	Arrow (Flat)
Color	🟥 255; 25; 0
Alpha	1
Arrow Length	0.3
LaserScan	✓
Pose	✓
Status: Ok	

ROS Time: 1750.84 ROS Elapsed: 233.47 Wall Time: 1536643493.80 Wall Elapsed: 269.52 ☐ Experimental

The following image shows the robot at the goal with *odom_alpha** = 0.05.

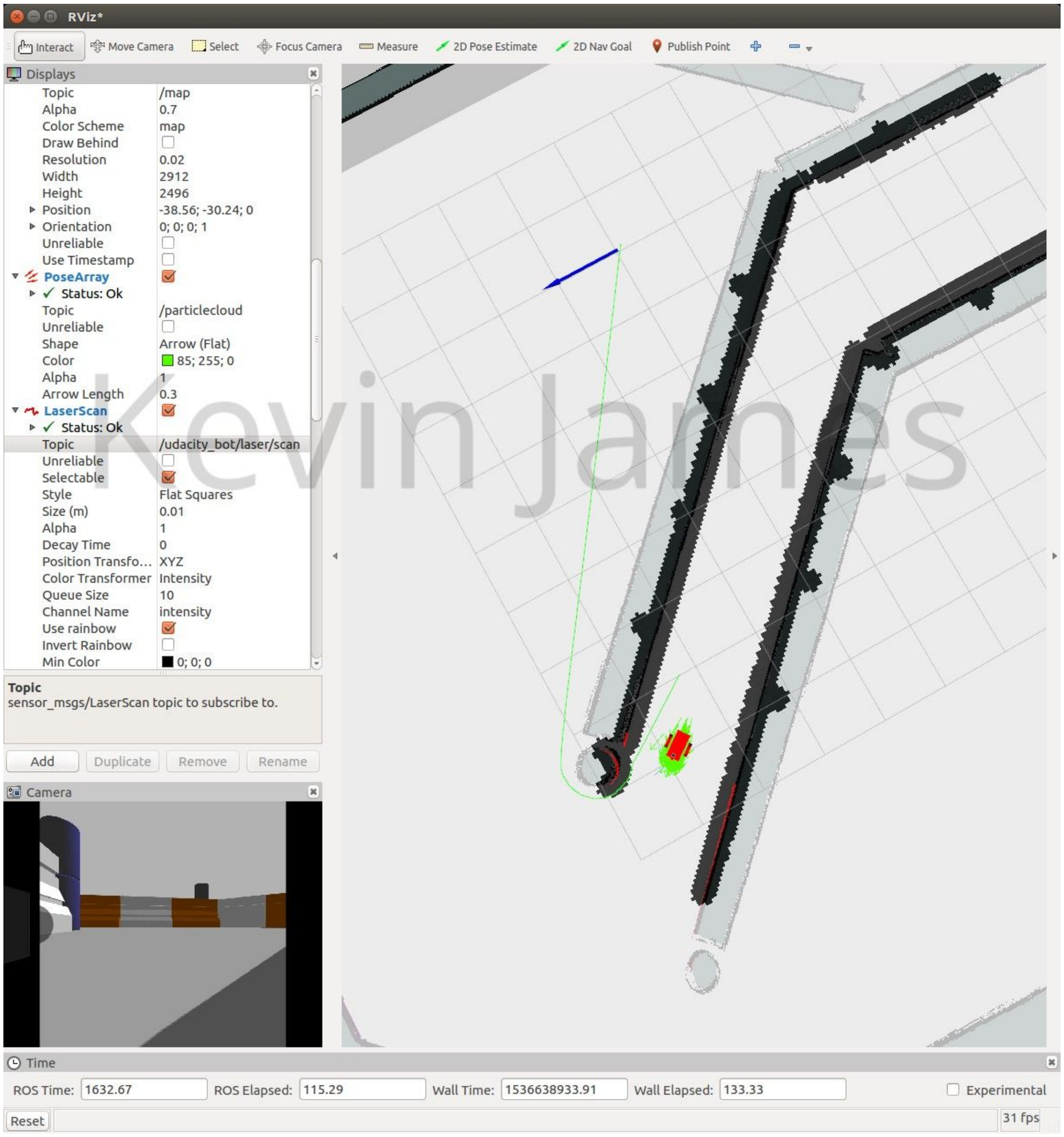


Topic	/map
Alpha	0.7
Color Scheme	map
Draw Behind	☐
Resolution	0.02
Width	2912
Height	2496
Position	-38.56; -30.24; 0
Orientation	0; 0; 0; 1
Unreliable	☐
Use Timestamp	☐
PoseArray	✓
Status: Ok	
Topic	/particlecloud
Unreliable	☐
Shape	Arrow (Flat)
Color	🟩 85; 255; 0
Alpha	1
Arrow Length	0.3
LaserScan	✓
Status: Ok	
Topic	/udacity_bot/laser/scan
Unreliable	☐
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transfo…	XYZ
Color Transformer	Intensity
Queue Size	10
Channel Name	intensity
Use rainbow	✓
Invert Rainbow	☐
Min Color	■ 0; 0; 0

Topic
sensor_msgs/LaserScan topic to subscribe to.

ROS Time: 2012.58 ROS Elapsed: 495.19 Wall Time: 1536639374.32 Wall Elapsed: 573.81 ☐ Experimental

The following image shows the robot midway to the goal, again *odom_alpha** = 0.05.



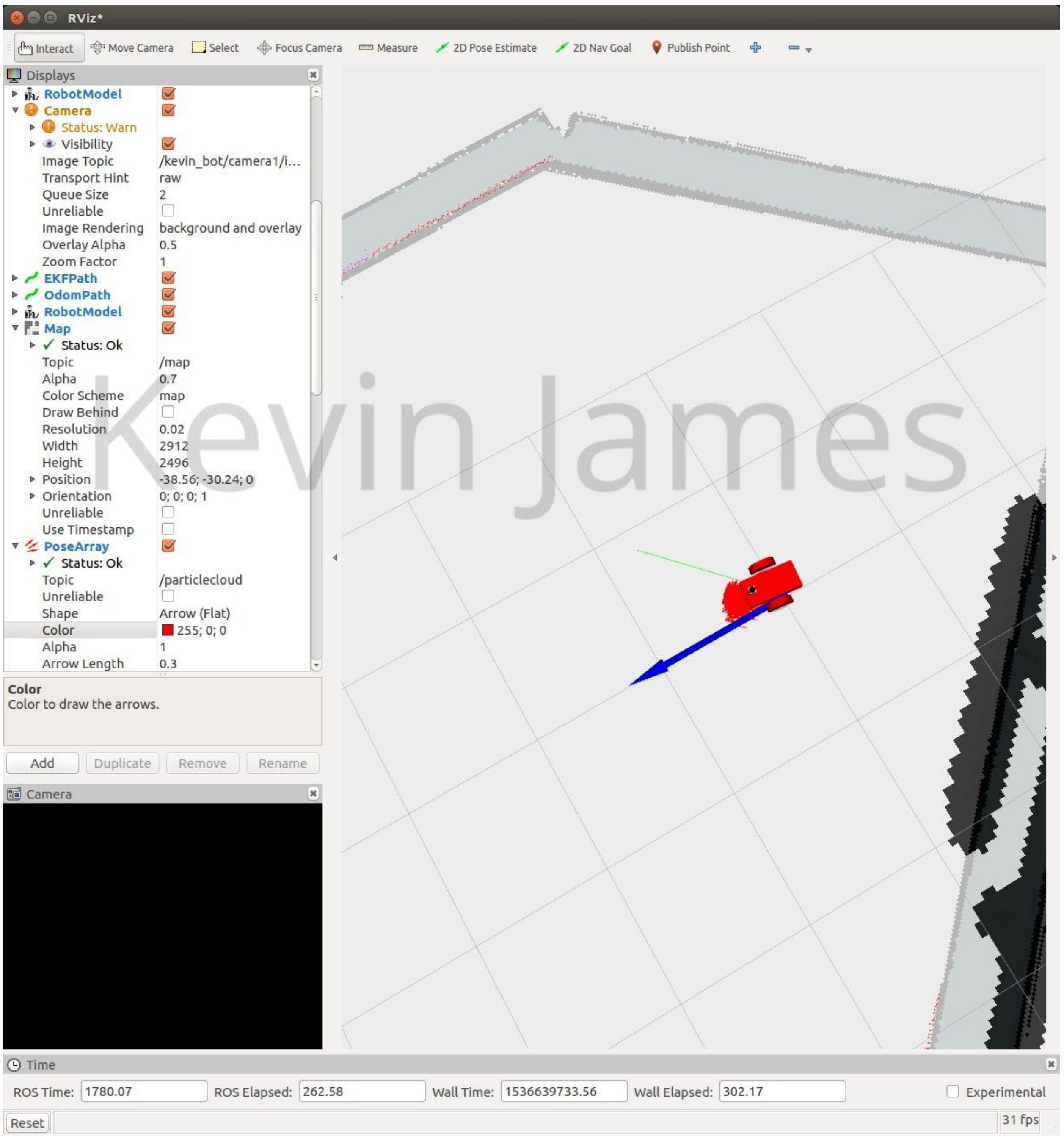
Topic	/map
Alpha	0.7
Color Scheme	map
Draw Behind	☐
Resolution	0.02
Width	2912
Height	2496
Position	-38.56; -30.24; 0
Orientation	0; 0; 0; 1
Unreliable	☐
Use Timestamp	☐
PoseArray	☑
✓ Status: Ok	
Topic	/particlecloud
Unreliable	☐
Shape	Arrow (Flat)
Color	🟩 85; 255; 0
Alpha	1
Arrow Length	0.3
LaserScan	☑
✓ Status: Ok	
Topic	/udacity_bot/laser/scan
Unreliable	☐
Selectable	☑
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transfo...	XYZ
Color Transformer	Intensity
Queue Size	10
Channel Name	intensity
Use rainbow	☑
Invert Rainbow	☐
Min Color	⬛ 0; 0; 0

Topic
sensor_msgs/LaserScan topic to subscribe to.

ROS Time: 1632.67 ROS Elapsed: 115.29 Wall Time: 1536638933.91 Wall Elapsed: 133.33 ☐ Experimental

Reset 31 fps

The following image shows the robot at the goal, this time with *odom_alpha** = 0.01. The localization in this case is better than with higher *alpha* values. Localization is good, with particles clustered under the robot's footprint.

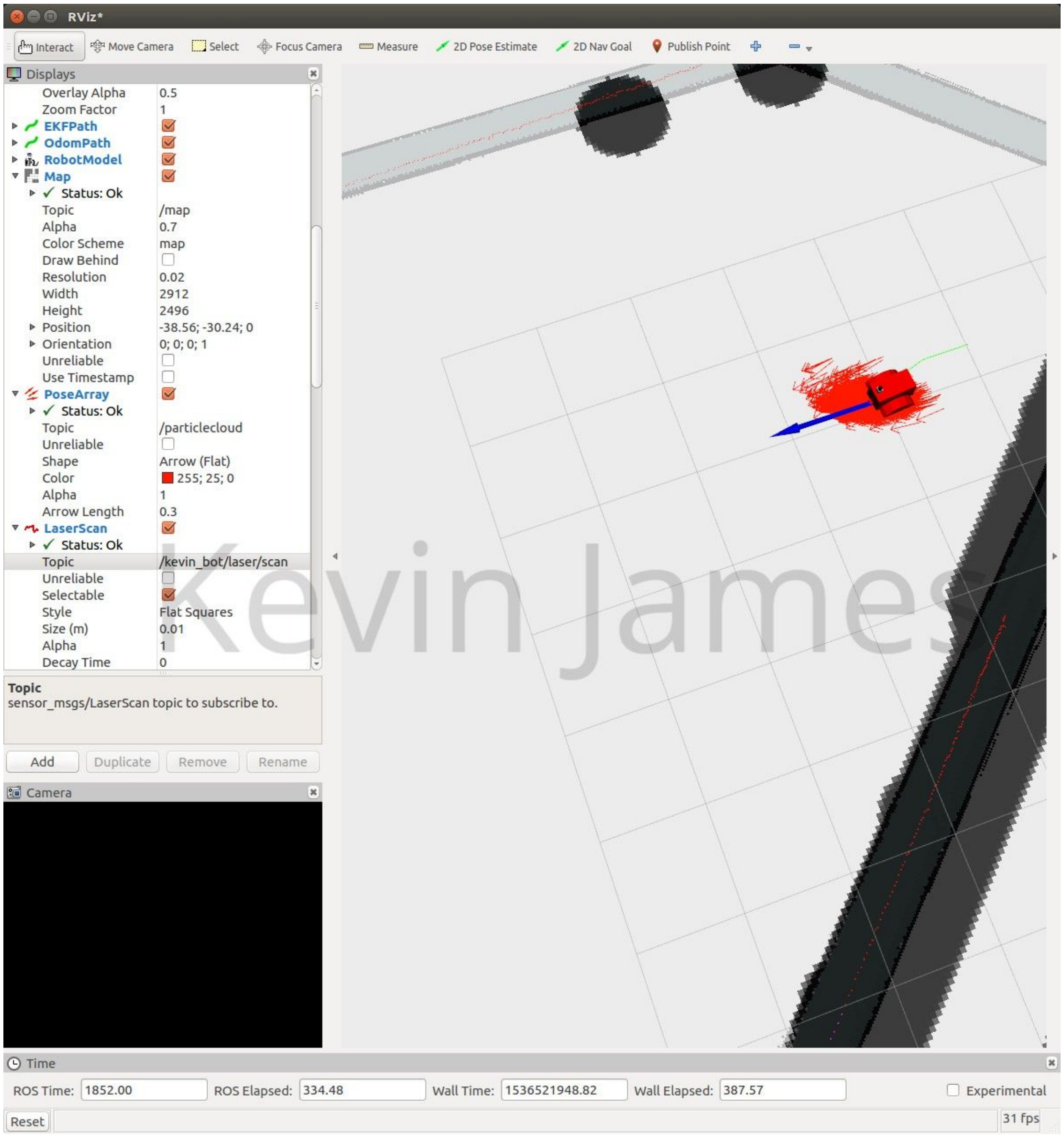


```
RobotModel        ✓
Camera            ✓
  Status: Warn
  Visibility      ✓
  Image Topic     /kevin_bot/camera1/i...
  Transport Hint  raw
  Queue Size      2
  Unreliable      ☐
  Image Rendering background and overlay
  Overlay Alpha   0.5
  Zoom Factor     1
EKFPath           ✓
OdomPath          ✓
RobotModel        ✓
Map               ✓
  Status: Ok
  Topic           /map
  Alpha           0.7
  Color Scheme    map
  Draw Behind     ☐
  Resolution      0.02
  Width           2912
  Height          2496
  Position        -38.56; -30.24; 0
  Orientation     0; 0; 0; 1
  Unreliable      ☐
  Use Timestamp   ☐
PoseArray         ✓
  Status: Ok
  Topic           /particlecloud
  Unreliable      ☐
  Shape           Arrow (Flat)
  Color           🟥 255; 0; 0
  Alpha           1
  Arrow Length    0.3

Color
Color to draw the arrows.

ROS Time: 1780.07   ROS Elapsed: 262.58   Wall Time: 1536639733.56   Wall Elapsed: 302.17
```

Modified Robot

The modified robot localization results are shown in this section. Additional parameter settings were tested with this robot and some examples are presented. As in the previous section, *odom_alpha** parameters had by far the greatest effect on improving localization.

The following image shows the robot at the goal with *laser_max_beam* = 128 and *max_particles* = 10k. While the location is generally correct, the distribution of particles is not tight. Further, additional beams did not improve localization.

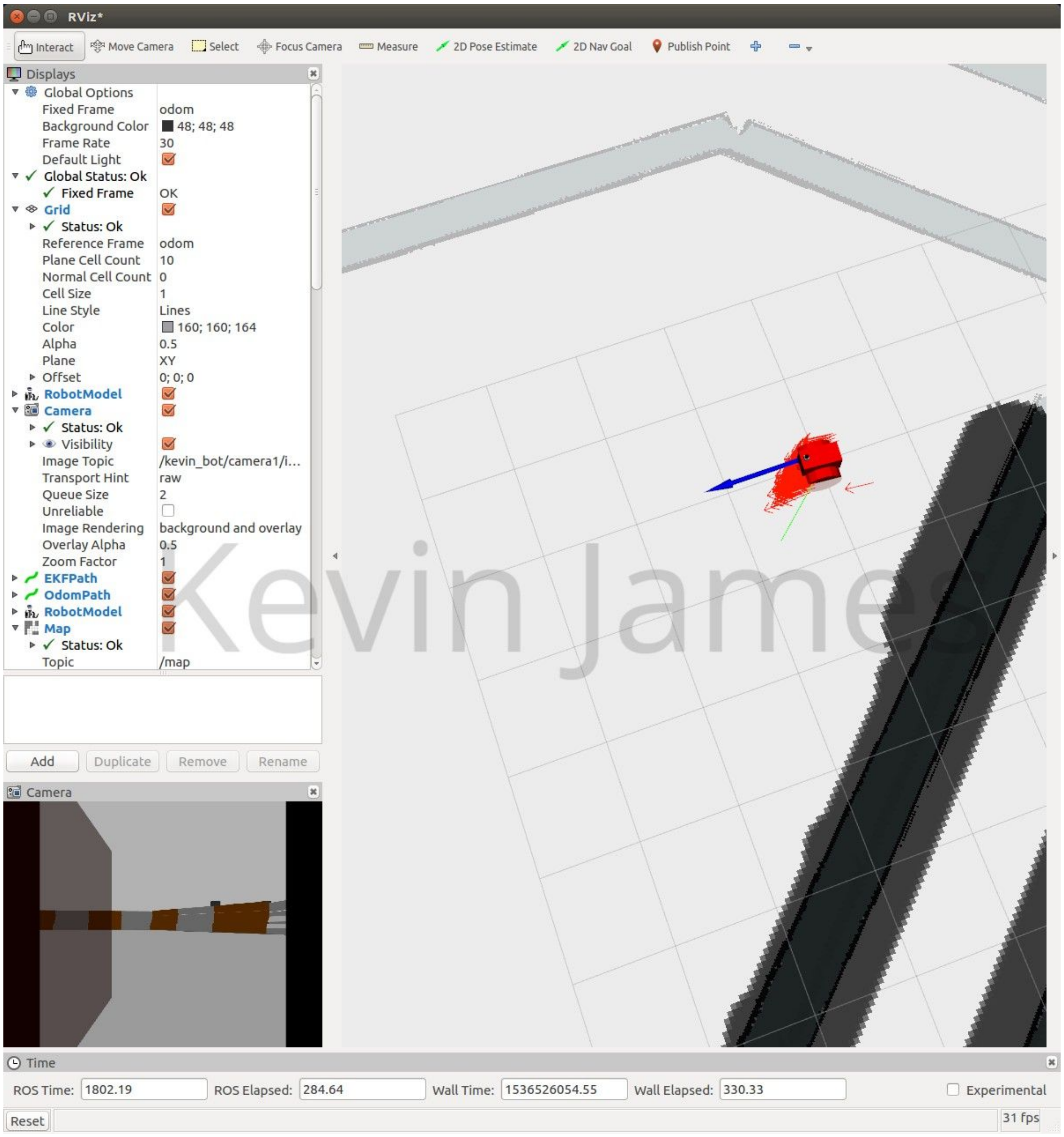


Overlay Alpha	0.5
Zoom Factor	1
EKFPath	✓
OdomPath	✓
RobotModel	✓
Map	✓
Status: Ok	
Topic	/map
Alpha	0.7
Color Scheme	map
Draw Behind	☐
Resolution	0.02
Width	2912
Height	2496
Position	-38.56; -30.24; 0
Orientation	0; 0; 0; 1
Unreliable	☐
Use Timestamp	☐
PoseArray	✓
Status: Ok	
Topic	/particlecloud
Unreliable	☐
Shape	Arrow (Flat)
Color	🟥 255; 25; 0
Alpha	1
Arrow Length	0.3
LaserScan	✓
Status: Ok	
Topic	/kevin_bot/laser/scan
Unreliable	☐
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0

Topic
sensor_msgs/LaserScan topic to subscribe to.

ROS Time: 1852.00 ROS Elapsed: 334.48 Wall Time: 1536521948.82 Wall Elapsed: 387.57 Experimental ☐

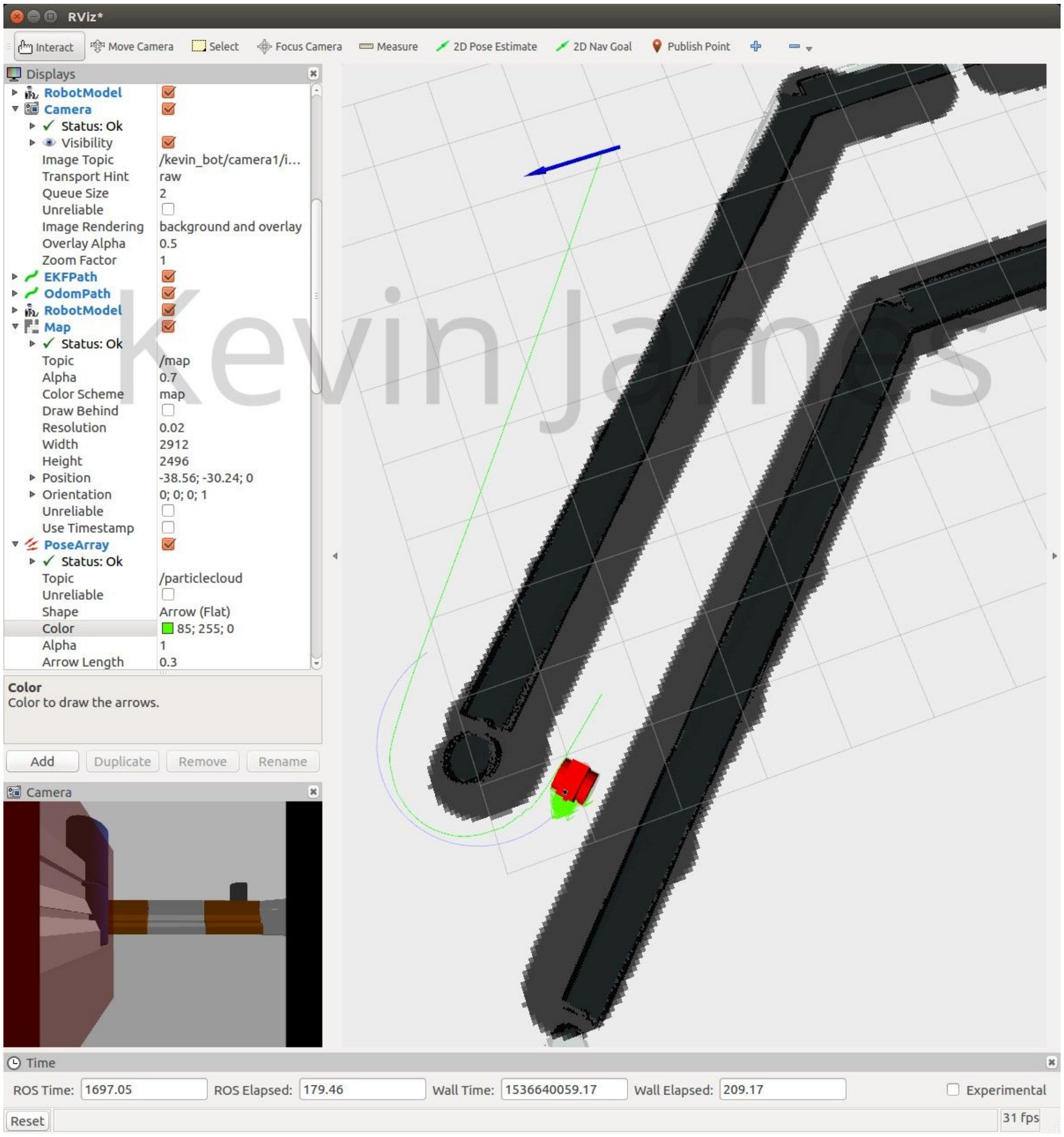
In contrast, the following image shows the same 128 beam configuration with 1k particles and *odom_alpha** set to 0.05. The particle distribution is finally tightening.



Global Options	
Fixed Frame	odom
Background Color	48; 48; 48
Frame Rate	30
Default Light	✓
Global Status: Ok	
Fixed Frame	OK
Grid	✓
Status: Ok	
Reference Frame	odom
Plane Cell Count	10
Normal Cell Count	0
Cell Size	1
Line Style	Lines
Color	160; 160; 164
Alpha	0.5
Plane	XY
Offset	0; 0; 0
RobotModel	✓
Camera	✓
Status: Ok	
Visibility	✓
Image Topic	/kevin_bot/camera1/i...
Transport Hint	raw
Queue Size	2
Unreliable	
Image Rendering	background and overlay
Overlay Alpha	0.5
Zoom Factor	1
EKFPath	✓
OdomPath	✓
RobotModel	✓
Map	✓
Status: Ok	
Topic	/map

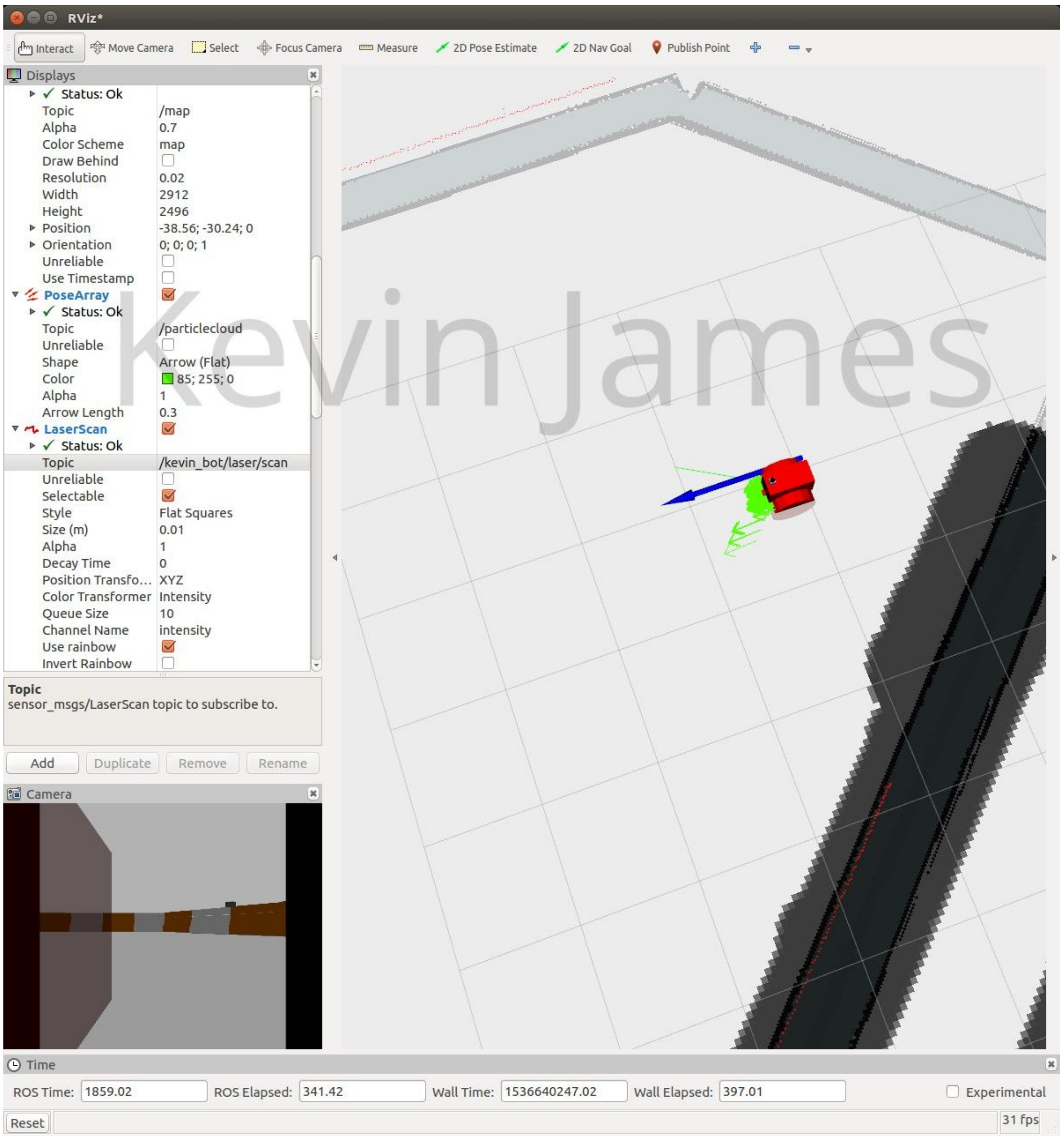
ROS Time: 1802.19 ROS Elapsed: 284.64 Wall Time: 1536526054.55 Wall Elapsed: 330.33 Experimental

Continuing down this path, the following image shows the robot midway to the goal with *odom_alpha** = 0.01.



```
RobotModel          ✓
Camera              ✓
  ✓ Status: Ok
  Visibility        ✓
  Image Topic       /kevin_bot/camera1/i...
  Transport Hint    raw
  Queue Size        2
  Unreliable        ☐
  Image Rendering   background and overlay
  Overlay Alpha     0.5
  Zoom Factor       1
EKFPath             ✓
OdomPath            ✓
RobotModel          ✓
Map                 ✓
  ✓ Status: Ok
  Topic             /map
  Alpha             0.7
  Color Scheme      map
  Draw Behind       ☐
  Resolution        0.02
  Width             2912
  Height            2496
  Position          -38.56; -30.24; 0
  Orientation       0; 0; 0; 1
  Unreliable        ☐
  Use Timestamp     ☐
PoseArray           ✓
  ✓ Status: Ok
  Topic             /particlecloud
  Unreliable        ☐
  Shape             Arrow (Flat)
  Color             🟩 85; 255; 0
  Alpha             1
  Arrow Length      0.3

Color
Color to draw the arrows.

ROS Time: 1697.05   ROS Elapsed: 179.46   Wall Time: 1536640059.17   Wall Elapsed: 209.17   ☐ Experimental
31 fps
```

Finally, the robot is shown at the goal (same parameters as above). The robot is well localized here, with almost all particles under its footprint (the green line above the goal arrow is the global path).



Status: Ok	
Topic	/map
Alpha	0.7
Color Scheme	map
Draw Behind	☐
Resolution	0.02
Width	2912
Height	2496
Position	-38.56; -30.24; 0
Orientation	0; 0; 0; 1
Unreliable	☐
Use Timestamp	☐
PoseArray	✓
Status: Ok	
Topic	/particlecloud
Unreliable	☐
Shape	Arrow (Flat)
Color	🟩 85; 255; 0
Alpha	1
Arrow Length	0.3
LaserScan	✓
Status: Ok	
Topic	/kevin_bot/laser/scan
Unreliable	☐
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transfo...	XYZ
Color Transformer	Intensity
Queue Size	10
Channel Name	intensity
Use rainbow	✓
Invert Rainbow	☐

Topic
sensor_msgs/LaserScan topic to subscribe to.

ROS Time: 1859.02 ROS Elapsed: 341.42 Wall Time: 1536640247.02 Wall Elapsed: 397.01 ☐ Experimental

31 fps

Model Configuration

The table below lists the parameters used unless otherwise stated. In general, the *laser* settings improved range and accuracy of the laser returns but had little effect on particle distribution. *update_min_** helped keep the robot moving early in the process.

Parameter	Value
laser_min_range	0.25
laser_max_range	25.0
laser_z_hit	0.90
laser_z_rand	0.10
laser_likelihood_max_dist	1.0
min_particles	50
max_particles	1000
recovery_alpha_slow	0.001
update_min_d	0.1
update_min_a	0.1

The modified robot (*kevin_bot.xacro*) is a taller version of *udacitybot* with larger, heavier wheels. Because of its added height, the laser sensor is mounted higher. The platform has greater wheel inertia (https://en.wikipedia.org/wiki/List_of_moments_of_inertia) and higher torque specified in the differential drive controller (*kevin_bot.gazebo*). Prior to adjusting the inertia values, the large-wheeled robot had a pronounced dive periodically as the robot (presumably) reduced its velocity. Some hint remains but is much less apparent. While not tested in this project, the configuration should improve the robot's ability to roll over uneven surfaces or small obstacles. The higher sensor mount location will improve visibility over objects. In this project, the difference in sensor height causes an offset due to the profile of the barrier. For this reason, *inflation_radius* is increased in the costmap configuration.

Number of Particles

Increasing the number of particles was not effective at improving localization. Too few particles could fail to localize the robot, but in this environment, even 10 particles may suffice. Such a small number would not be very robust and would not handle more complex maps. While the distribution of a small number of particles appears tighter, *enough* particles should be used, and with that level set at 1000, tightening the distribution required reducing the *odom_alpha** parameter values.

Discussion

odom-alpha

This project was completed on Ubuntu with ROS Kinetic; behavior on the Udacity Workspace may vary.

Using the diff_corrected odometry model, the parameters *odom_alpha1 - odom_alpha5* must be updated. The fact that the default values correspond to the uncorrected algorithm is mentioned in the ROS documentation (http://wiki.ros.org/amcl, https://answers.ros.org/question/227811/tuning-amcls-diff-corrected-and-omni-corrected-odom-models/).

The values used are only lightly tuned but show a clear improvement in localization performance. In retrospect, and after reviewing *Probabilistic Robotics*, it is clear that the particle filter performance must depend on the platform odometry estimation since the measurement probabilities are computed using the pose of the robot. Earlier in the tuning process, the *alpha* values were considered more important for routing, which was performing very poorly, but was not the focus of the project. An unfortunately oversight.

Kidnapped Robot

The adaptive nature of AMCL would enable the algorithm to function for the kidnapped robot problem. This problem requires not only that the robot localize itself from an unknown initial position, but also that the robot be able to adapt to unexpected relocation during its operation. If the probability distribution were determined only at the start of operation, teleporting the robot would leave it in a bad state. However, AMCL will adjust the number of particles in response to uncertainty, such that poor correlation with measurements causes new particles to be created with random pose. These new particles represent new hypotheses that can then guide the filter's distribution toward the new location. Note: due to limited documentation of internal details for ROS AMCL, the preceding two sentences are speculative but the concept is implied, for example in http://papers.nips.cc/paper/1998-kld-sampling-adaptive-particle-filters.pdf.

Industry Application

The robots tested in this project could be deployed in a mapped environment, such as a warehouse. A robot could be powered on in an arbitrary location, localize itself, and proceed to specified locations to retrieve products for shipping. To achieve a good result in this application, the warehouse should have distinguishing features that enable the robot to differentiate regions of the warehouse.

Future Work

The next robot should be be sized to contain a Jetson TX2 board and equipped with 4 omnidirectional wheels. Single-beam non-rotating lidar is available for hobbyist use. A servo-driven platform and measurement accumulation stage will be needed to compose a set of inputs for the particle filter. The Jetson's embedded processor is expected to have sufficient computational resources for the range of particle counts used in this project. For indoor use, the chassis should sized to accommodate the board and sensors but not larger.

While not the intended focus of the project, the routing component, specifically the local path determination and controls for following the local path, are the main area requiring attention. The robot frequently moved in an entirely incorrect direction for no apparent reason, and often became stuck for a frustratingly long time moving back and forth in the corridor. Part of the cause is likely to be discrepancies in the physical response of the robot to commands and consequent excessive error in motion. To take advantage of accurate localization, the robot needs to be able to move reliably.