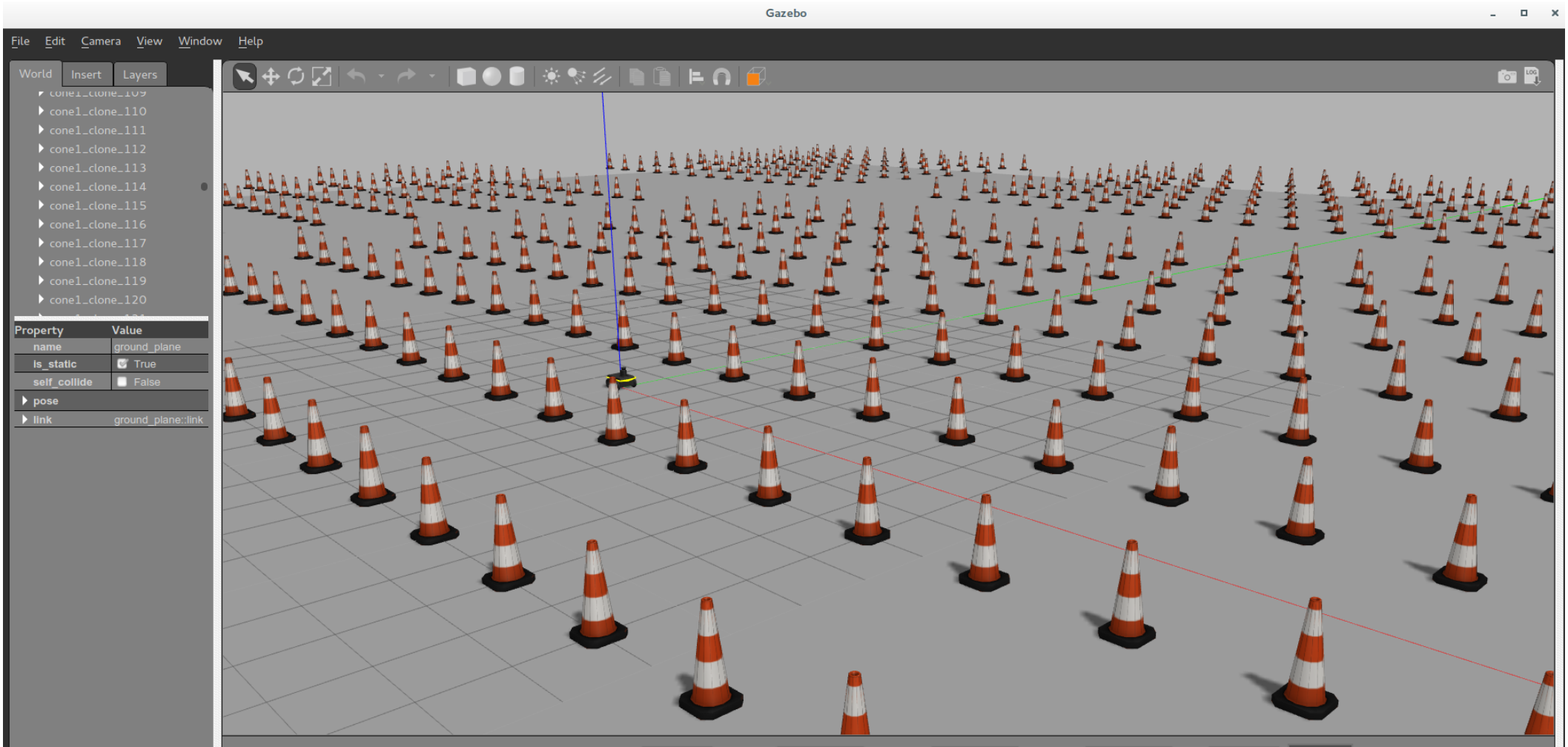
The effect of orchard heterogeneity on navigating an autonomous vehicle for agricultural tasks

Submits: Liron Shenkar

Supervisors Prof. Amir Degani , Omer Shalev



Project Highlights

Project Goal: To examine the performance of AMCL <u>global</u> localization algorithm under various heterogeneities of orchards of cones & cylinders.

- The AMCL (Adaptive Monte Carlo Localization) algorithm is a version of a particle filter.
- Like any localization algorithm, It gets a map of the area of interest (called “ground truth”) and some measurements (laser scan).
- Having both, the algorithm shall try to find the robot’s location on the map ,which is called “hypothesis”. The robot is CLEARPATH® jackal.
- Like humans, localization algorithms seek for some distinguishable landmarks in the area (“anchors”) in order to locate themselves on the given map.

- ***The higher the homogeneity of the area-the harder it is to deliver good localization results.***

- ***We wish to know how exactly homogeneities influences AMCL performance.***

Project Highlights

- A detailed info about AMCL implementation in ROS can be found here http://wiki.ros.org/amcl

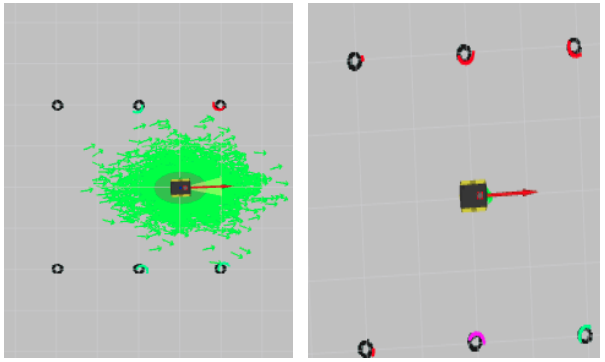
- The AMCL implementation has two modes of operation:

  Local localization

  The default mode of the AMCL algorithm whenever it is being invoked. Besides the ***map*** and the ***laser scan***, the algorithm's optional inputs are:
  - Initial robot's pose in meters [x,y,z].
  - Initial 6X6 Covariance matrix (x, y, z, rotation about X axis, rotation about Y axis, rotation about Z axis). **This matrix states the amount of uncertainty of the initial position and rotation of the robot.**

    The covariance can be visualized in 2D in Rviz as an elliptical particle could. Seen below is a particle cloud of [0.5,0.5] covariance matrix (other matrix values were zeroed) Vs a very small particle cloud concentrated under the robot:

    The particle cloud corresponds to the area in which the robot might be with high probability



Project Highlights

Local localization (Cont.)

- The output of the algorithm is the robot's location on the map (hypothesis) . It is seen in Rviz simply as the jackal (see previous slide).

- **Just as the input, the output (hypothesis) of the algorithm is always attached to a covariance matrix**, **which states (again) the amount of its uncertainty** .

- Local localization is not really practical. The main reasons are:
  i. The search is local-only around the area of the relatively small covariance matrix.
  ii. It tends to converge quickly and stick to the given initial pose, whether it is true or not. It does so even when the map and the measurements disagree with the hypothesis.

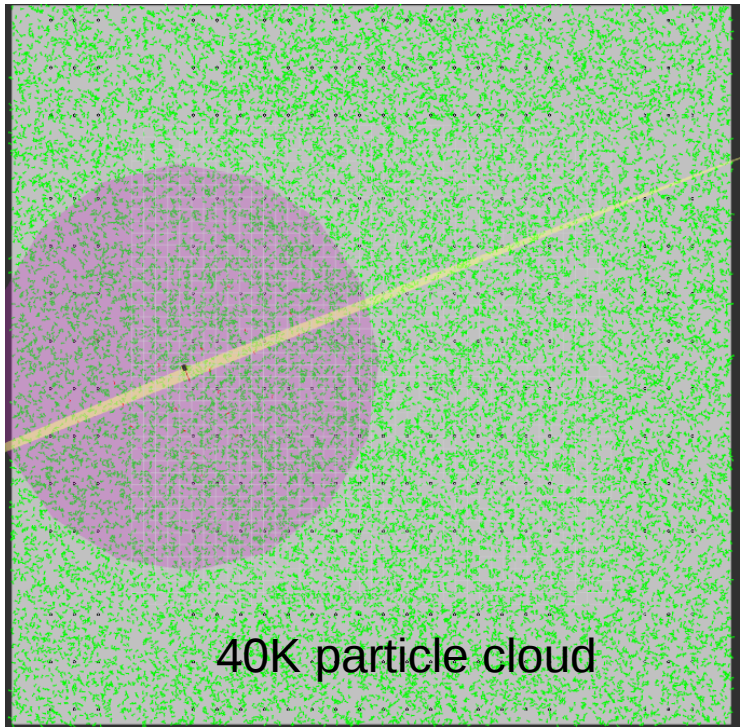
Global localization

- Invoked by the user by calling a service : *rosservice call /global_localization*. It runs "on top" of the local search and does not get pose/covariance of its own. **All the algorithm parameters are defined in the *amcl.launch* file, inside the navigation pack.**
- The most important parameter is the **"max_particles"** . It determines the maximum number of particles used by the algorithm. **It is the only parameter I changed**.

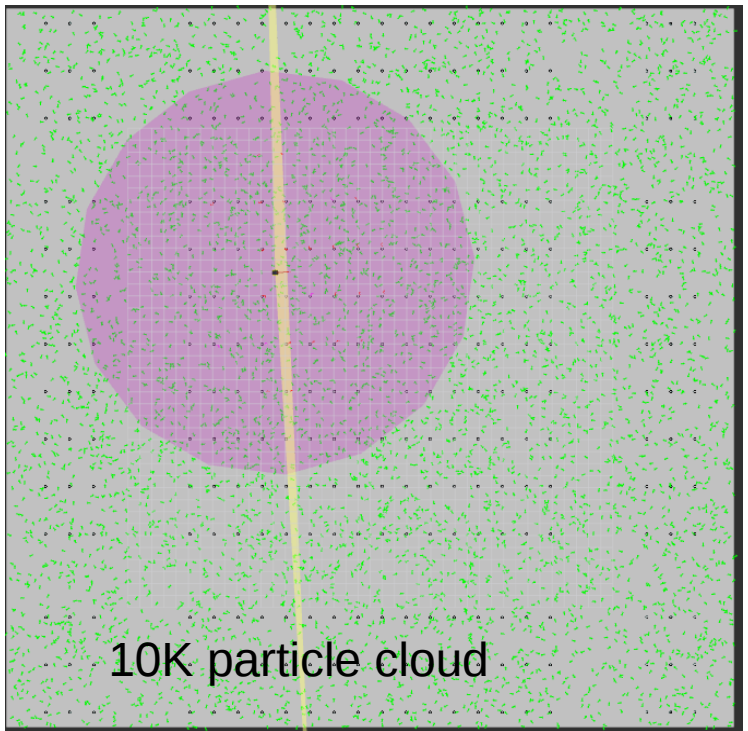
Project Highlights

Global localization (Cont.)

- When the *global_localization* service is being called, A particle cloud of **max_particles** size covers uniformly the *entire* map.
- **Initial or current estimated pose & covariance matrix are ignored. Global search is a “new game”- Only the map and the laser scan are taken into account, past knowledge Is overridden.**
- During search, the uniformly-distributed particle cloud changes its shape until convergence, when only a very small cloud, just under the robot’s estimated location, remains.
- The images below shows clouds of 40K & 10K particles, uniformly distributed.



40K particle cloud



10K particle cloud

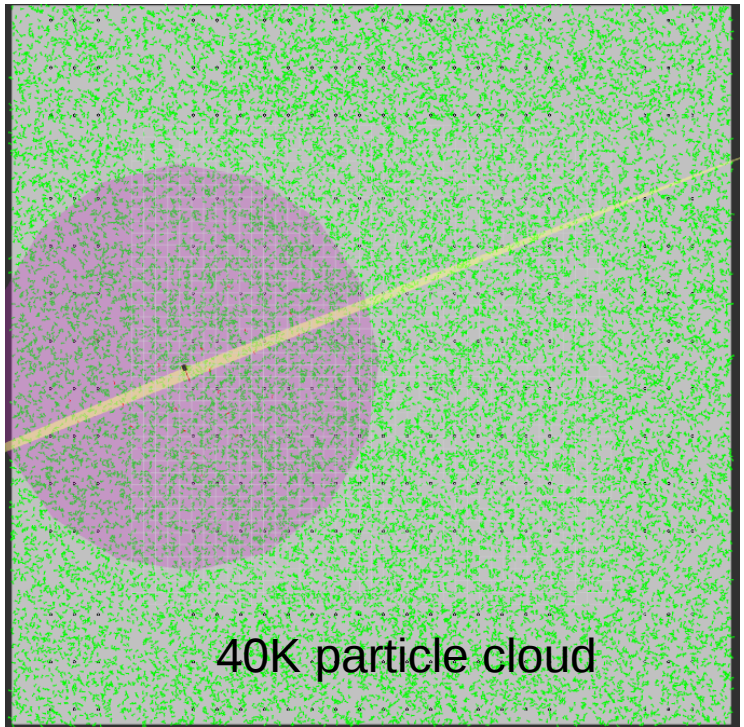
Project Highlights

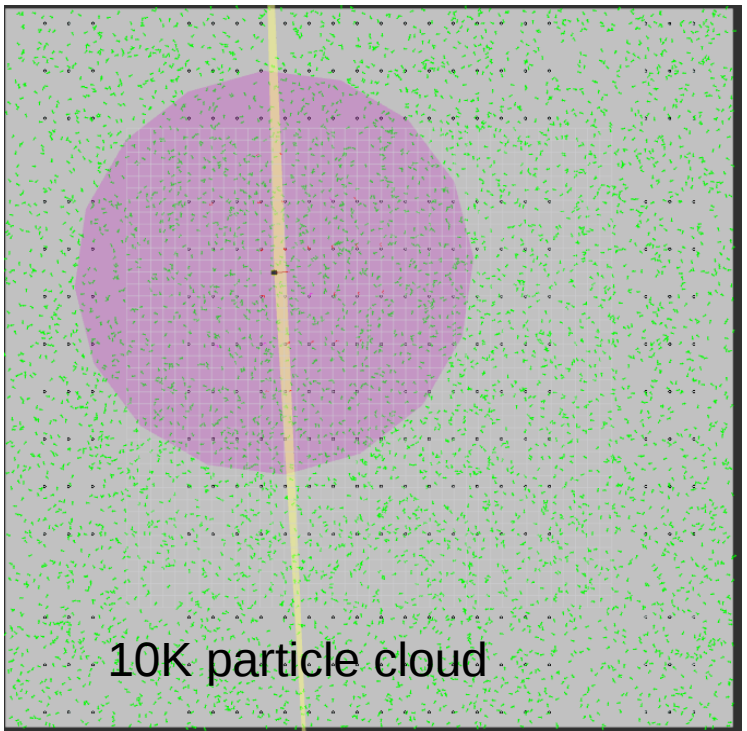
Global localization (Cont.)

- The pink area and the yellow triangle seen below (in Rviz) are the current best estimated position and orientation ,respectively.

- Local search is not the scope of this work. I only focused on the global search algorithm performance under various heterogeneity conditions. The only parameter that was changed was the maximum number of particles.

- In order to be consistent the global search was initiated right at the beginning of the simulations (there here are more reasons which are detailed later).





Project Highlights

Experiments methodology

- Each experiment is a ***series*** of simulations over a Gazebo world.

- **Definitions of success/failure in a simulation**
  - Success- Algorithm finds true pose(X **AND** Y coordinates) & true orientation.
  - Partial success- Algorithm finds true X **OR** Y coordinates & true orientation.
  - Failure- Algorithm finds **Neither** true X **Nor** Y coordinates.

- Each simulation was repeated several times, usually between 5-10. ***Why? -->***

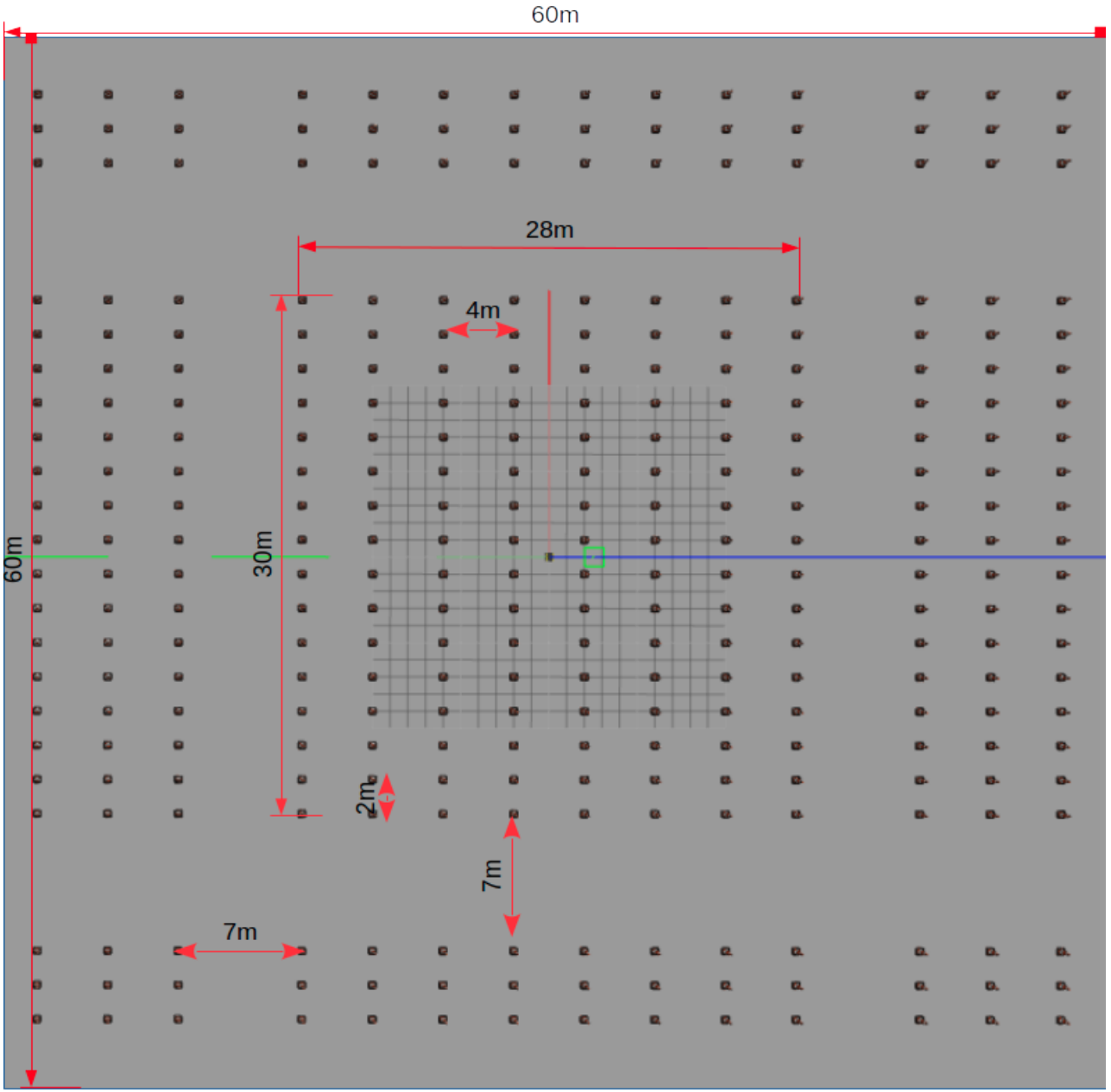
- **Successive simulations over the same world do NOT necessarily yield the same results**.

- Some of the reasons are:

  - Odometry noise and errors are not identical in all simulations.
  - Measurement noise is also not identical.
  - The AMCL algorithm is statistical in nature. The re-sampling process and the distribution of the particle cloud in each step aren't deterministic.

- In the end, the results are summed , and the success rate of each experiment is calculated.

The world



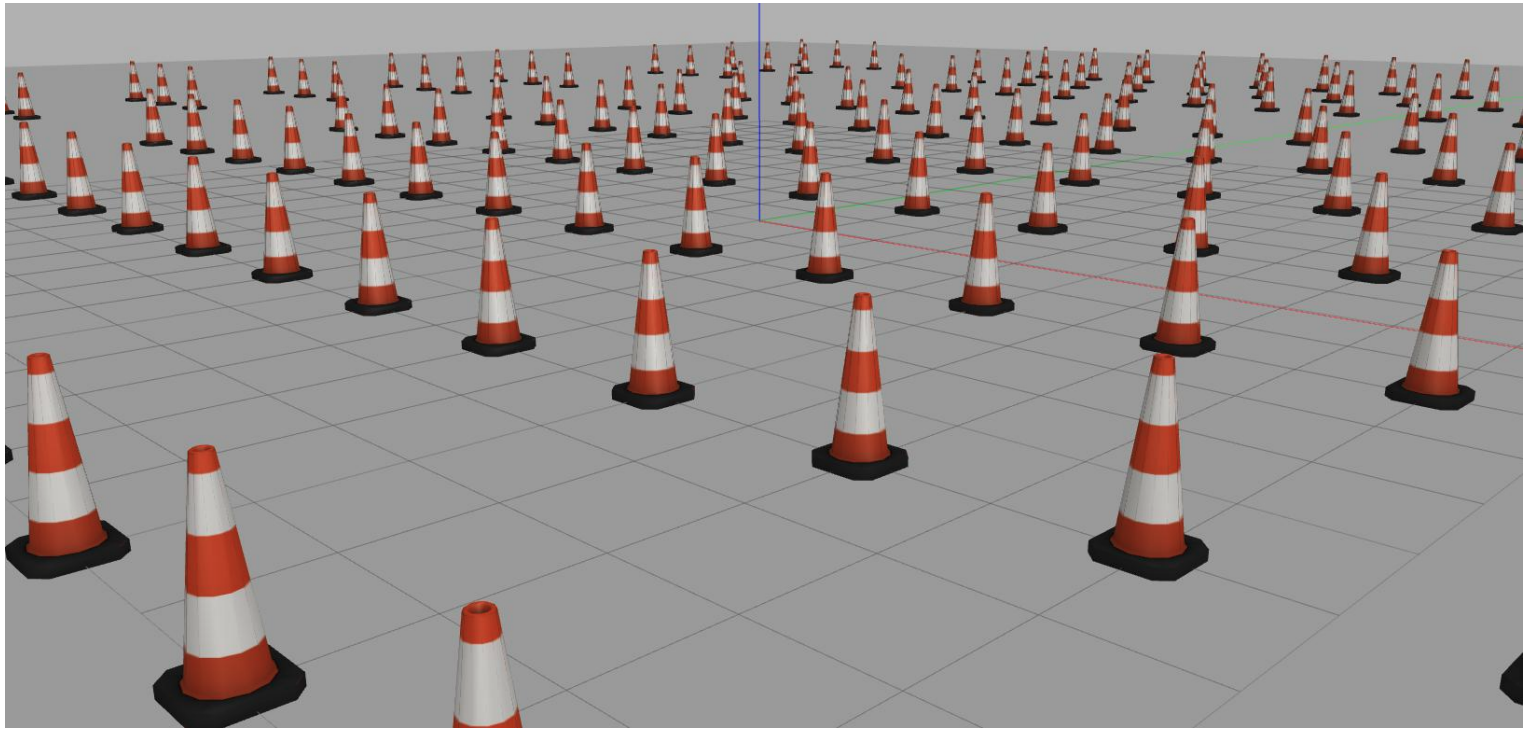
Project Highlights

- The basic (homogeneous) world was created in **Gazebo 7.2** , exploiting the "***Population of models***" feature, which was not available prior to gazebo 7.0.

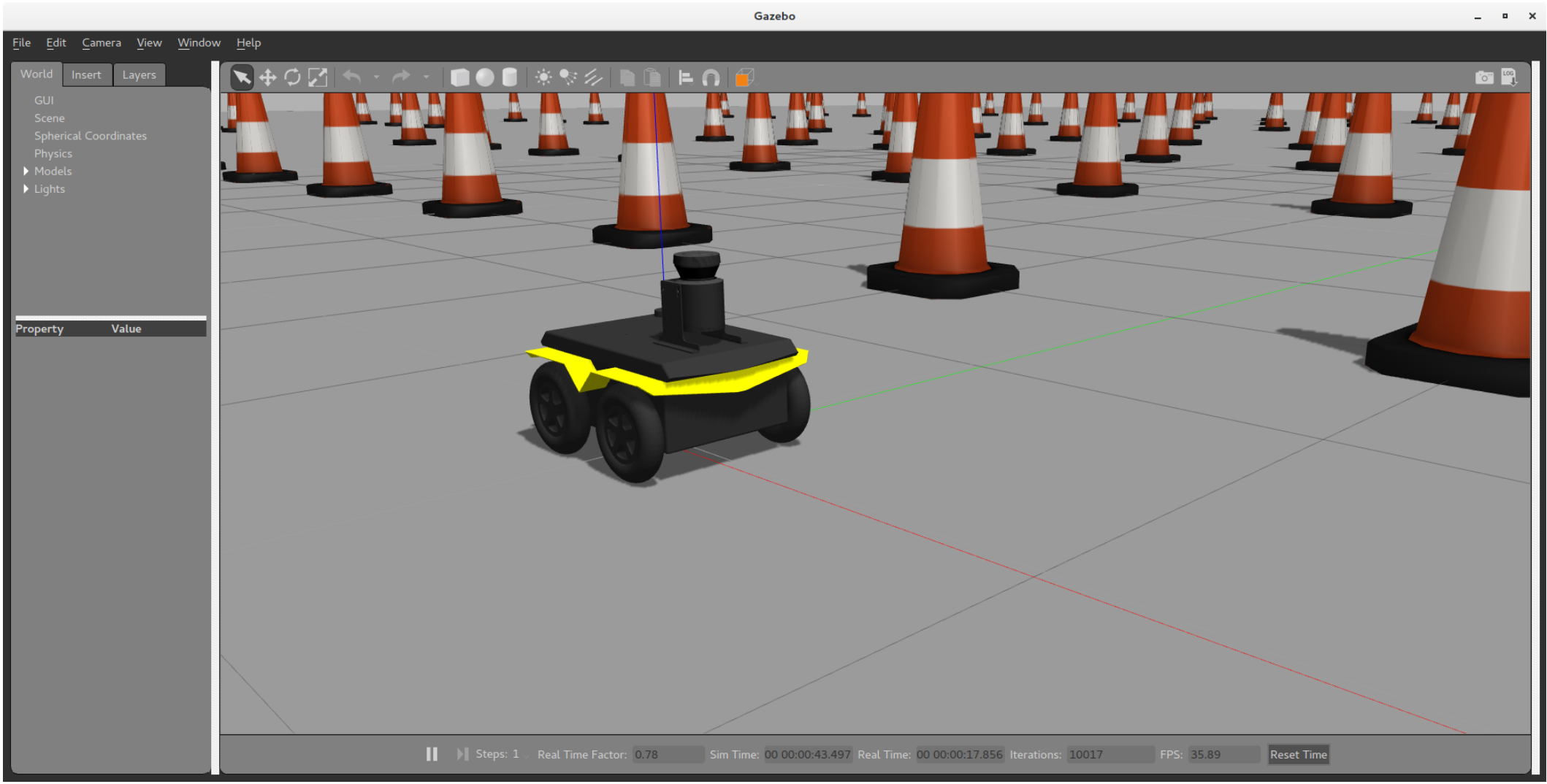
- "Population of models" enables to create collections of models easily. It offers a simple and accurate way of creating structures of rows,columns and arrays of the same model.

- In order to make modifications of the basic world, changes can be made directly on the world file, or via the Gazebo GUI (and then saving the modified world).

- For more information see http://gazebosim.org/tutorials?tut=model_population&cat=

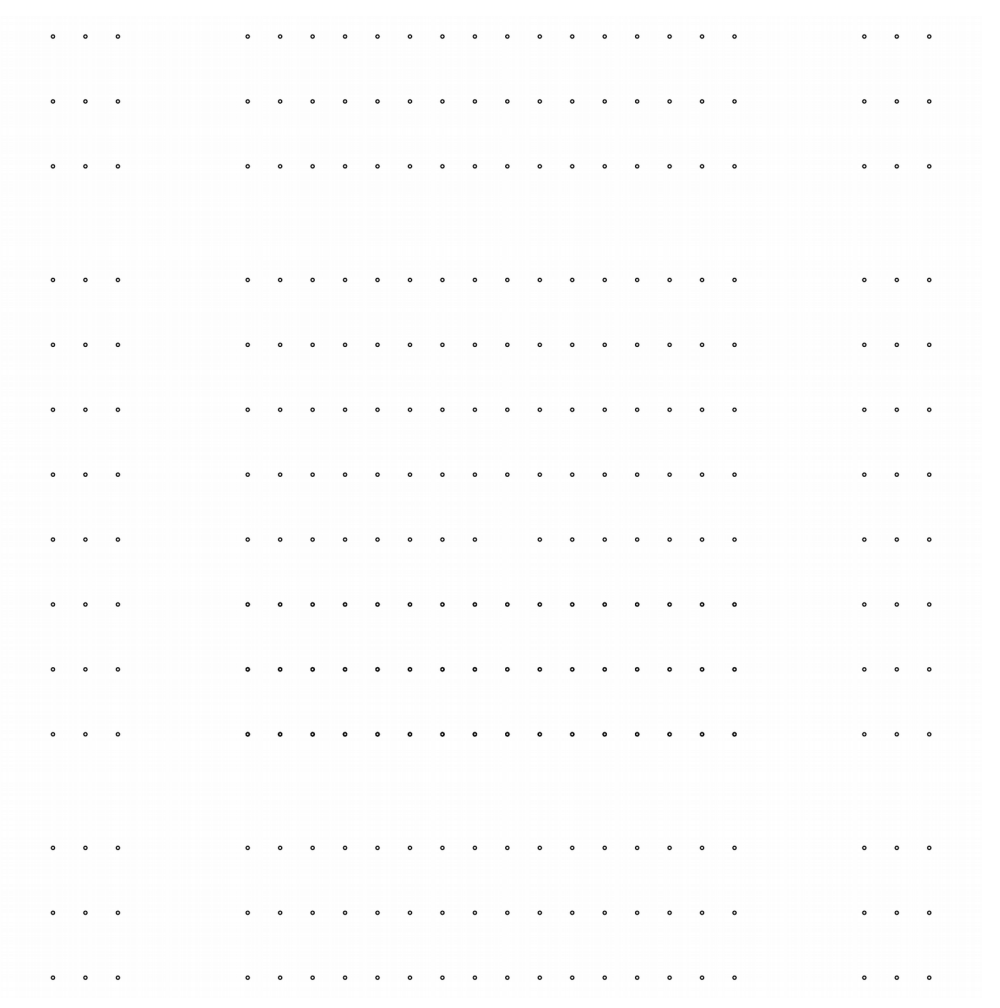


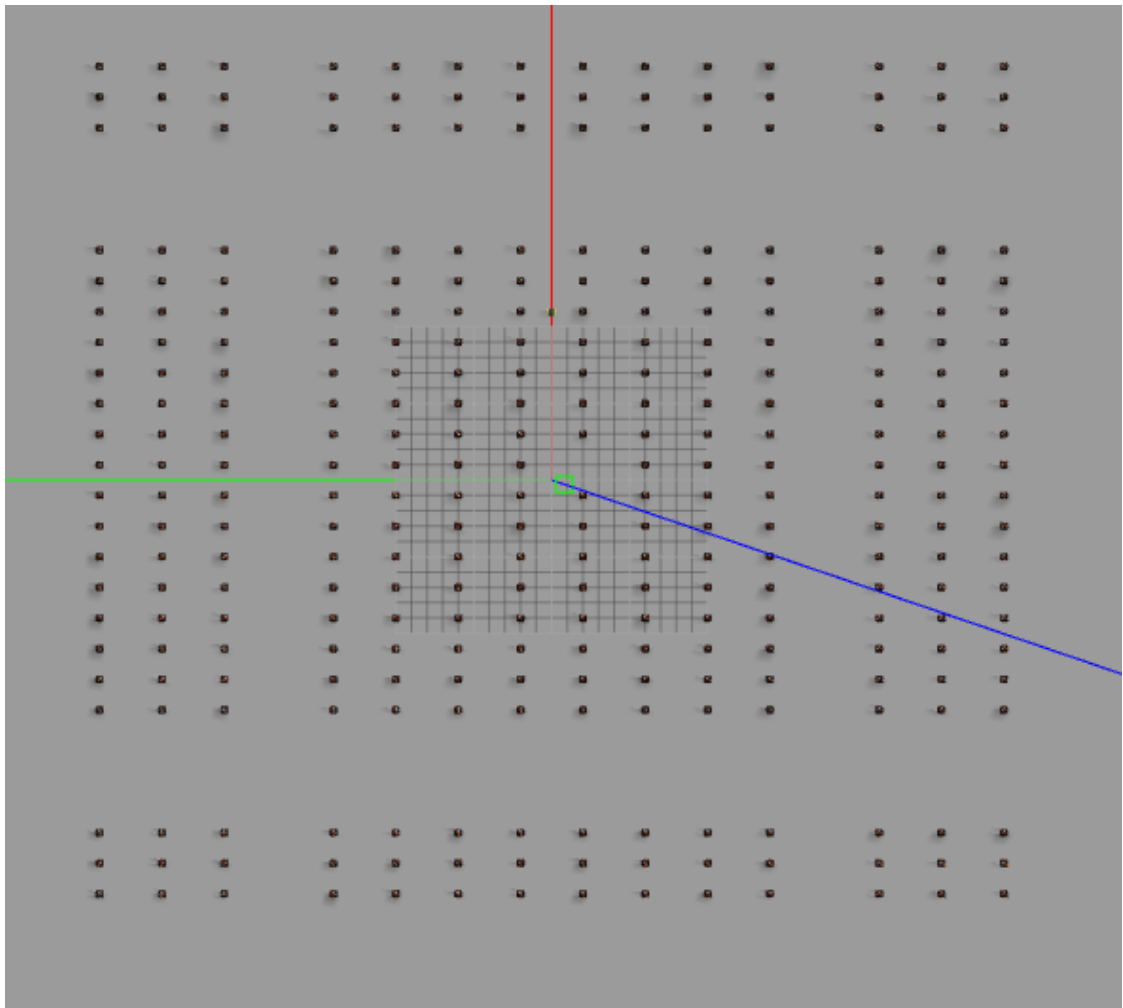
Experiments



Experiment #1 : single cone missing

- **Wold name**: cone_orcahrd5d
- **World Description:** single cone missing, near the origin.
- **Special notes**: 12,000 particles (max).
- **Experiment motivation:** Checking whether a negative landmark (i.e.- an absence of cones) shall lead the AMCL algorithm to a valid localization solution.



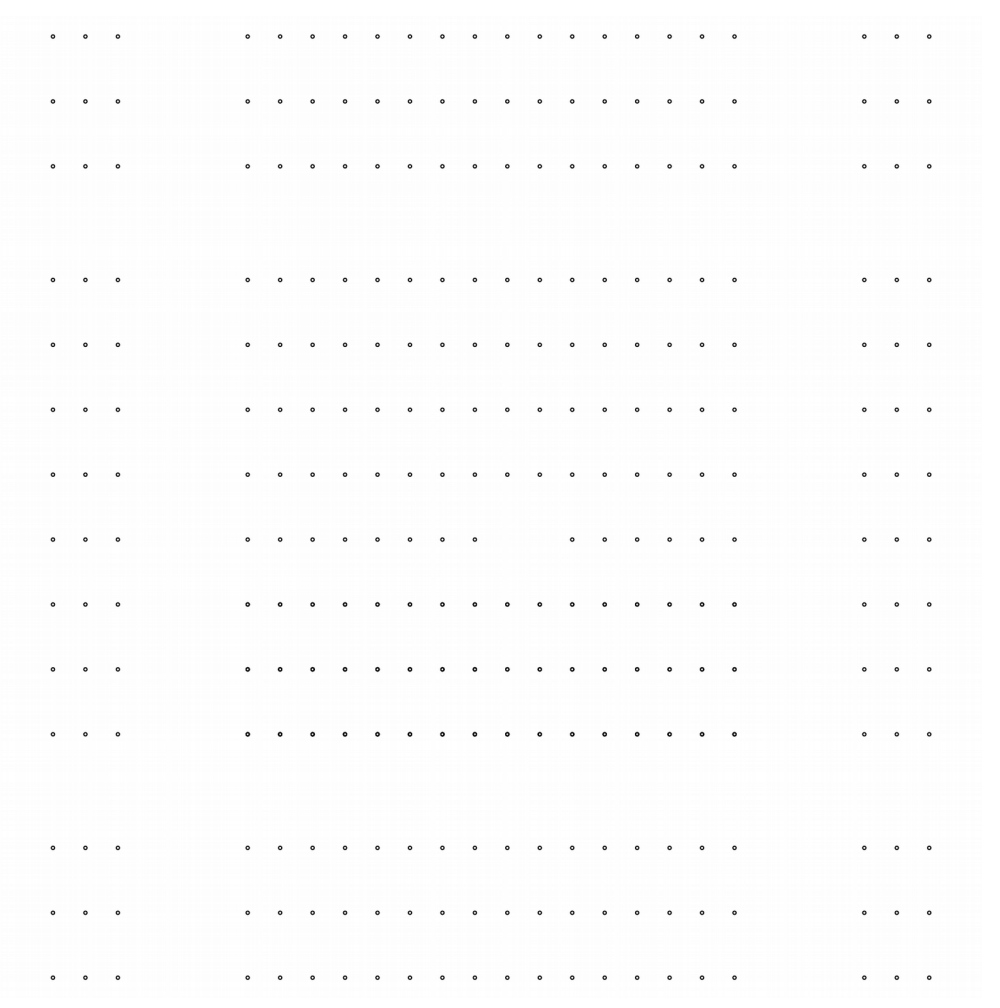


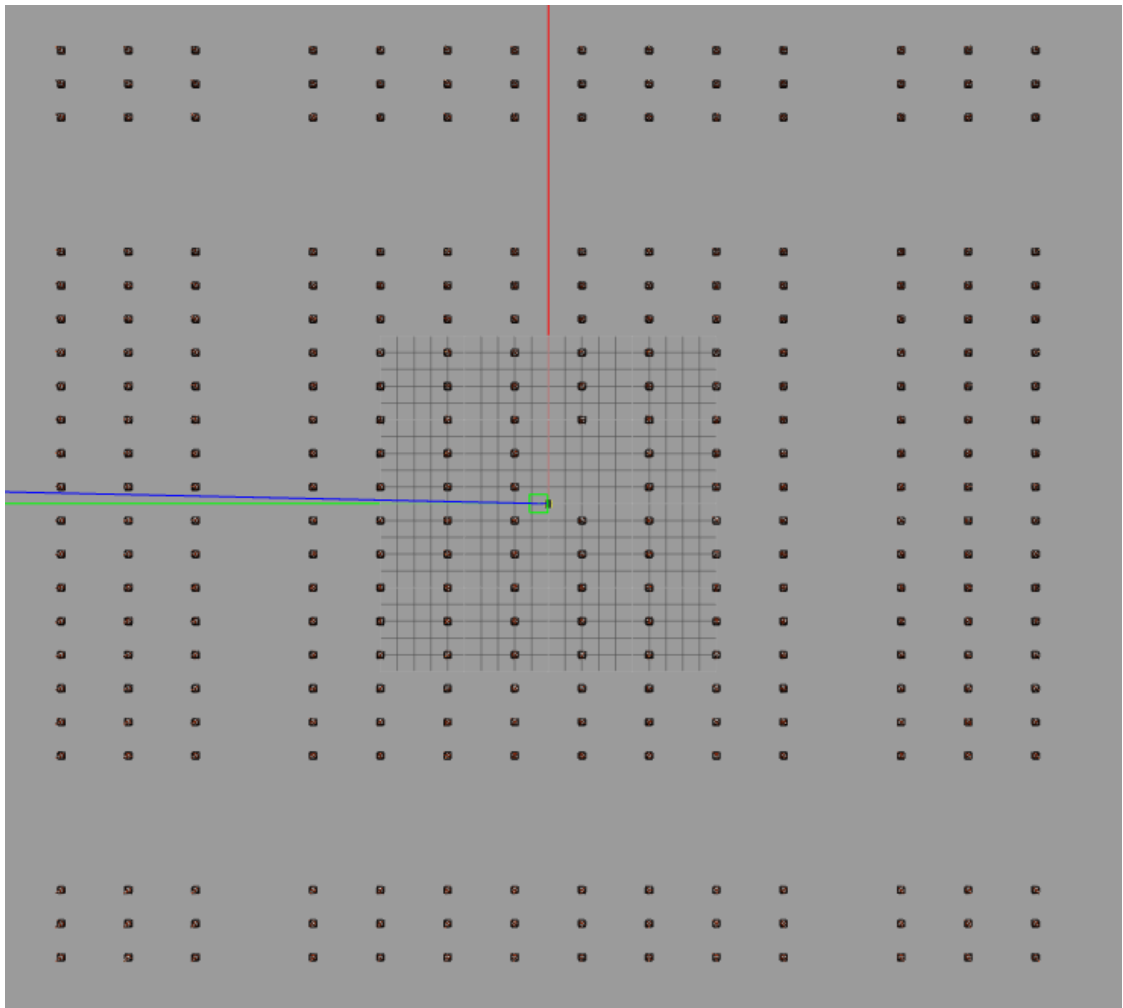
Experiment #1 : single cone missing

- **Results: 7** successive simulations- 0/7(0%) success rate . I stopped there.
- Example success graph:-N.A
- Note: I simulated with larger/smaller number of particles. None succeeded.

Experiment #2 : two cones missing

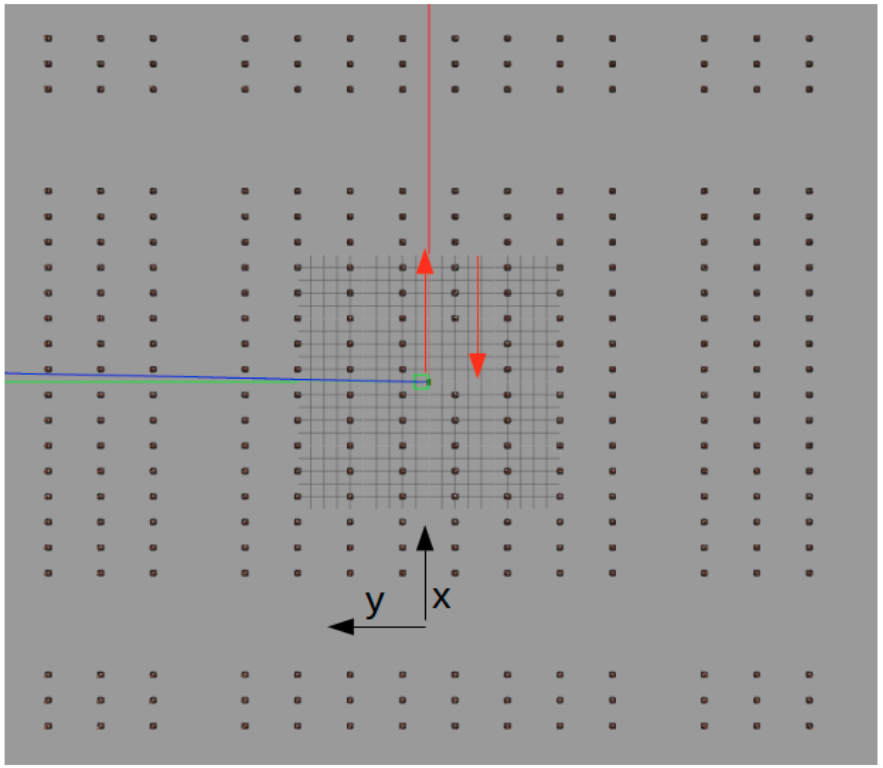
- **Wold name**: cone_orcahrd5b
- **World Description:** Two cones missing near the origin.
- **Special notes**: 12,000-30000 particles (max).
- **Experiment motivation:** Checking whether a negative landmark (i.e.- an absence of cones) shall lead the AMCL algorithm to a valid localization solution.





Experiment #2 : two cones missing

- **Special note**: “Success” in this experiment is considered one of the two possibilities of the AMCL solutions-
  I. Locating the jackal in the row y=0 heading in the positive direction.
  II. Locating the jackal in the row y=-4 heading in the negative direction.
- This is because both cases might look the same form the Jackal’s point of view having only laser scan and the map. The x coordinate must be correct in order to count a “success”.

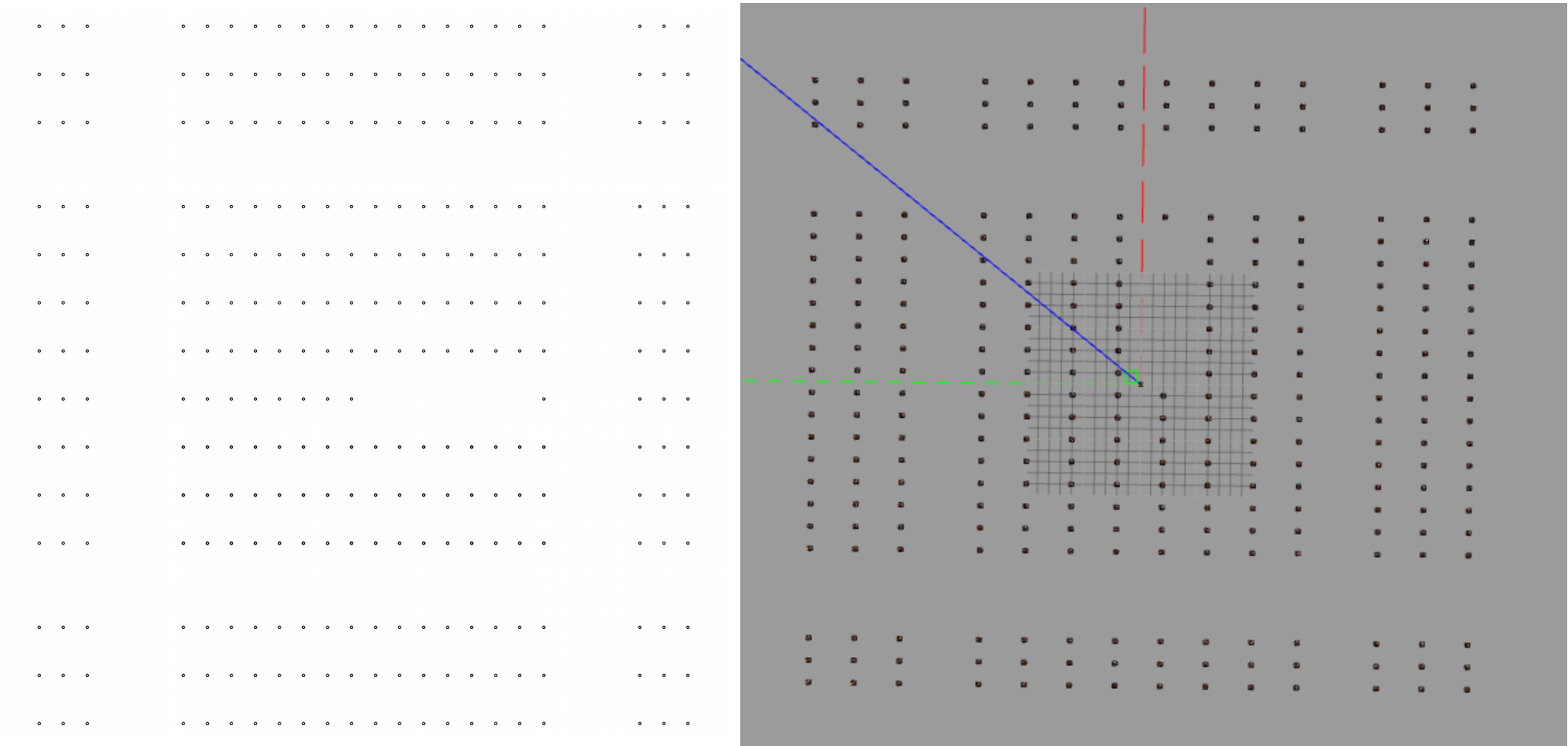


Experiment #2 : two cones missing

- **Results: 22** successive simulations- **0**/22 (0%) success rate.
- Example success graph-None:

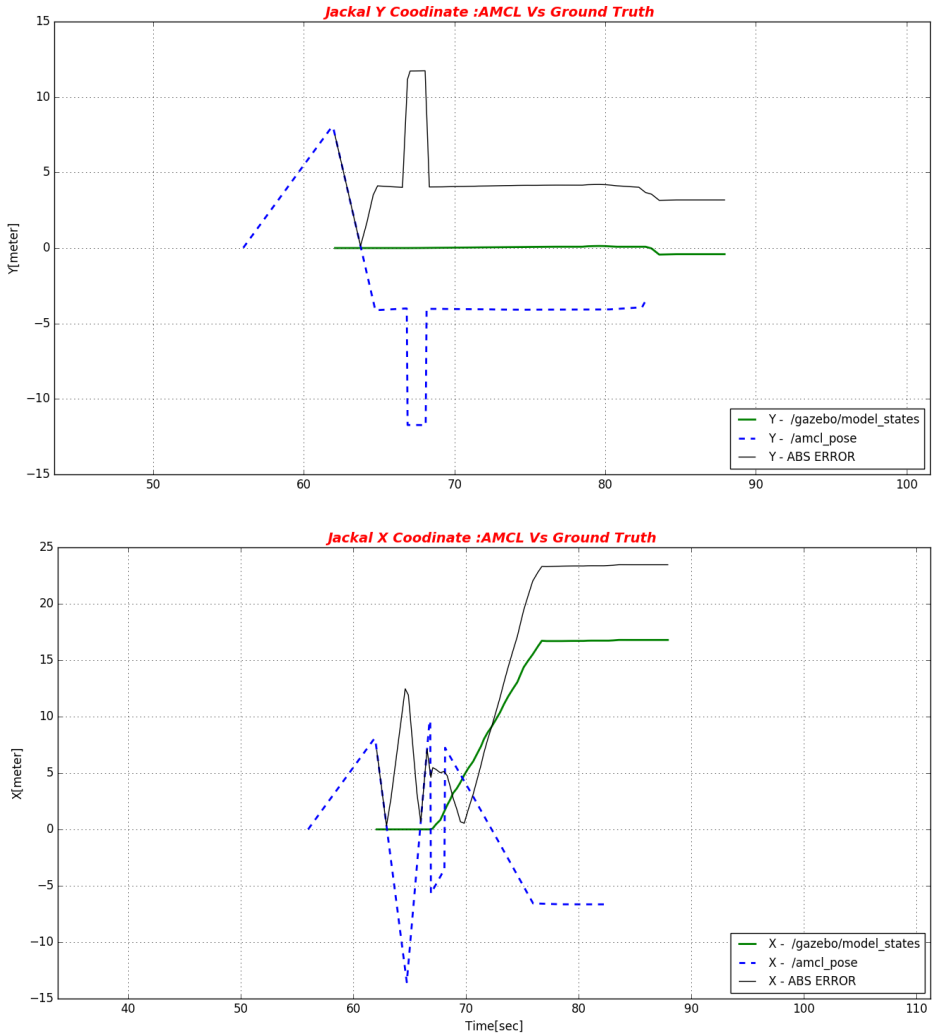
Experiment #3 : half row missing

- **Wold name**: cone_orcahrd5c
- **World Description:** (nearly) half a row of cones missing.
- **Special notes**: 12,000 particles (max).
- **Experiment motivation:** Checking whether a negative landmark (i.e.- an absence of cones) shall lead the AMCL algorithm to a valid localization solution.



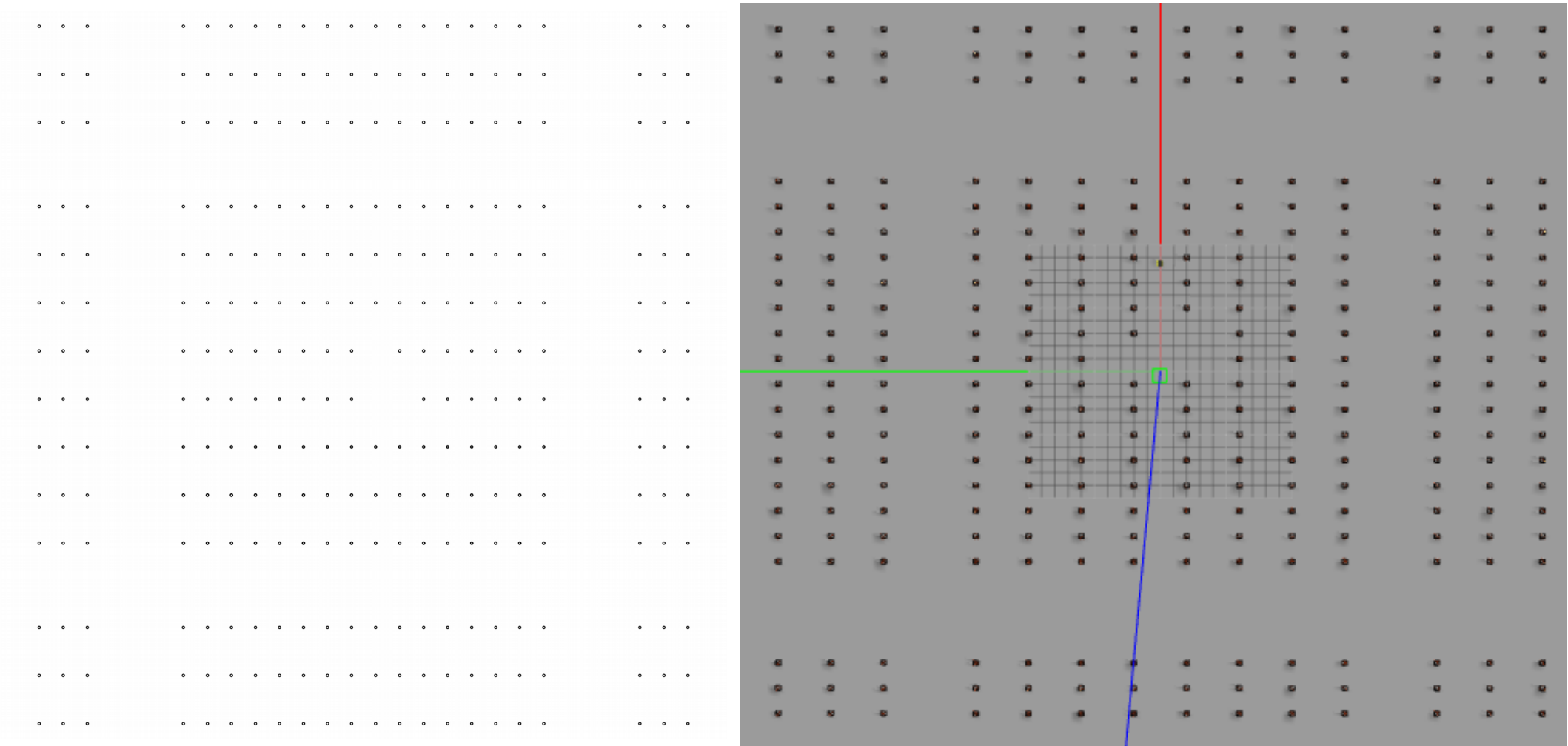
Experiment #3 : half row missing

- **Results:** 10 successive simulations- 6/10 (60%) success rate .
- **Note** : This is a single-axis problem. Again, from symmetry, a negative X-direction velocity and Y=-4 and not 0, is also considered valid, no matters the X.
- Example success graph:



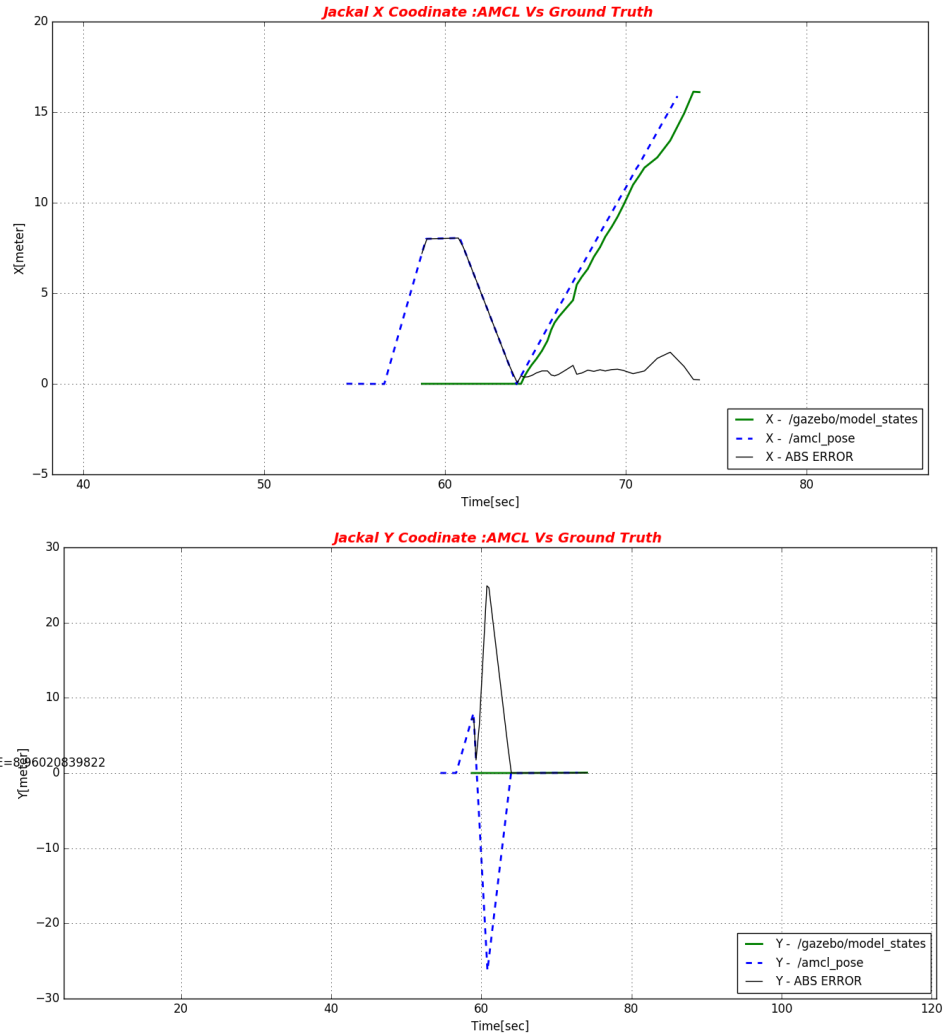
Experiment #4 : 3 cones missing both sides

- **Wold name**: cone_orcahrd5e
- **World Description:** 3 cones missing near the origin, on both sides.
- **Special notes**:. various number of particles tested .
- **Experiment motivation:** Checking whether a negative landmark (i.e.- an absence of cones) shall lead the AMCL algorithm to a valid localization solution.



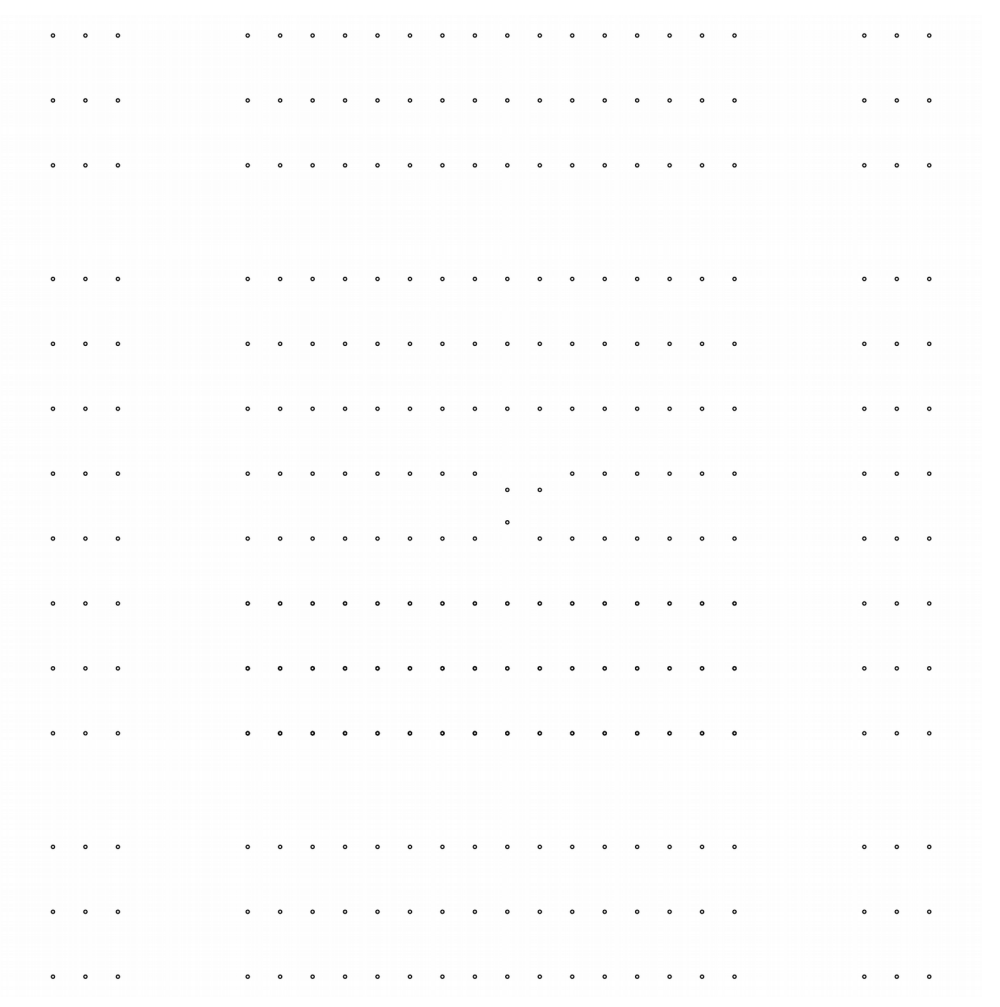
Experiment #4 : 3 cones missing both sides

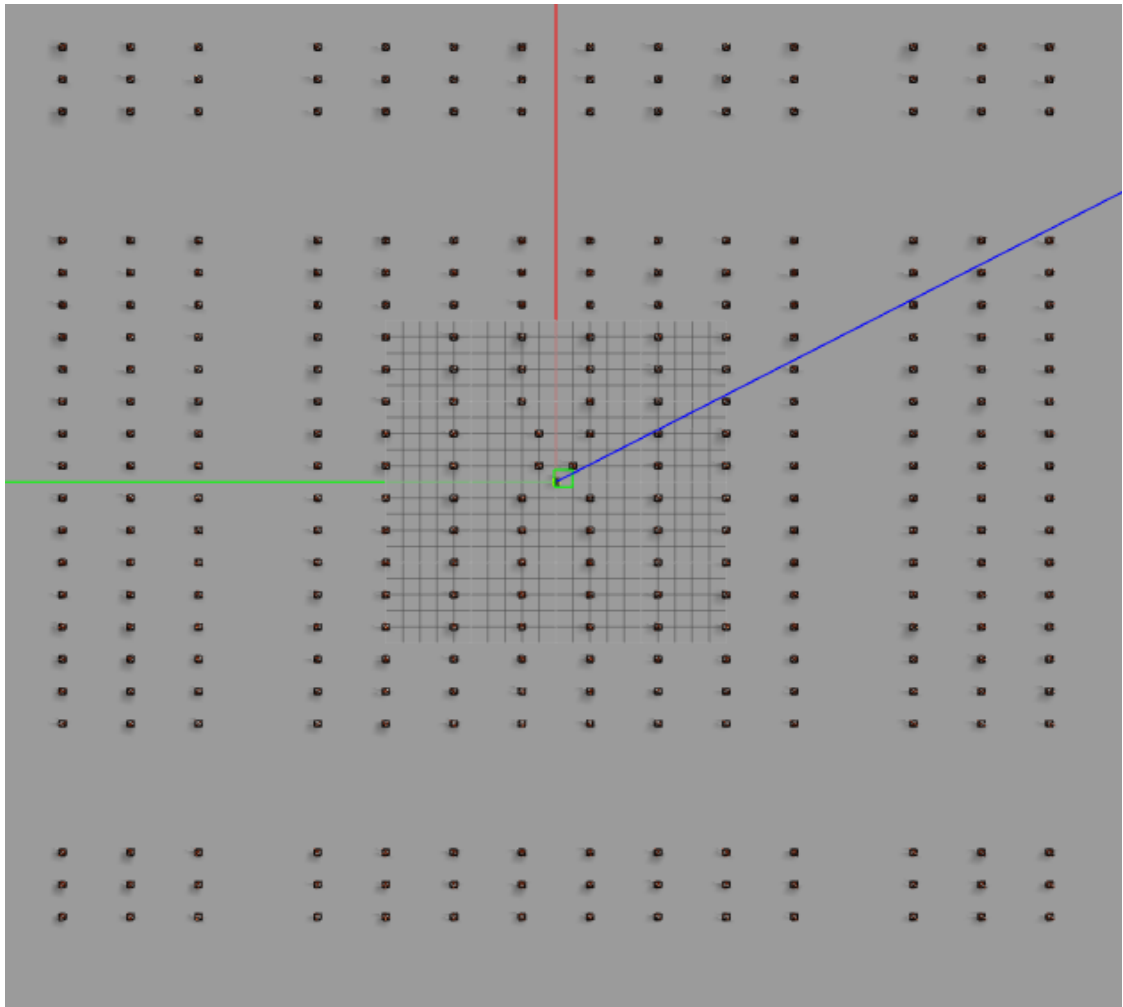
- Success rate- 4/6 with 18000 particles. Other numbers failed (30,000/12000) .
- **<u>The algorithm located the robot correctly in both axis.</u>**
- Example success graph:



Experiment #5 : 3 cones out of line both sides

- **Wold name**: cone_orcahrd6e/6f/6g (several constellations tested)
- **World Description:** 3 cones are 1m ahead, on both sides near the origin.
- **Special notes**: Various number of particles tested .
- **Experiment motivation:** Checking whether an out-of order positive landmark (3 cones) shall lead the AMCL algorithm to a valid localization solution.





Experiment #5 : 3 cones out of line both sides

- Results: 19 successive simulations- 0/19(0%) success rate . I stopped there.
- Example success graph:-N.A
- Experiments within the interval of 8000-30000 particles were made.

Conclusions and insights-experiments 1-4

- **The number of particles is a key characteristic of the AMCL algorithm.** The performance of the algorithm may be much worst than optimal unless this parameter is **carefully chosen by the user**.
- **As a rule of thumb – the more the merrier**. If the number of particles is too low the algorithm obviously yields poor localizations results since the sampling of the map does not cover it in enough detail/resolution.
- **Taking a too high parameter may lead to a heavy load on the hardware resources (CPU usage, memory) , a very long convergence time** . If the landmark is visible for a relatively short time while the convergence time is very long, By the time the algorithm re-sampling process gets to its final steps, the landmark has already become invisible to the sensors a long time before ("obsolete"), and might be ignored. Experiment 4 failed with 30,000 particles but succeeded with 18,000.
- **The best thing to do is to scale this parameter on a given world** (running several trials) in order to get better results. Maybe there is a method for optimizing the number of particles to a given map, but I am not aware of such method. The default number in the navigation stack (2000) was very far form those I found to work best.
- **One must mention that the user must initiate the global localization service himself**. I would expect a good localization algorithm to invoke a "global search" automatically, especially in cases where the map does not fit the measurements for a long (predefined) period of time.

Conclusions and insights-experiments 1-5

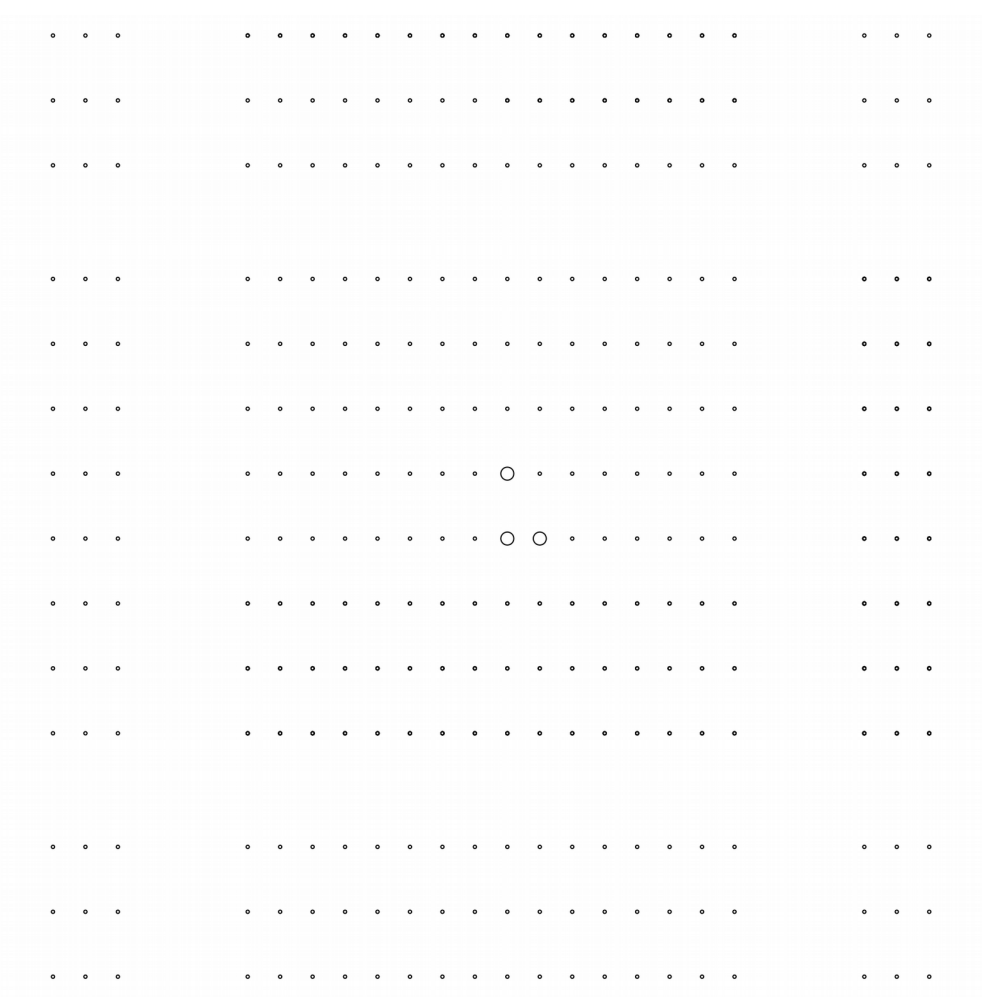
- **Surprisingly, negative landmarks are not good enough .** Having the map and measurements, “The reasonable human being” would have solve the localization problem easily. I would expect that a few missing cones in a small area shall lead the AMCL to a valid localization solution, but this is not the case.
- **Surprisingly, 3 cones out of place on both sides were not enough**. I don’t have a good explanation for that besides determining that the algorithm is rather “spoiled”. Experiment 5 yielded 0% success.

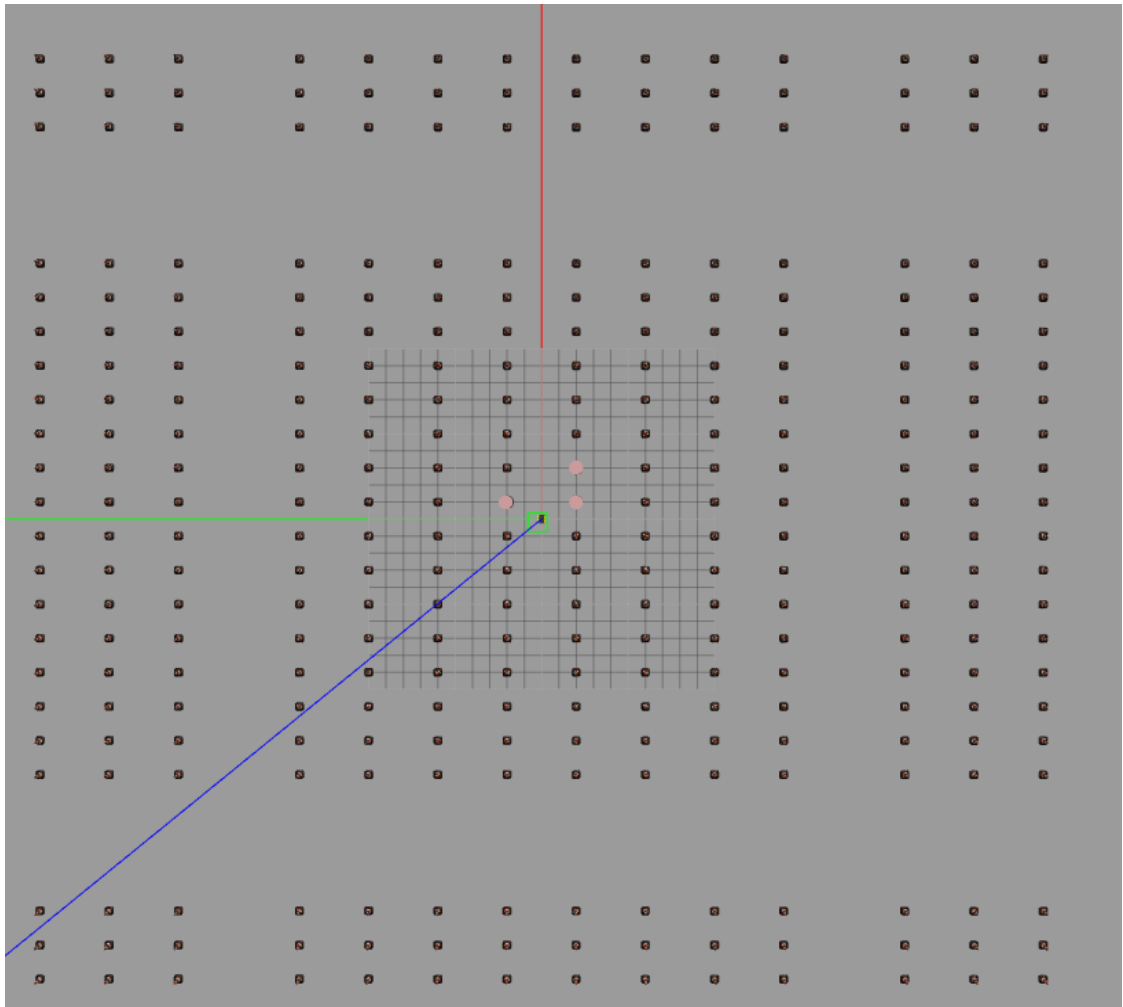
Conclusions and insights-experiments 1-5

- **Surprisingly, negative landmarks are not good enough .** Having the map and measurements, “The reasonable human being” would have solve the localization problem easily. I would expect that a few missing cones in a small area shall lead the AMCL to a valid localization solution, but this is not the case.
- **Surprisingly, 3 cones out of place on both sides were not enough**. I don’t have a good explanation for that besides determining that the algorithm is rather “spoiled”. Experiment 5 yielded 0% success.

Experiment #6 : 3 large cylinders both sides

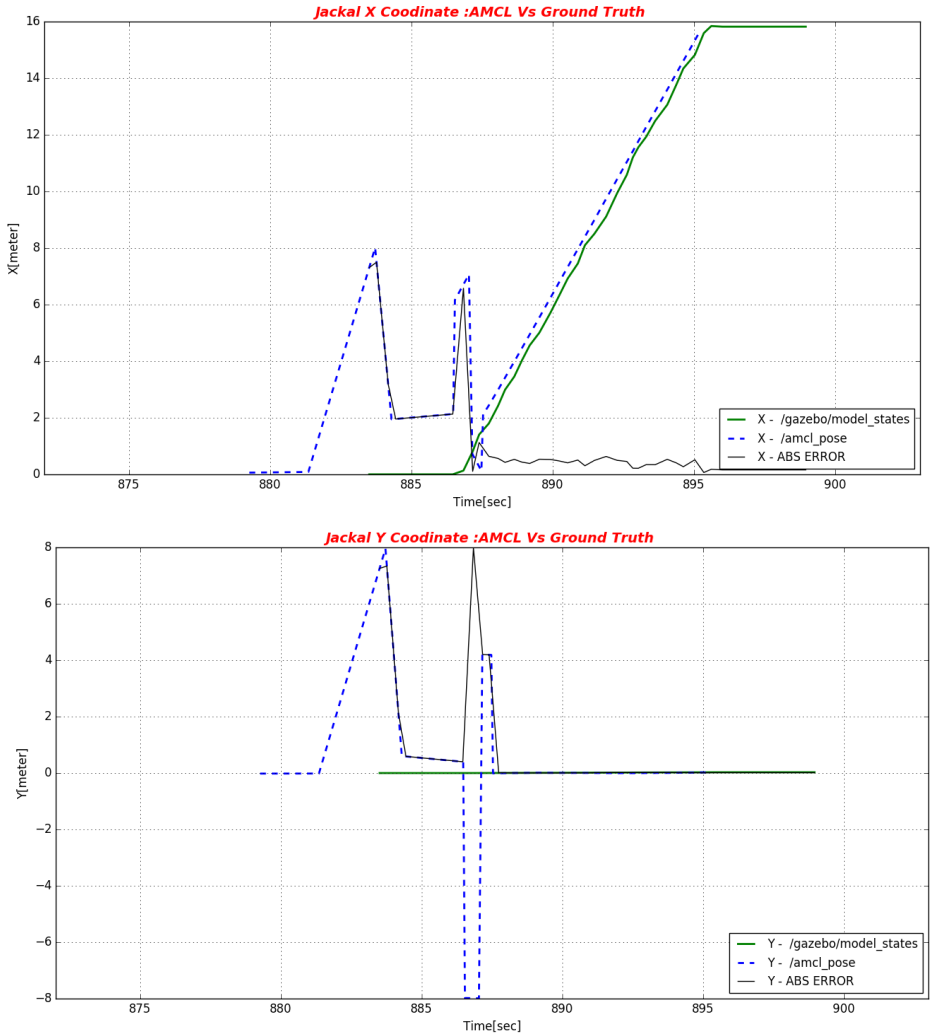
- **Wold name**: cone_orcahrd6d
- **World Description:** 3 cylinders with radius=0.4m near the origin.
- **Special notes**: various number of particles tested .
- **Experiment motivation:** Checking whether distinguishable positive landmark (3 large cylinders) shall lead the AMCL algorithm to a valid localization solution.





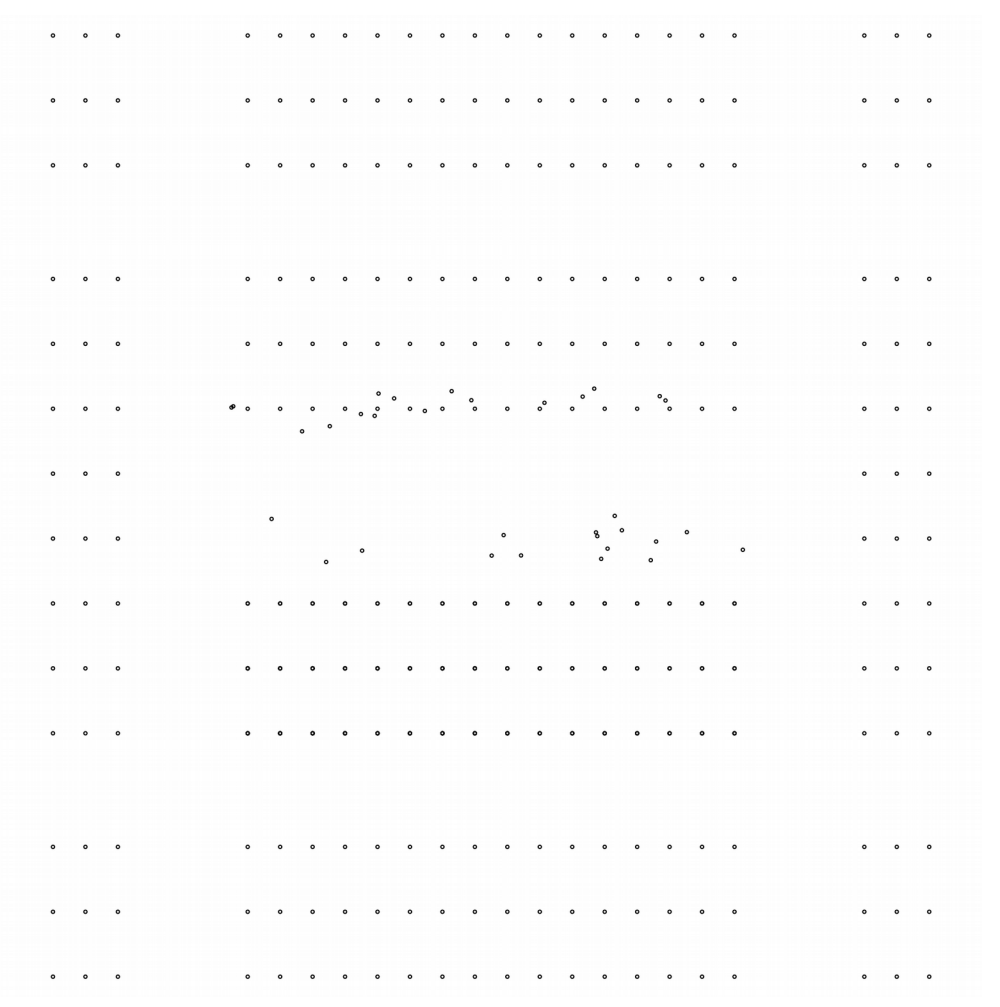
Experiment #6 : 3 large cylinders both sides

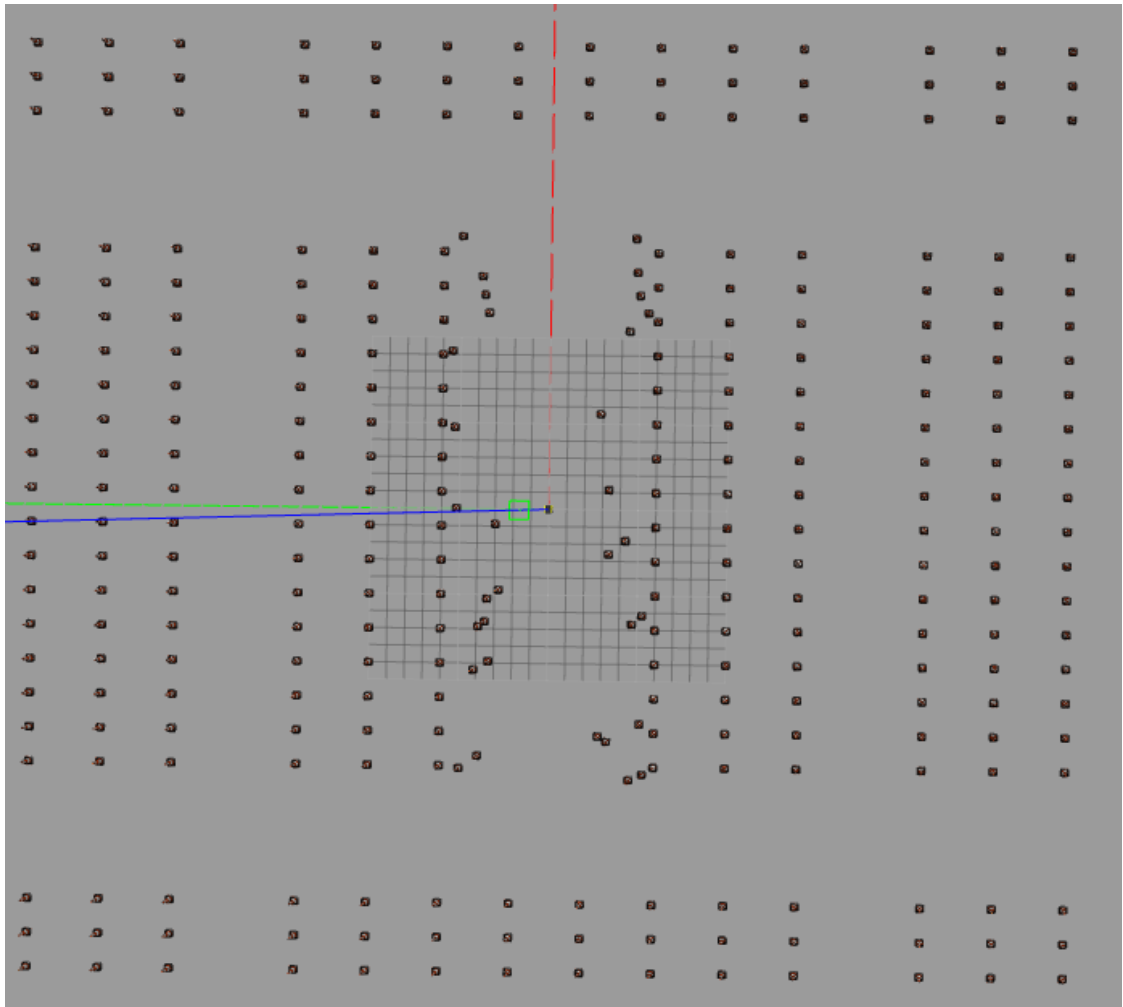
- Results: 5 successive simulations- 5/5 (100%) success rate with 3000 particles.
- 20,000 particles-1/3 success (33%). Using less particles turned to be worse.
- **The algorithm located the robot correctly in both axis**
- Example success graph:



Experiment #7 : two random rows of cones

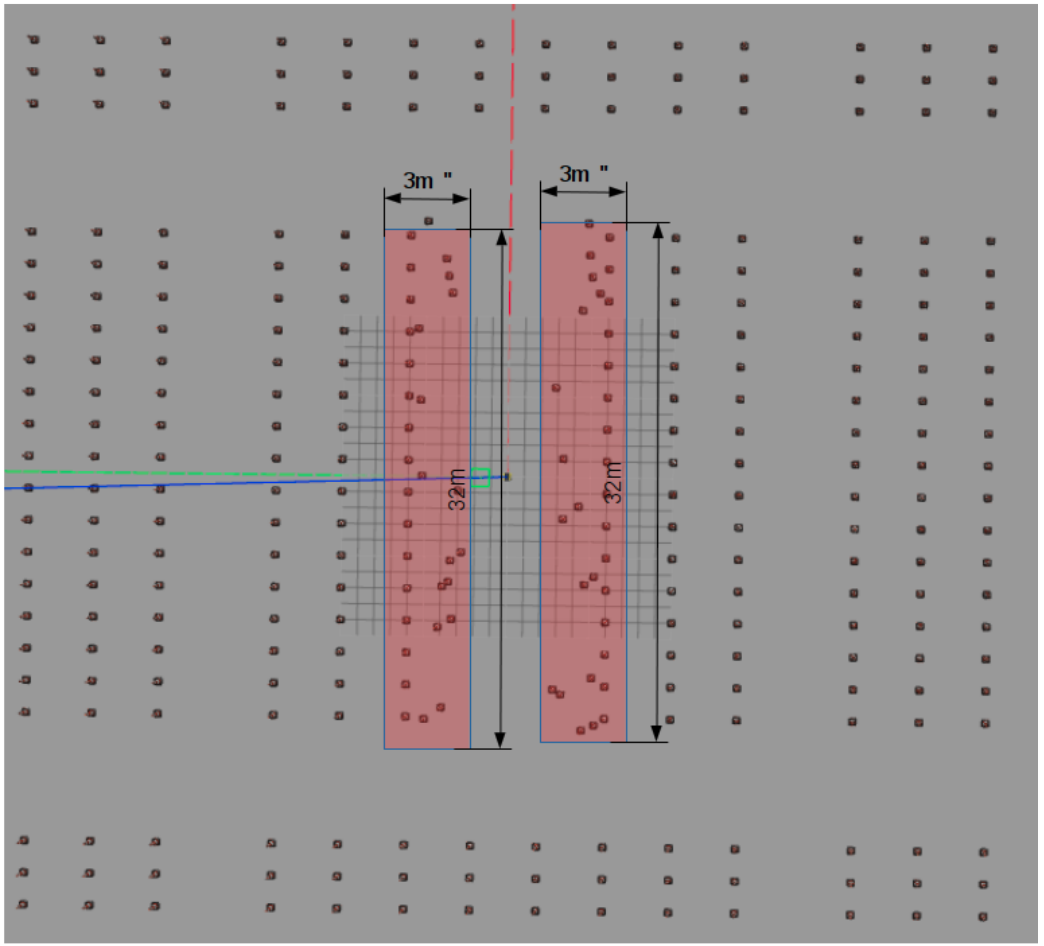
- **Wold name**: cone_orcahrd7b
- **World Description:** two row of cones ordered randomly on both sides of the origin
- **Special notes**: 30000 particles tested .
- **Experiment motivation:** Checking whether large disorder shall lead the AMCL algorithm to a valid localization solution.





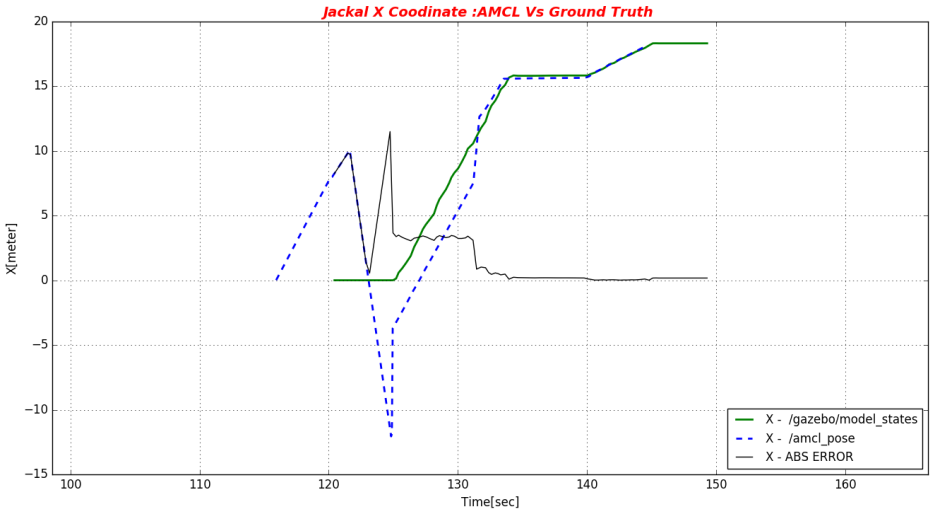
Experiment #7 : two random rows of cones

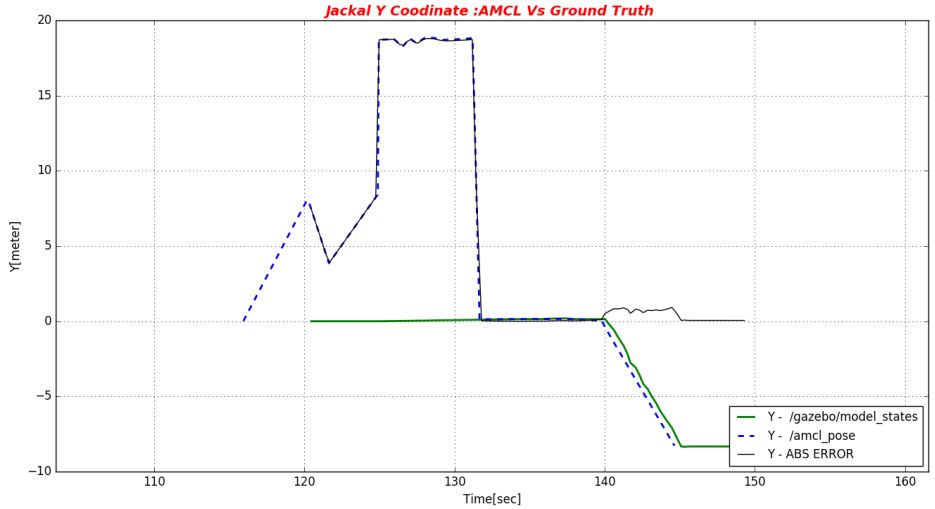
- **Wold name**: cone_orcahrd7b
- **Special notes:** The “random row” was created using gazebo 7 “population of models” feature. See http://gazebosim.org/tutorials?tut=model_population&cat=
- As a result, each experiment has been carried out on a **different** world, using a different **new** map created for that experiment only.
- The map was saved together with the results graphs.



Experiment #7 : two random rows of cones

- Results: 10 successive simulations- 8/10 (80%) success rate with 30000 particles.
- **<u>The algorithm located the robot correctly in both axis</u>**
- Example success graph:





Conclusions and insights-experiments 6-7

- **<u>Distinguishable landmarks work. Large disorder also works.</u>** Experiments 6 and 7 are sort of “control” experiment. I wanted to see if taking very large landmarks or large disorder (both introduce quite large heterogeneity), shall lead to good localization results, and indeed- they did. Although these experiments do not reflect regular orchards, It was important to see if AMCL can work well at all.

World of Cylinders

- **World Description:** In order to simulate better “real life” orchards, I created a world of cylinders instead of cones. The main difference between the two is that the structure of the gazebo’s “ready made” cone model cannot be changed. The Jackal’s laser scan sensor “cuts” a cross section of the cone which results in a circle with a diameter of 0.2m.
- **Special notes**:

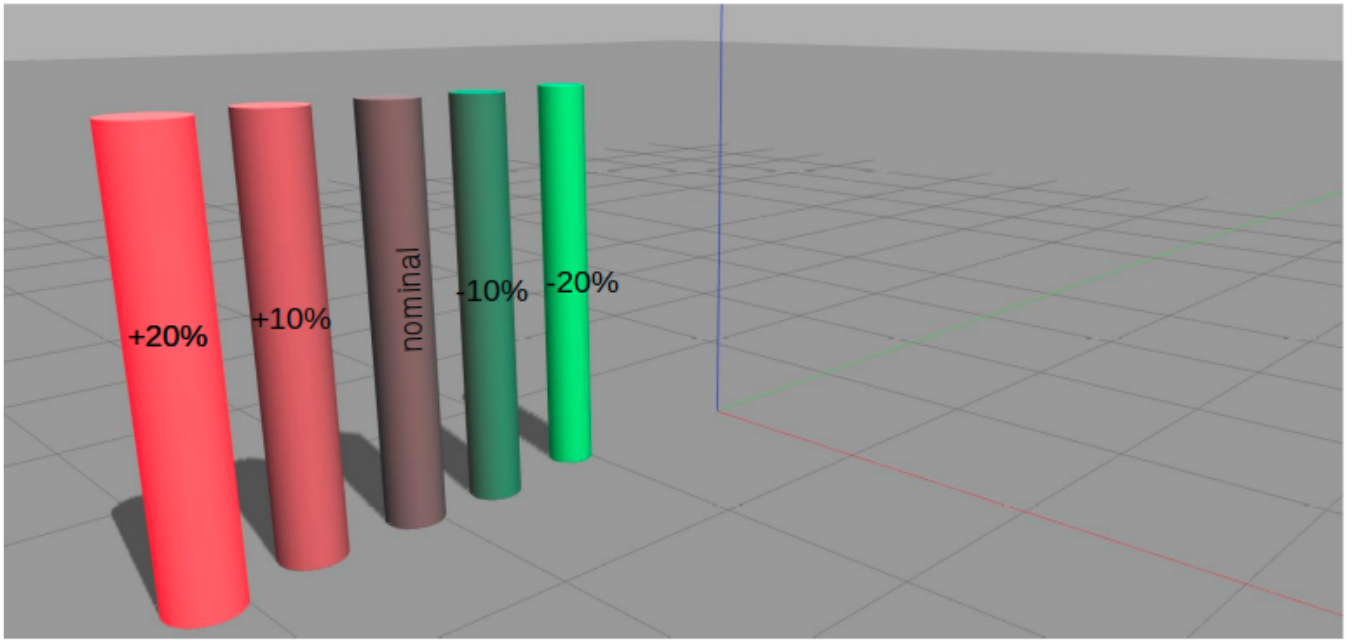
  The trees were simulated using 5 models of cylinders :

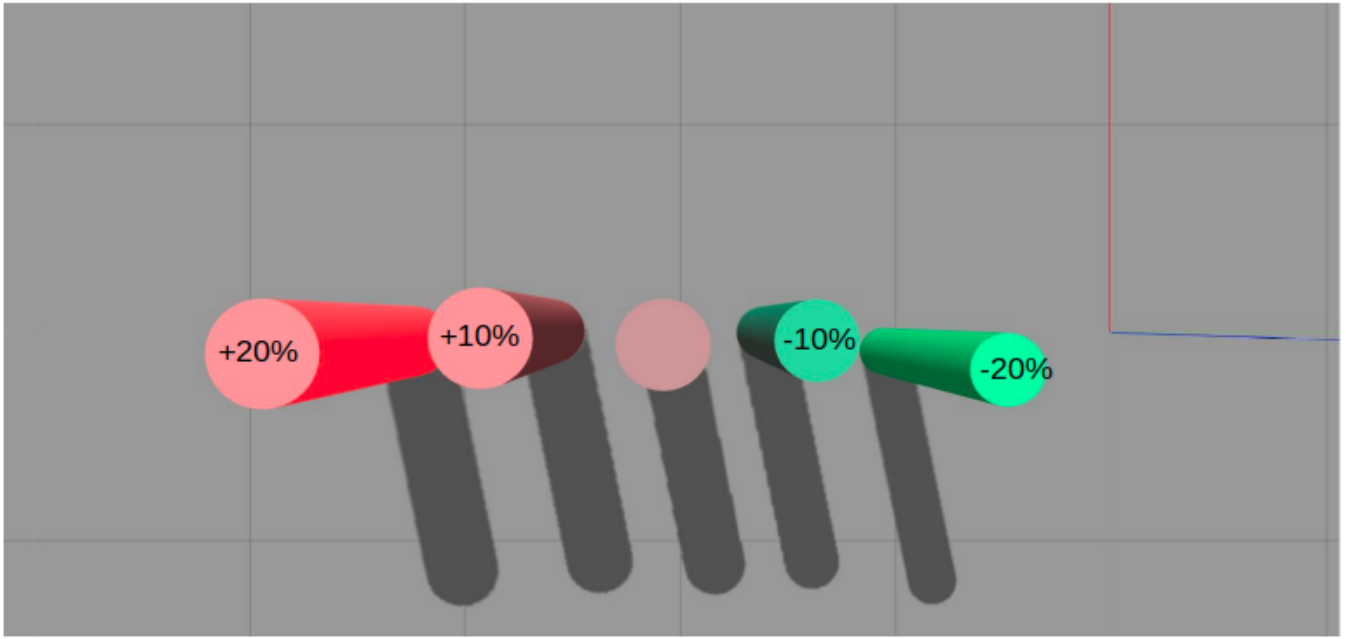
  - Nominal tree with a diameter of 0.28m
  - +- 10 % of the nominal diameter (0.308m , 0.252m)
  - +- 20 % of the nominal diameter (0.224m , 0.336m)
- Each model was colored in a different color in gazebo, in order for me to distinguish between them easily. The colors of do not influence the ground truth map /laser scan in any way.
- The allowed deviation in the location of <u>each tree</u> In the orchard is limited to

  +-20 [Cm] in each coordinate.
- Each cylinder was of course also modeled on the ground truth map.

World of Cylinders

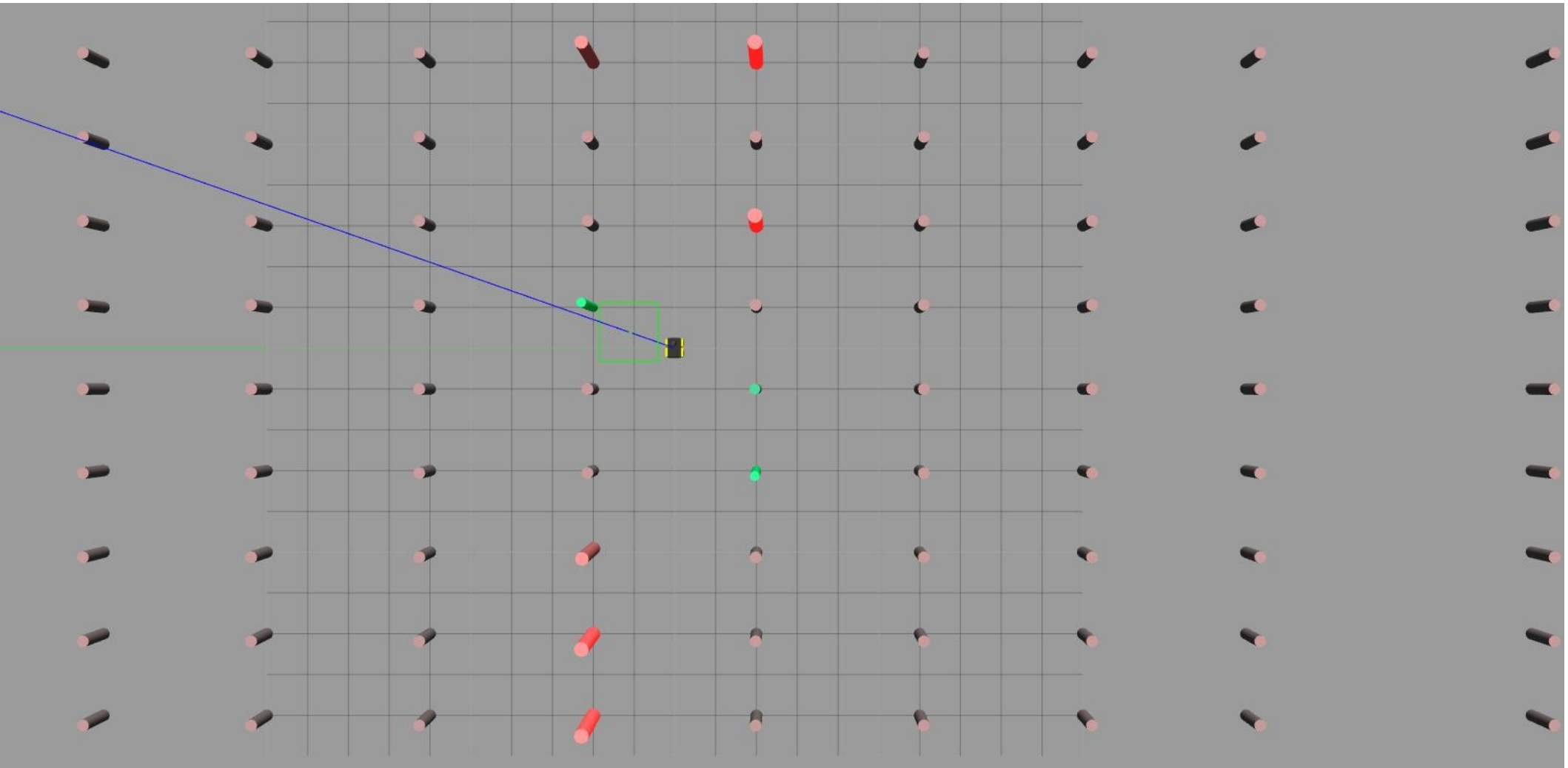
- The cylinders models:





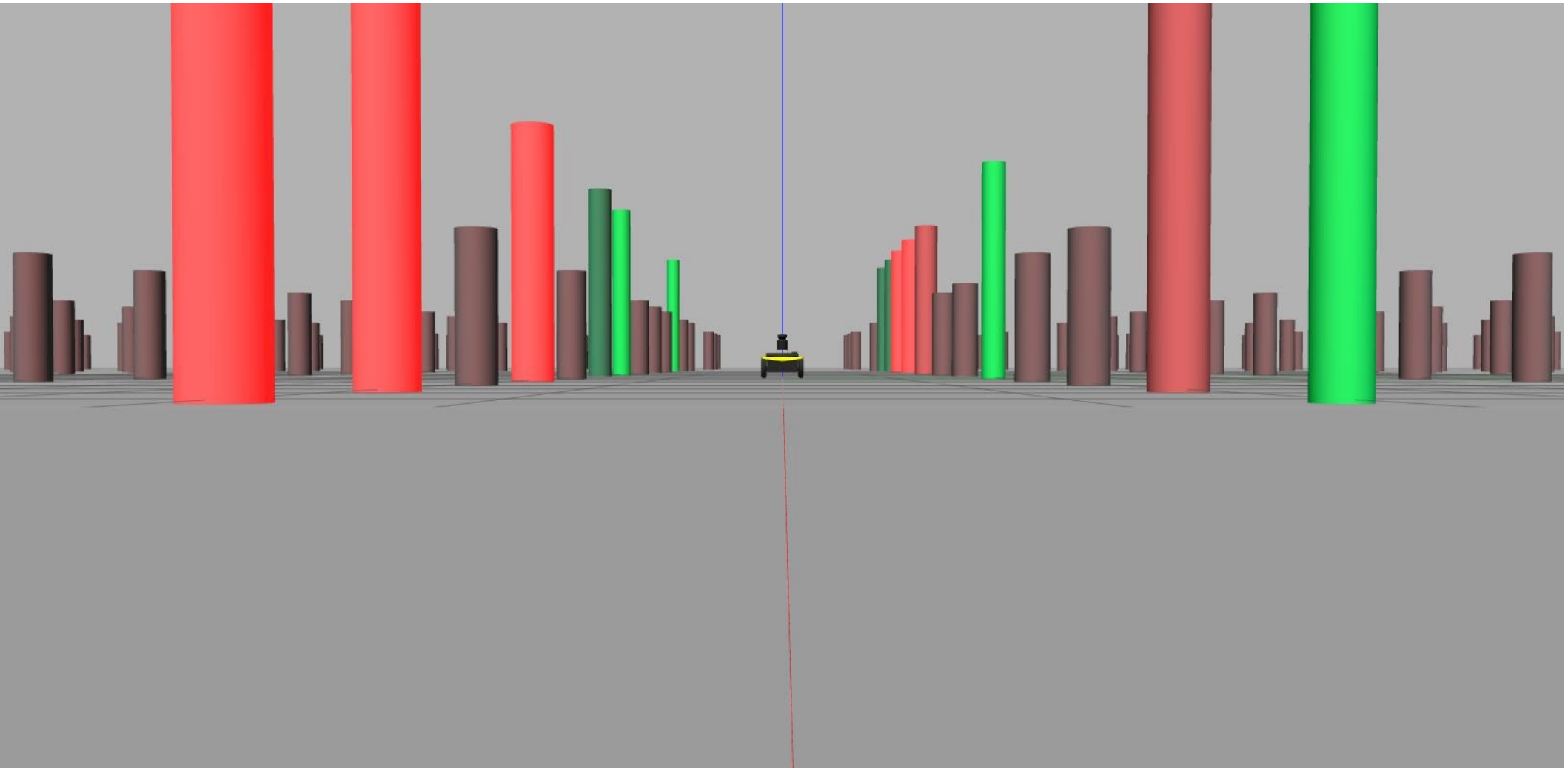
World of Cylinders

- A world example:



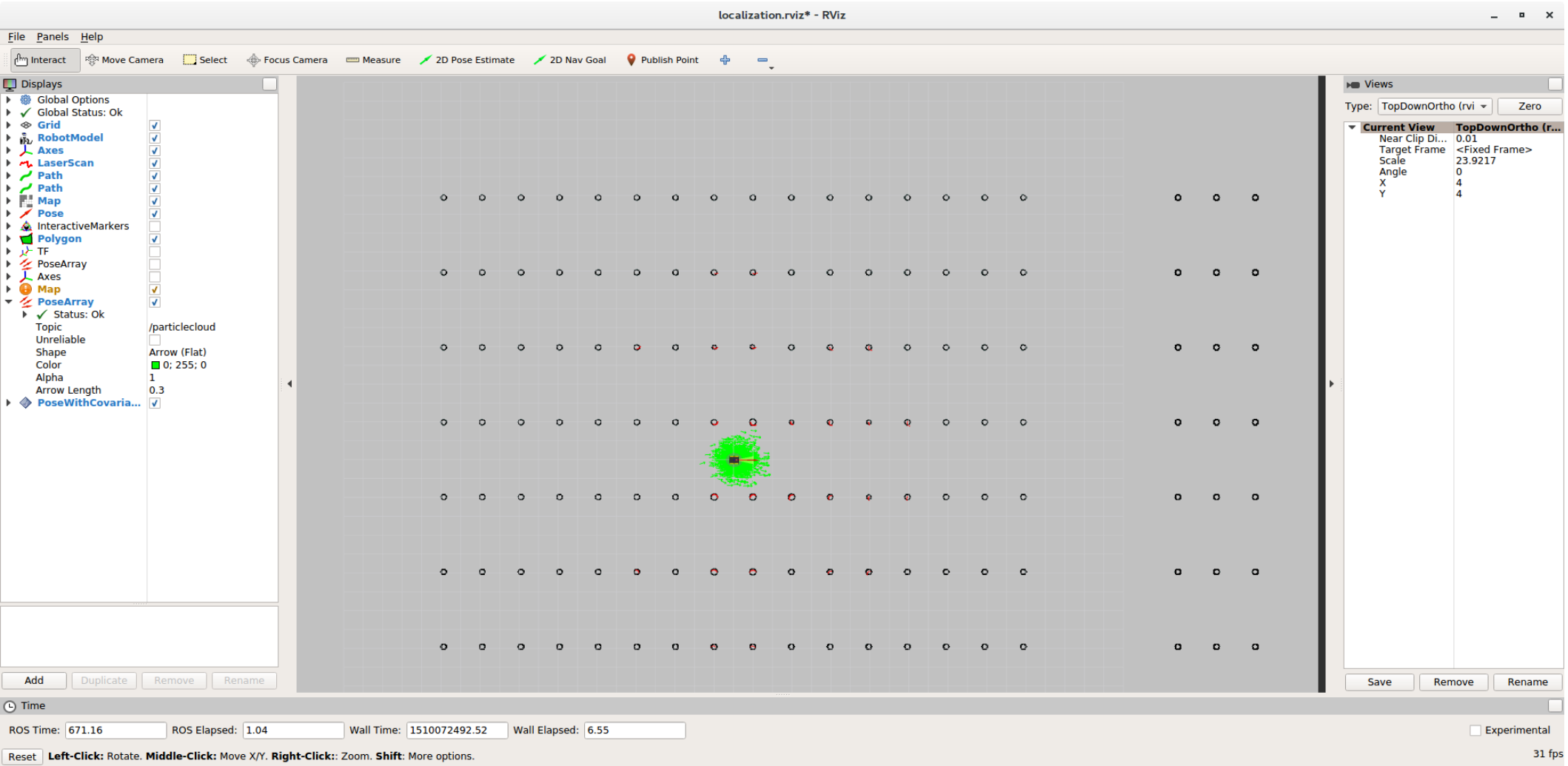
World of Cylinders

- A world example. The height difference is a strange bug in gazebo, all models are at the same height and they all appear on the map and seen by the laser scan.



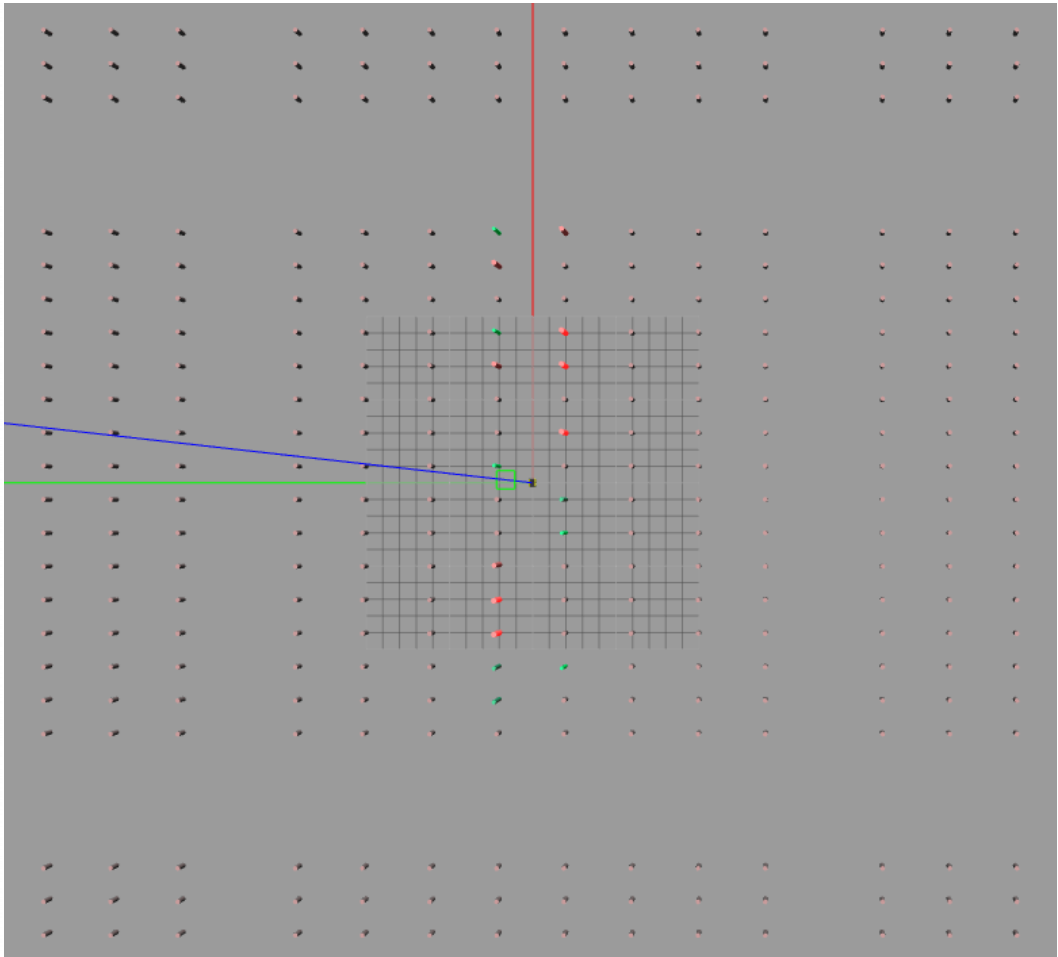
World of Cylinders

- A Rviz window with map example. One can see the various diameters of the cylinders.



Experiment #8 : World of cylinders-original locations

- **Wold name**: cone_orcahrd8b
- **World Description:** mixture of cylinders, on both sides near the origin. The locations were not changed at all, all cylinders at their original positions.
- **Special notes**: 30000 particles tested
- **Experiment motivation:** Checking whether heterogeneity in the cylinder's diameter alone shall lead the AMCL algorithm to a valid localization solution

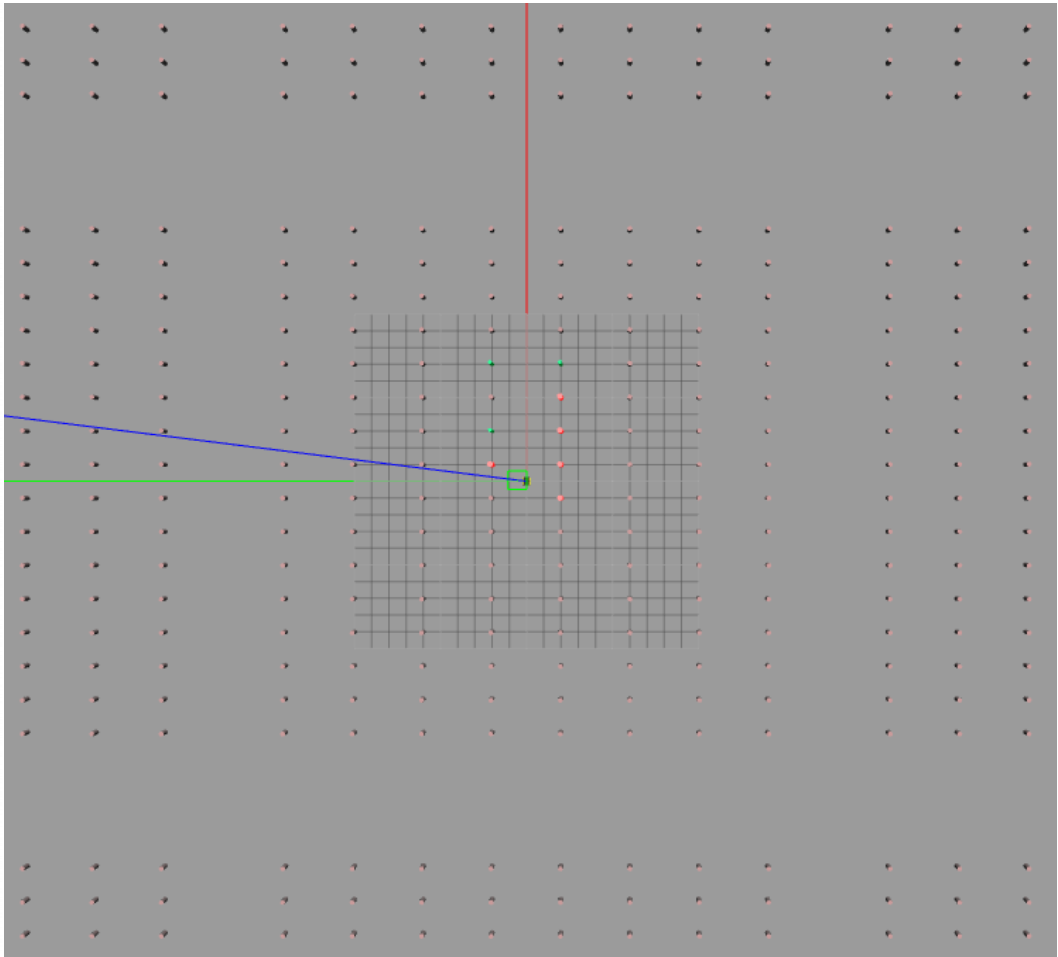


Experiment #8 : World of cylinders-original locations

- Results: 10 successive simulations- 0/10(0%) success rate . I stopped there. In two cases the algorithm managed to get the right Y coordinate.
- Example success graph:-N.A

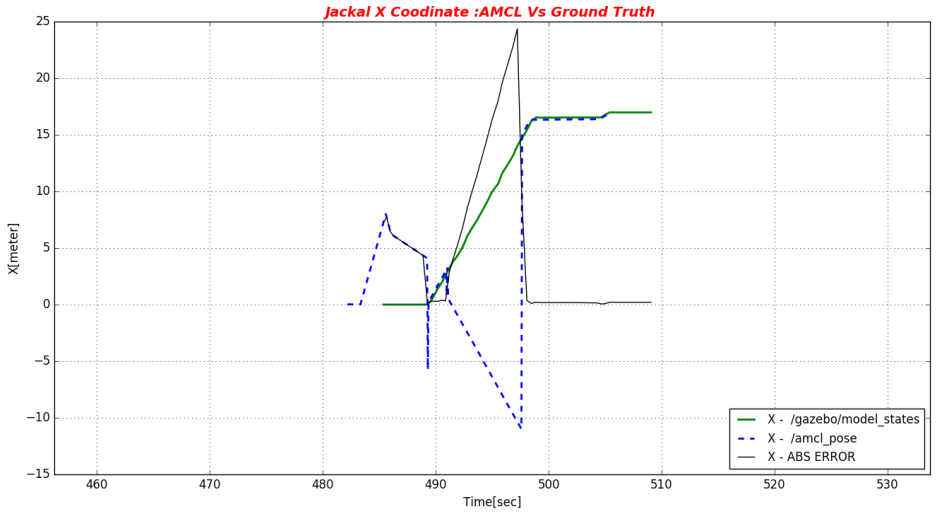
Experiment #9 : World of cylinders-original locations

- **Wold name**: cone_orcahrd8e
- **World Description:** mixture of cylinders, on both sides near the origin. The locations were not changed at all, all cylinders at their original positions.
- **Special notes**: 30000/4000 particles tested
- **Experiment motivation:** Checking whether heterogeneity in the cylinder's diameter alone shall lead the AMCL algorithm to a valid localization solution



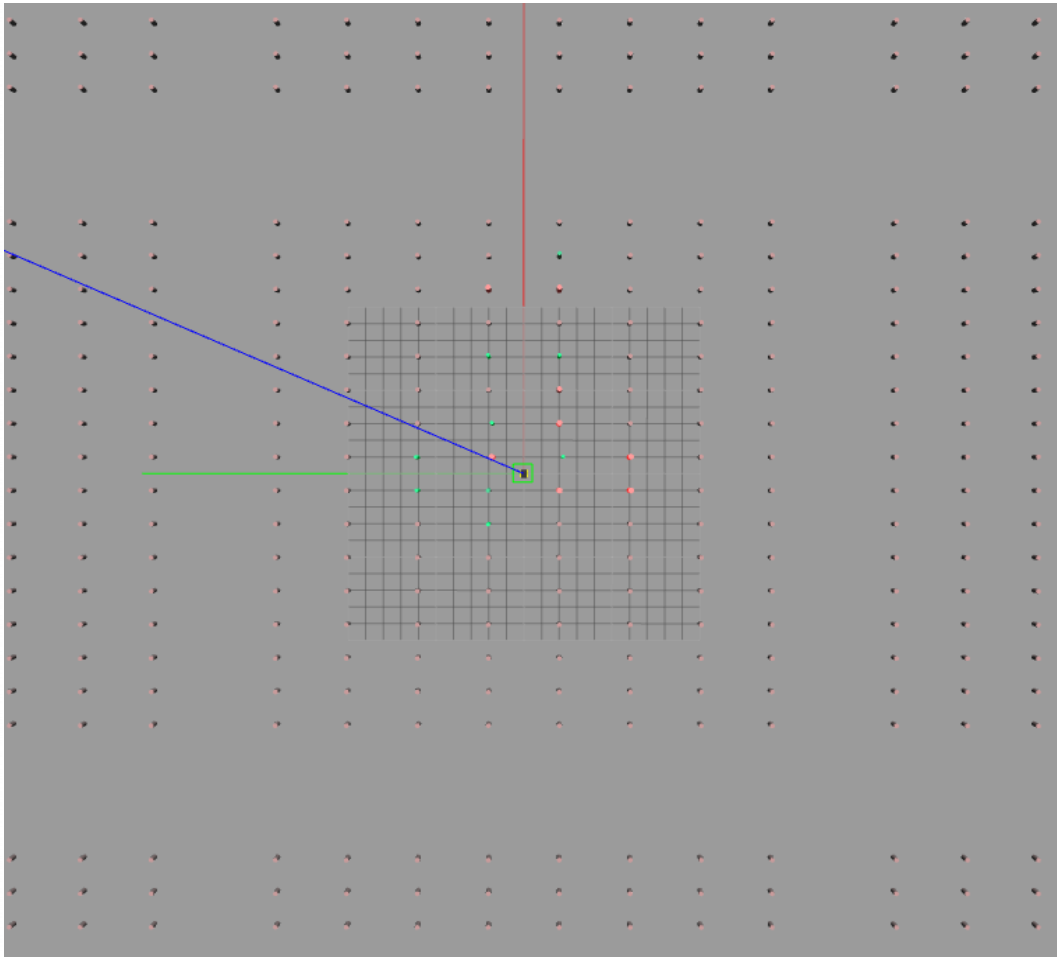
Experiment #9 : World of cylinders-original locations

- Results: 5 successive simulations with 30,000 particles- 0/5(0%) success rate . I stopped there. 5 with 40,000 particles- 3/5 (60%) success in finding the true X coordinate only.
- Example success graph below (X coordinate only)



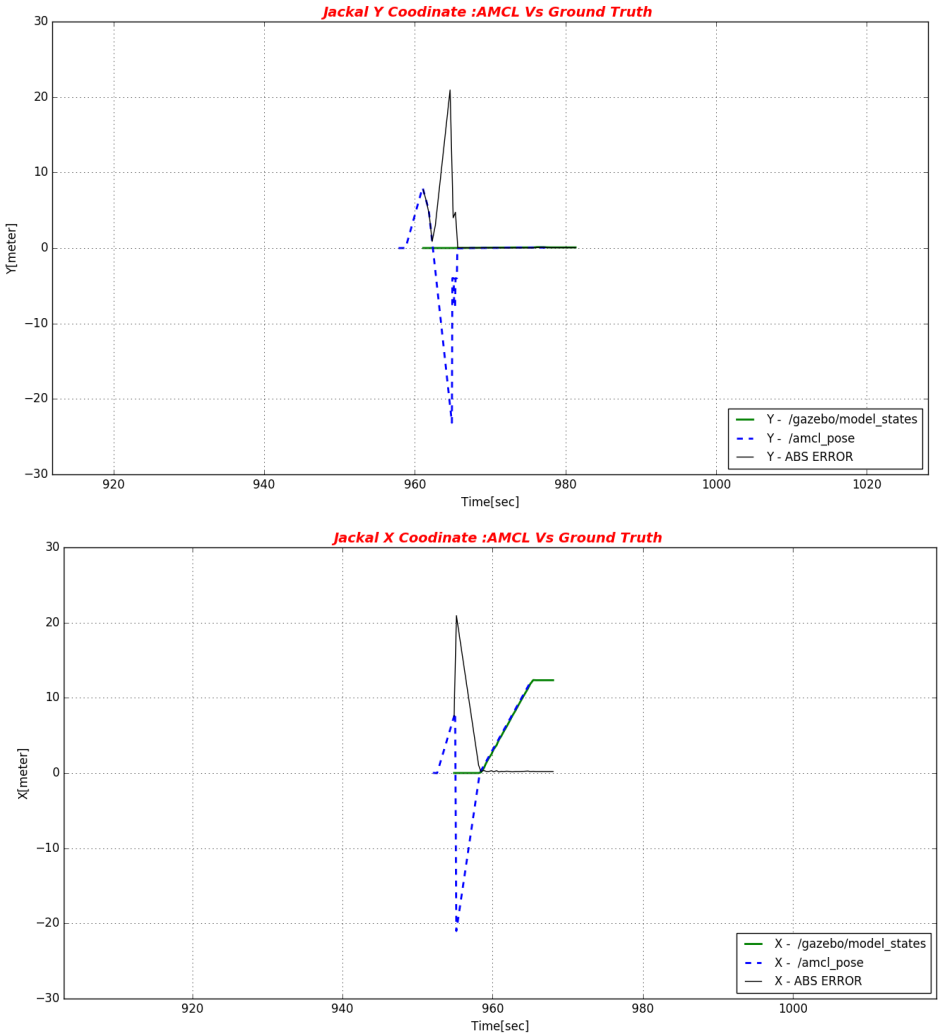
Experiment #10 : World of cylinders

- **Wold name**: cone_orcahrd8ga
- **World Description:** mixture of cylinders, on both sides near the origin. Adding variance of +-20[Cm] to some of the trees location near the origin.
- **Special notes**: 30000/4000 particles tested
- **Experiment motivation:** Checking whether heterogeneity in the cylinder's diameter and locations lead the AMCL algorithm to a valid localization solution



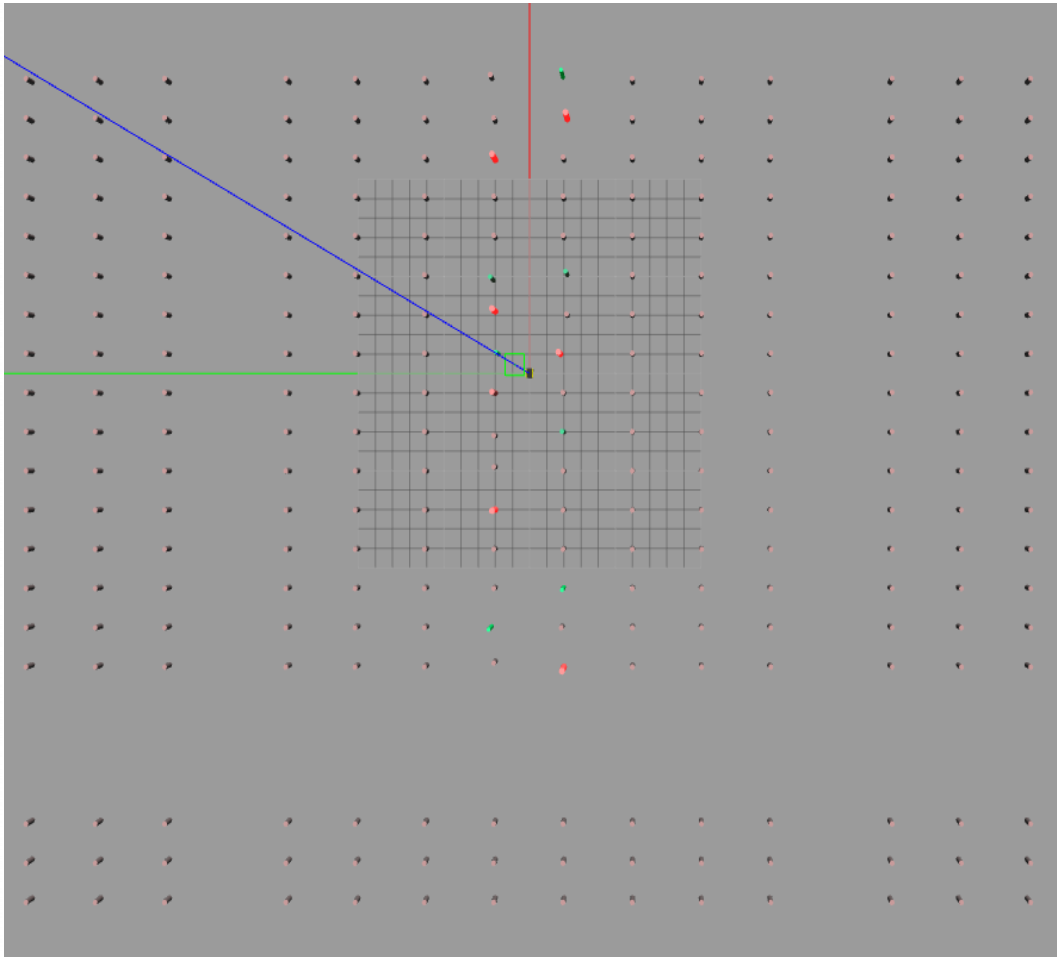
Experiment #10 : World of cylinders

- Results: 16 successive simulations- 8 with 30,000 particles- 4/8 (50%) success in finding the true Y coordinate only. 8 with 40,000 particles- 6/8 (75%) success in finding the true X coordinate only.
- Example success graph below- the graphs are from <u>different</u> experiments



Experiment #11 : World of cylinders

- **Wold name**: cone_orcahrd8gc
- **World Description:** mixture of cylinders, on both sides near the origin. Adding variance of +-20[Cm] to some of the trees location near the origin.
- **Special notes**: 30000/4000 particles tested
- **Experiment motivation:** Checking whether heterogeneity in the cylinder's diameter and locations lead the AMCL algorithm to a valid localization solution



Experiment #11 : World of cylinders

- Results: 14 successive simulations-0/14 (0%) success rate.
- Example success graph- N.A

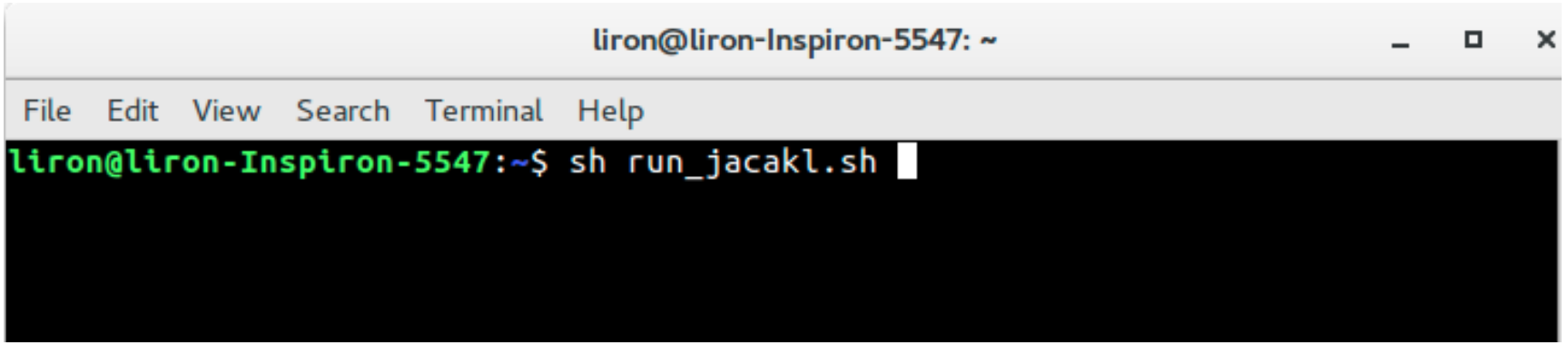
Conclusions and insights-experiments 8-11

- **“Natural” Heterogeneity in the cylinder’s diameter alone is not enough.** From experiments 8 and 9 it is quite clear that the variance of +-20% in the cylinder’s diameter is not enough for the AMCL to determine the robot’s location. The cylinders looks nearly identical in the map and to the laser scanner.
- **Adding variance in the cylinder’s coordinates (+-20cm) is also not enough, but may improve the results.** From experiment 10 we can see that changing the locations of the cylinders may lead to some good “half” solutions (meaning, finding the true X **or** Y coordinate). Yet, even with 40,000 particles (where my PC was sweating) – a valid location solution in both coordinated was never found despite the larger heterogeneity introduced by moving the cylinders from their original positions. Experiment 11 is similar to 10 but yielded null results. It shows that the even the partial success of experiment 10 is still “fragile” and highly dependent on a specific arrangement of the cylinders. **In short, The heterogeneity introduced by +-20cm in location and +-20% of cylinder’s diameter is simply too small for the AMCL.**

Project Structure and general info

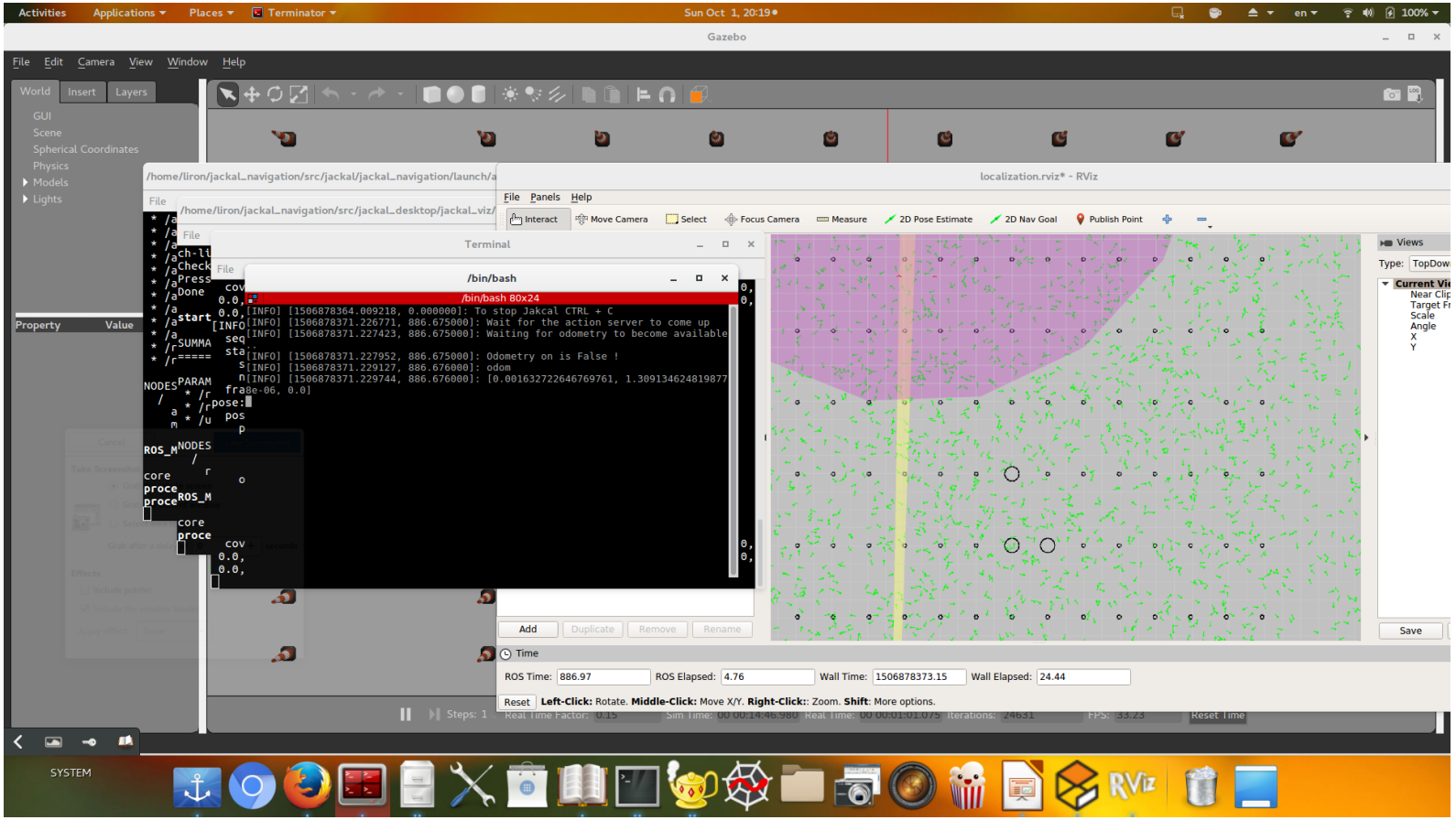
- All project files are in jackal_navigation dir, inside linux user's home directory.
- Directory contents in based on jackal navigation stack from CLEARPATH® website

  https://www.clearpathrobotics.com/assets/guides/jackal/simulation.html downloaded from source
- Project is running on **UBUNTU 16 (Xenial)** using **ROS Kinetic** and **Gazebo 7.2**
- **Single Command running**
  - There are two linux scripts inside the user's home directory:
    - *run_jacakl.sh* - calls all launch files, python scripts, etc.
    - *mybash.sh* – used from creating a new map. Usually called from *run_jacakl.sh* and not by the user.
  - Please install Terminator software on your machine as the scrips uses it.
  - Running example:



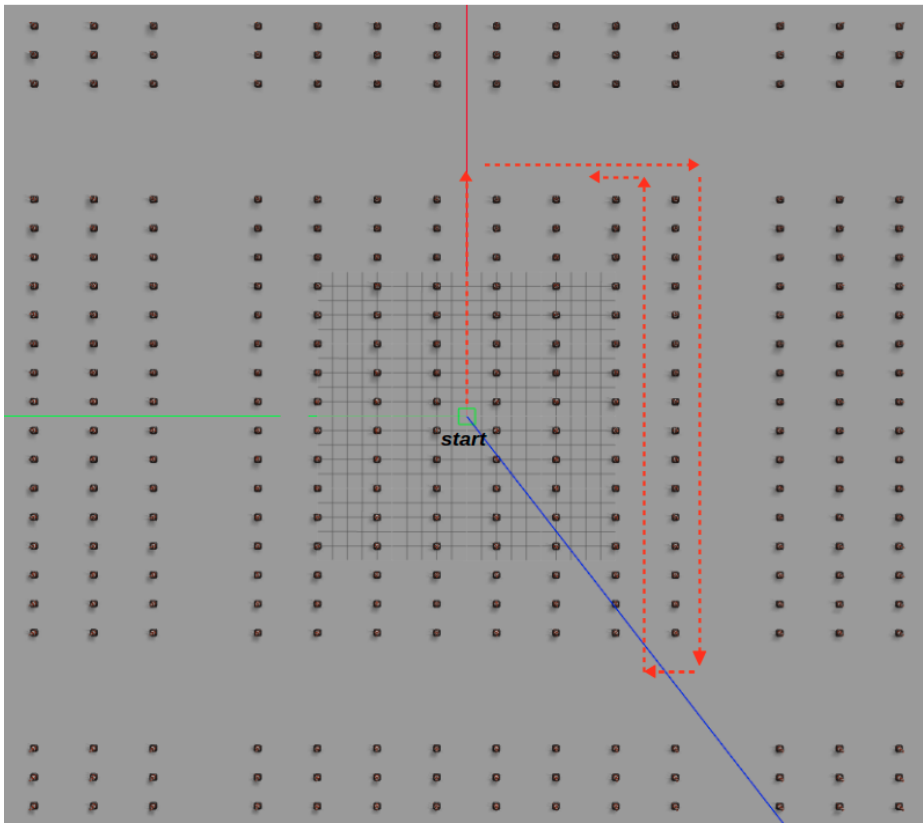
Project Structure and general info

- **What happens when the run_jackal script is invoked?**
- In short-Everything. The high level of automation was a key principle of the project since it enables to run many successive simulations easily and get the results rather quickly.
- Several terminals are opened one after the other- among them the Gazebo and the Rviz windows. The robot starts moving and the AMCL node is being launched.



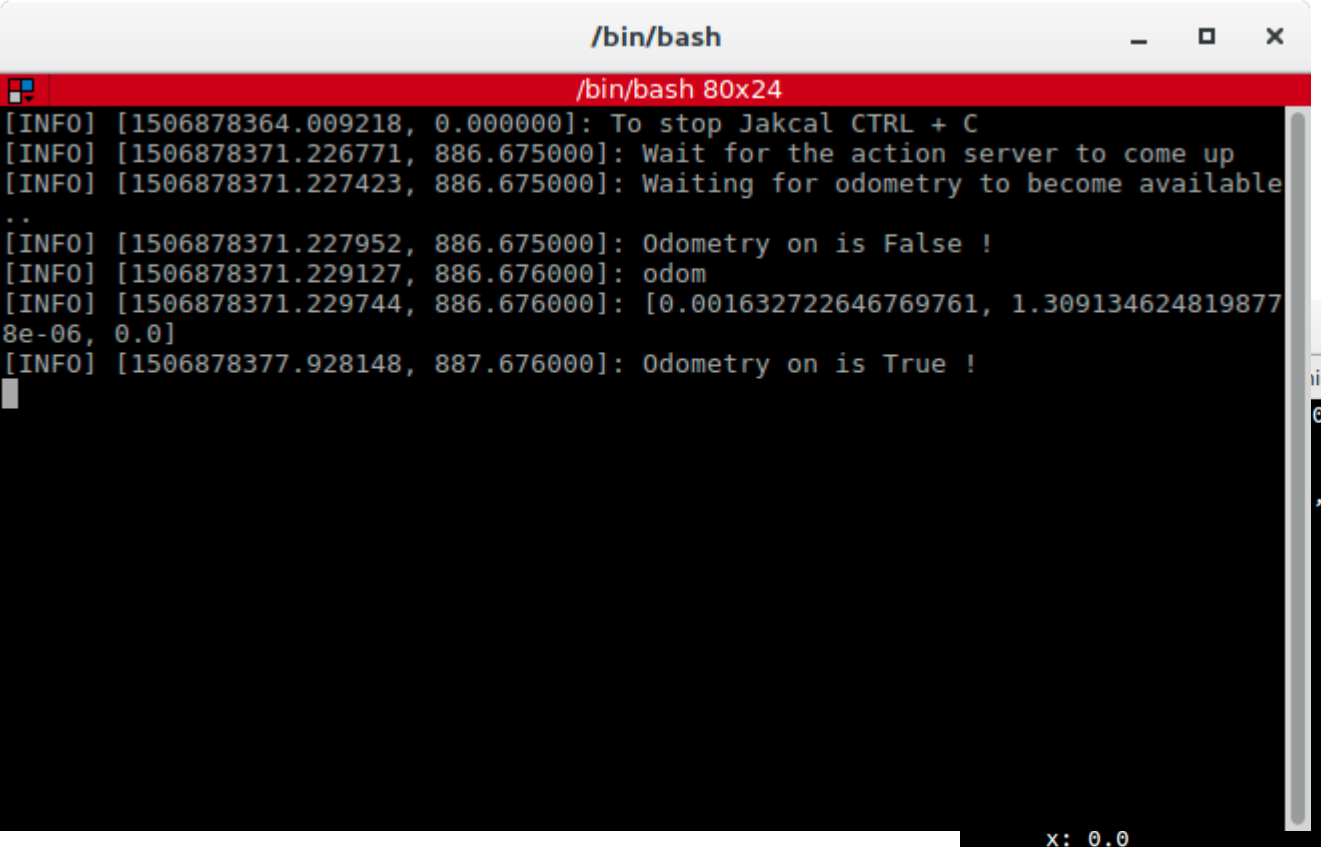
Project Structure and general info

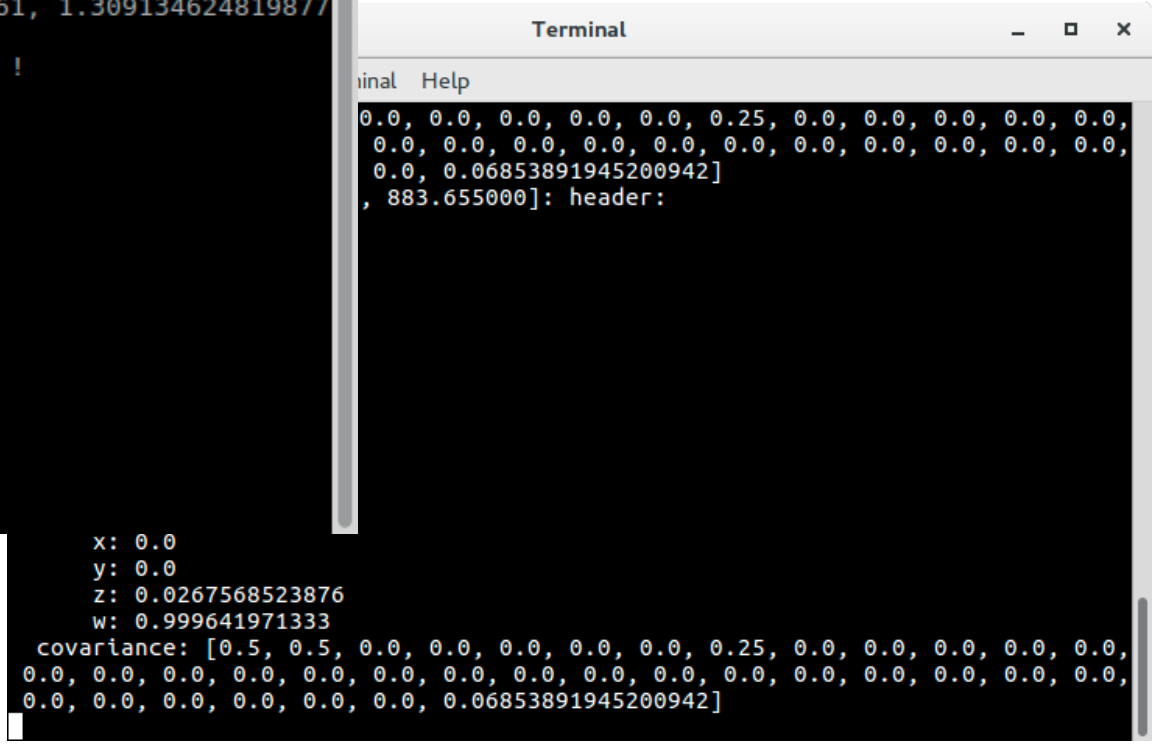
- **What happens when the run_jackal script is invoked (cont.)?**
- A script moves the robot in the path shown below- heading forward and rightwards, and than travel between the orchard's columns.
- Due to angular errors caused by odometry inaccuracy and timing Inaccuracies of the Gazebo simulator, the jackal always bumped into a cone. The actual path was not consistent, even when running successive simulations under the same conditions.
- *Even though- Understanding of the AMCL behavior and the ability of gathering insights about its performance remained intact.*
- The "heart" of the problem was the AMCL global localization, which Is invoked immediately, usually before the above errors takes a significant place.



Project Structure and general info

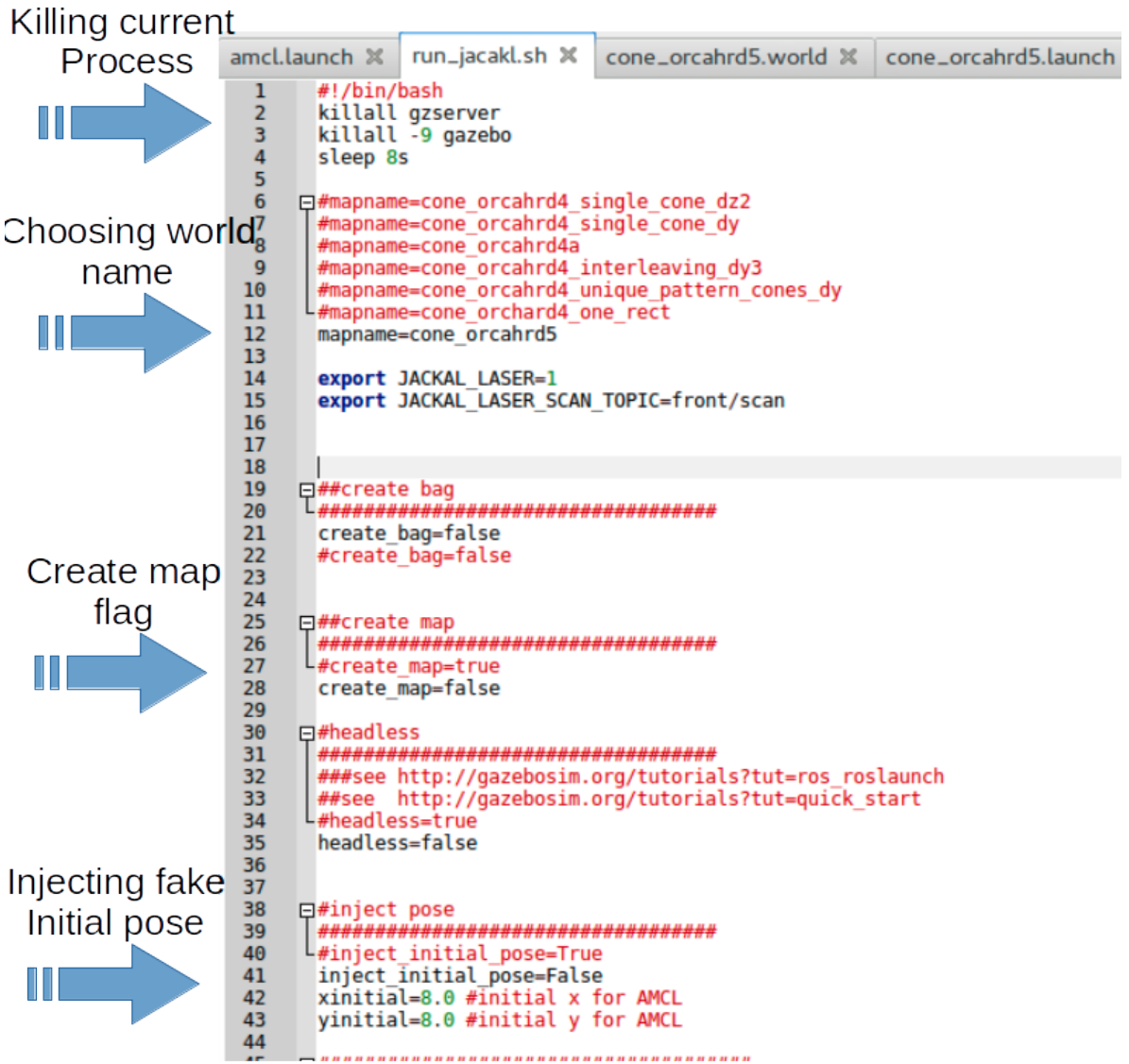
- **What happens when the run_jackal script is invoked (cont.)?**
- In order to stop the robot, one has to terminate (CTRL+C) the Terminator window seen below.
- In order to create maps (explained in the coming slides) , one has to terminate (CTRL+C) the terminal seen below.





Project Structure and info

- `run_jackal.sh` file structure:



Project Structure and info

- `run_jackal.sh` file structure (Cont.):

Launching the world

Creating a map

Launching amcl/Rviz

Running Scripts

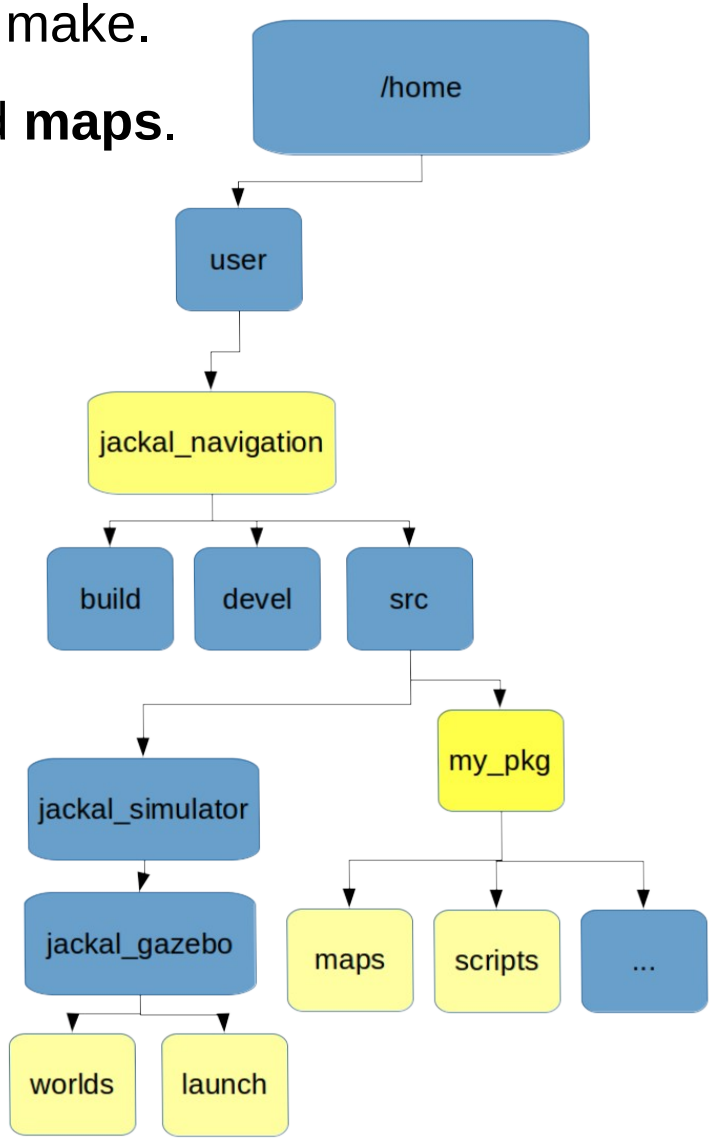
amcl.launch | run_jacakl.sh | cone_orcahrd5.world | cone_orcahrd5.launch | cone_orcahrd5.yaml | cone_orcahrd4.world | cone_

```
70  else
71  #gnome-terminal -e "roslaunch gazebo_ros empty_world.launch"&
72
73  gnome-terminal -e "roslaunch jackal_gazebo $mapname.launch gui:=true headless:=false"&
74  sleep 30s
75  #gnome-terminal -e "roslaunch jackal_gazebo jackal.launch gui:=true headless:=false"&
76  #gnome-terminal  -e "roslaunch jackal_navigation odom_navigation_demo.launch"&
77  sleep 5s
78
79  if $create_map; then
80      sleep 60s
81      sh mybash.sh $mapname
82  fi
83
84  gnome-terminal  -e "rosrun map_server map_server /home/liron/jackal_navigation/src/my_pkg/maps/$mapname/$mapname
85  sleep 10s
86  gnome-terminal  -e "roslaunch jackal_navigation amcl_demo.launch map_file:=/home/liron/jackal_navigation/src/my_
87  sleep 6s
88  gnome-terminal  -e "roslaunch jackal_viz view_robot.launch config:=localization"&
89
90  fi
91
92  ##########################################################################
93
94
95
96  ##ruunning scripts
97  ####################################
98
99  cd /home/liron/jackal_navigation/src/my_pkg/maps/$mapname
100
101
102 sleep 5s
103
104 gnome-terminal  -e "rosrun my_pkg doitall_node2.py $xinitial $yinitial $inject_initial_pose"&
105
106 sleep 6s
107
108
109
110 if $create_bag; then
111 gnome-terminal -e "rosrun teleop_twist_keyboard teleop_twist_keyboard.py"&
112 gnome-terminal -e "roslaunch jackal_gazebo bag_record.launch"&
```

Project Structure and info

- Folder Structure is shown on the diagram on the right.
- The worlds and launch files are exactly where they are on the navigation stack.
- One added package called **my_pkg** using Catkin make.
- The package contains mainly **python scripts** and **maps**.
- The maps folder contains maps and graphs.

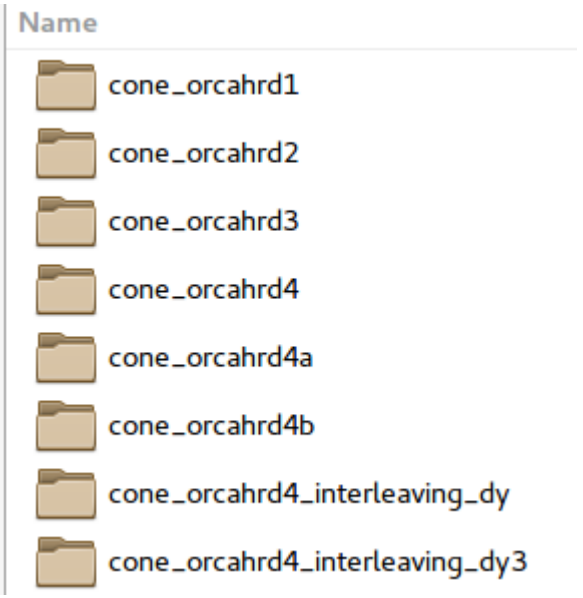
  Graphs are created automatically after every simulation is terminated using ctrl+c.
- Maps are created if the flag “create_map” on run_jackal script is set to *true*.

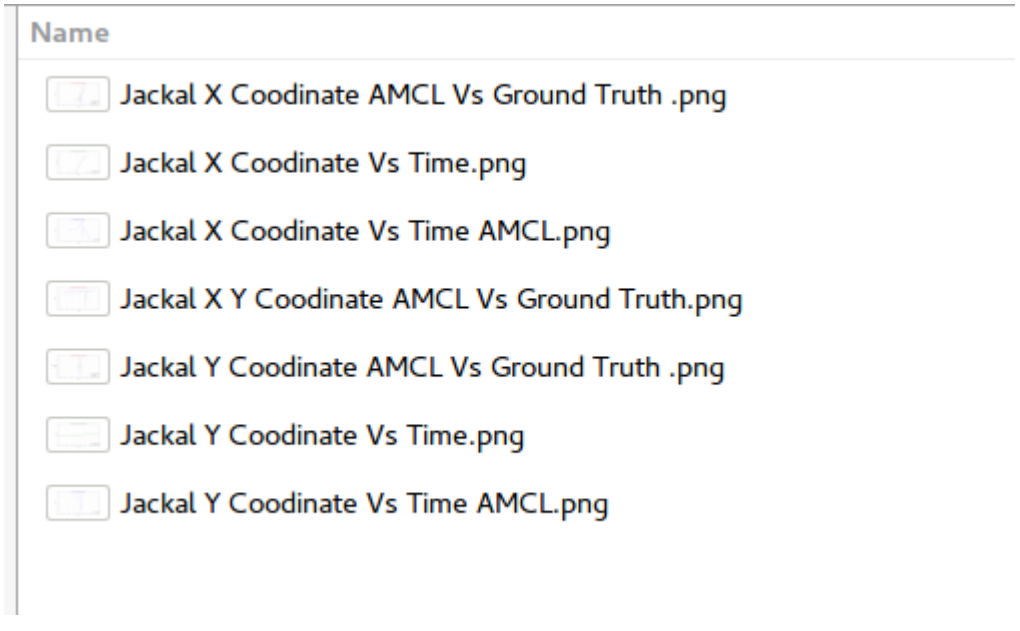


Project Structure and info

What's inside maps dir?

- Each folder carries the name of the Gazebo world on which the simulation is being running on.
- When running the simulation,several terminals are opened: for gazebo, Rviz, AMCL and some other nodes.
- When terminating the AMCL terminal (Ctrl+C) - graphs are created automatically and saved inside the appropriate folder.

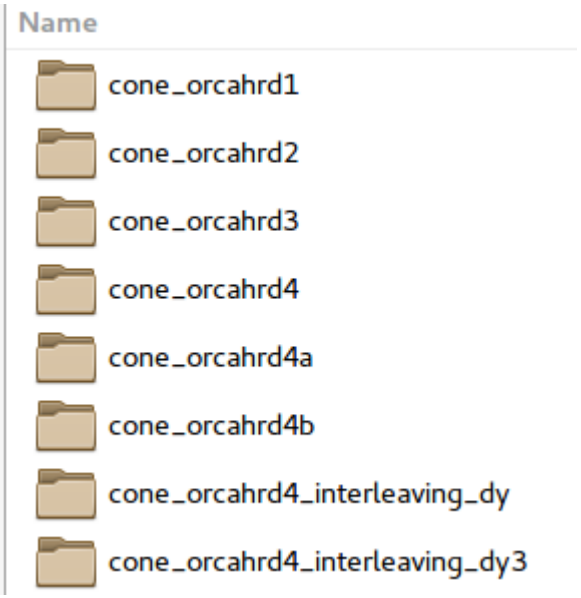


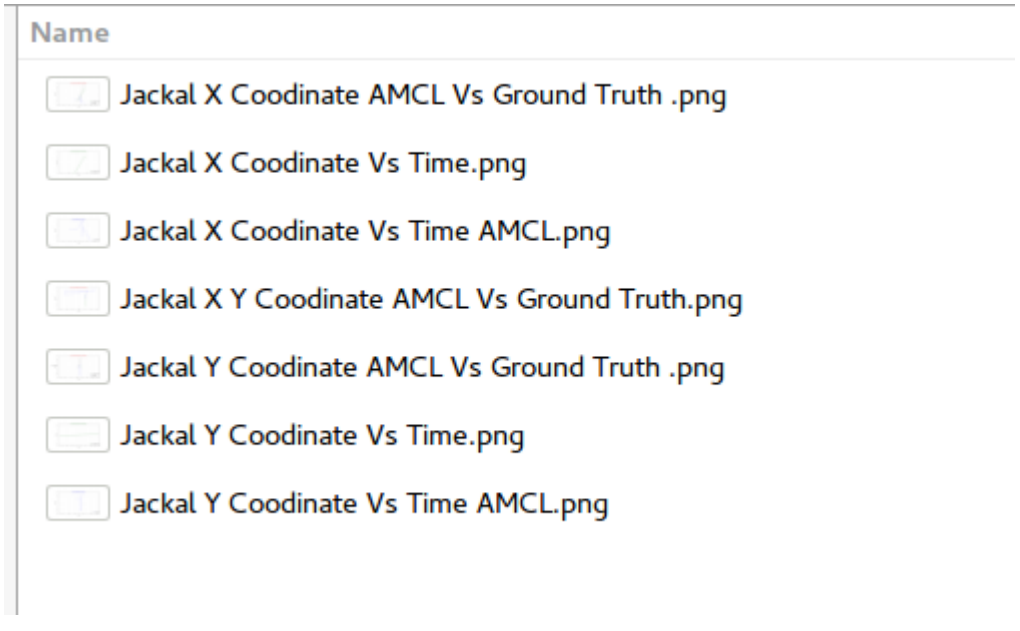


Project Structure and info

What's inside maps dir (cont.)?

- The graphs created are:
  - X vs time- ground truth
  - Y vs time- ground truth
  - X vs time- ground AMCL
  - Y vs time- ground AMCL
  - X vs time- ground truth & AMCL
  - Y vs time- ground truth & AMCL
  - X Y ground truth & AMCL (trajectory - no time)

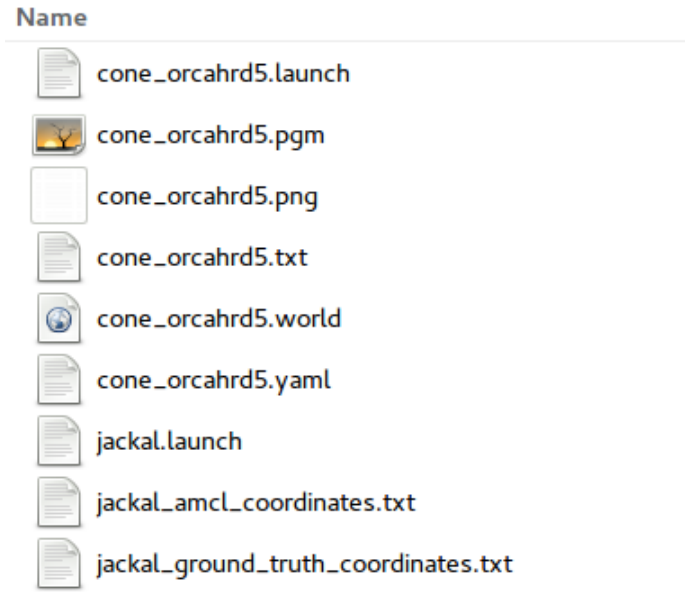


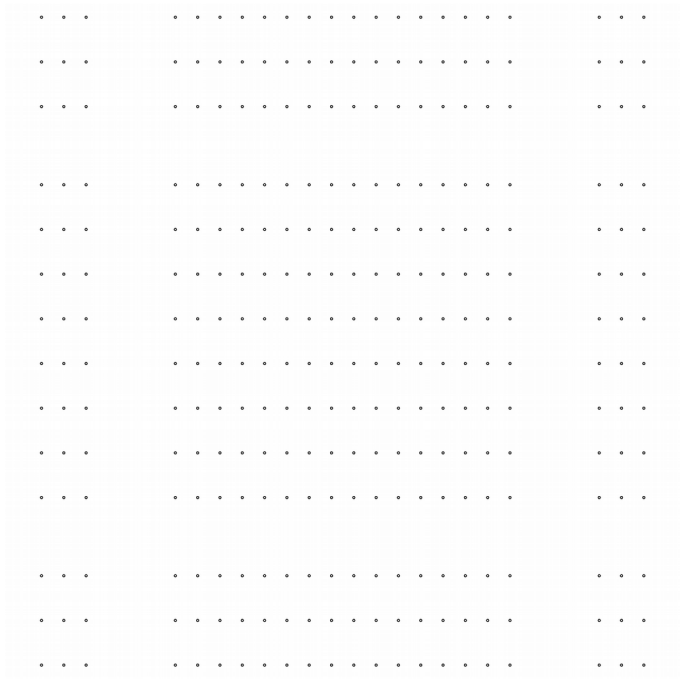


Project Structure and info

What's inside maps dir?

- There are copies of the world and launch files from the jackal_gazebo directory.
- Two images .pgm & .png which are the ground truth map. Both can be used.
- Text files of the ground truth/AMCL messages after parsing.
- worldname.txt – used for creating the map.
- .yaml file- used by Rviz for reading the map properly.



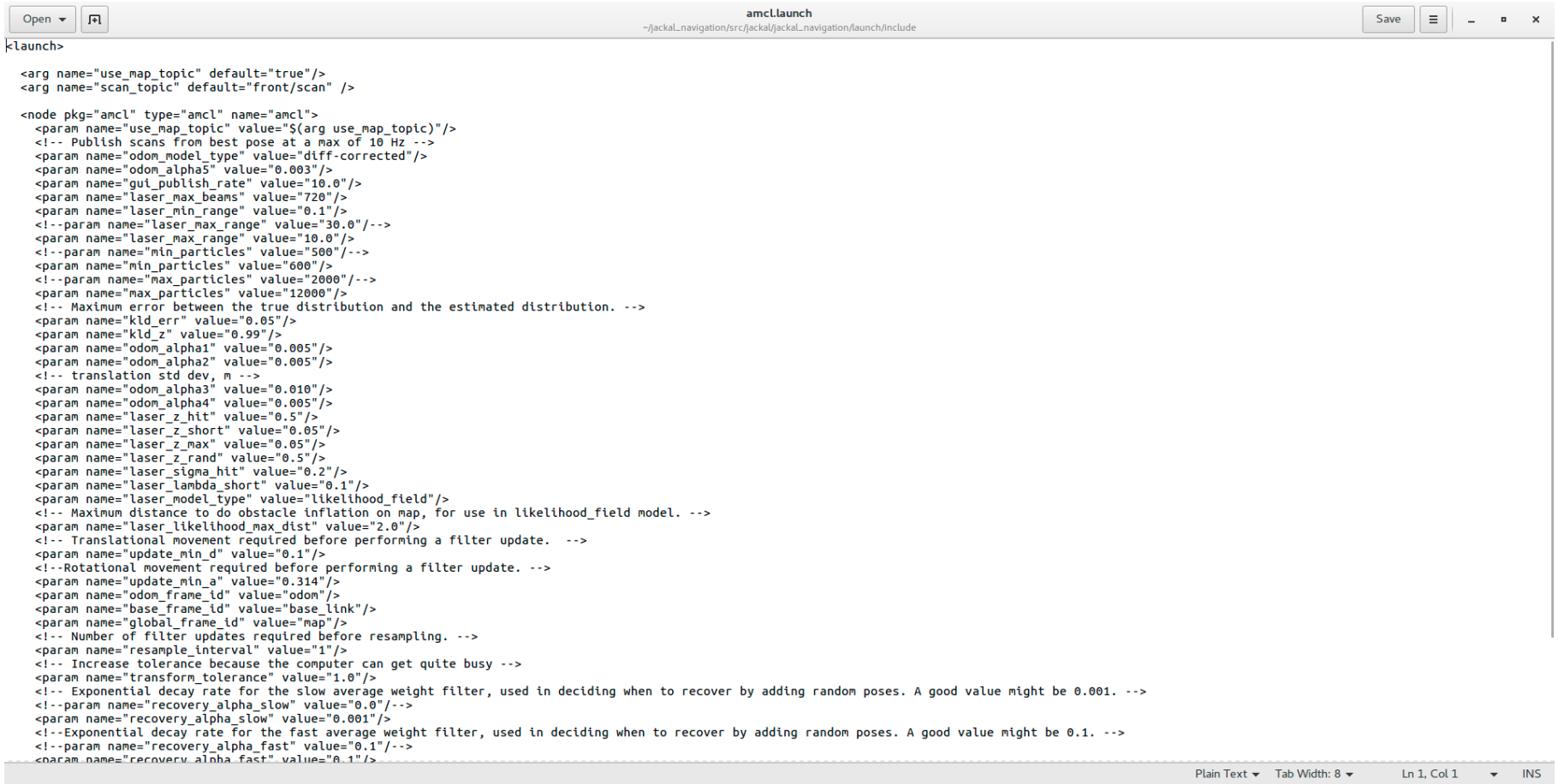


map

Project Structure and info

Amcl.launch file

- The amcl.launch file in the navigation stack contains all the parameters needed for the AMCL node.
- ***Important! -*** *The parameters were adjusted carefully . Please read http://wiki.ros.org/amcl , and especially https://answers.ros.org/question/227811/tuning-amcls-diff-corrected-and-omni-corrected-odom-models/*




```
<launch>

  <arg name="use_map_topic" default="true"/>
  <arg name="scan_topic" default="front/scan" />

  <node pkg="amcl" type="amcl" name="amcl">
    <param name="use_map_topic" value="$(arg use_map_topic)"/>
    <!-- Publish scans from best pose at a max of 10 Hz -->
    <param name="odom_model_type" value="diff-corrected"/>
    <param name="odom_alpha5" value="0.003"/>
    <param name="gui_publish_rate" value="10.0"/>
    <param name="laser_max_beams" value="720"/>
    <param name="laser_min_range" value="0.1"/>
    <!--param name="laser_max_range" value="30.0"/-->
    <param name="laser_max_range" value="10.0"/>
    <!--param name="min_particles" value="500"/-->
    <param name="min_particles" value="600"/>
    <!--param name="max_particles" value="2000"/-->
    <param name="max_particles" value="12000"/>
    <!-- Maximum error between the true distribution and the estimated distribution. -->
    <param name="kld_err" value="0.05"/>
    <param name="kld_z" value="0.99"/>
    <param name="odom_alpha1" value="0.005"/>
    <param name="odom_alpha2" value="0.005"/>
    <!-- translation std dev, m -->
    <param name="odom_alpha3" value="0.010"/>
    <param name="odom_alpha4" value="0.005"/>
    <param name="laser_z_hit" value="0.5"/>
    <param name="laser_z_short" value="0.05"/>
    <param name="laser_z_max" value="0.05"/>
    <param name="laser_z_rand" value="0.5"/>
    <param name="laser_sigma_hit" value="0.2"/>
    <param name="laser_lambda_short" value="0.1"/>
    <param name="laser_model_type" value="likelihood_field"/>
    <!-- Maximum distance to do obstacle inflation on map, for use in likelihood_field model. -->
    <param name="laser_likelihood_max_dist" value="2.0"/>
    <!-- Translational movement required before performing a filter update.  -->
    <param name="update_min_d" value="0.1"/>
    <!--Rotational movement required before performing a filter update. -->
    <param name="update_min_a" value="0.314"/>
    <param name="odom_frame_id" value="odom"/>
    <param name="base_frame_id" value="base_link"/>
    <param name="global_frame_id" value="map"/>
    <!-- Number of filter updates required before resampling. -->
    <param name="resample_interval" value="1"/>
    <!-- Increase tolerance because the computer can get quite busy -->
    <param name="transform_tolerance" value="1.0"/>
    <!-- Exponential decay rate for the slow average weight filter, used in deciding when to recover by adding random poses. A good value might be 0.001. -->
    <!--param name="recovery_alpha_slow" value="0.0"/-->
    <param name="recovery_alpha_slow" value="0.001"/>
    <!--Exponential decay rate for the fast average weight filter, used in deciding when to recover by adding random poses. A good value might be 0.1. -->
    <!--param name="recovery_alpha_fast" value="0.1"/-->
    <param name="recovery_alpha_fast" value="0.1"/>
```

Project Structure and info

Project location

- *Github : https://github.com/sliri/my_pkg.git*
- One has to first install the jackal_navigation stack and then add my_pkg package using Catkin_make
- What appears on Github is only
  - my_pkg directory (with the scripts and maps)
  - package.xml & Cmakelists.txt that should be added to my_pkg folder
  - worlds & launch files directories (their content should be copied to the same folders inside the navigation stack)
  - Presentation folder, where this presentation is being stored.
  - sh_files folder- for bash scripts. These should be copied to the user's home directory.
  - Videos
  - Cylinder models