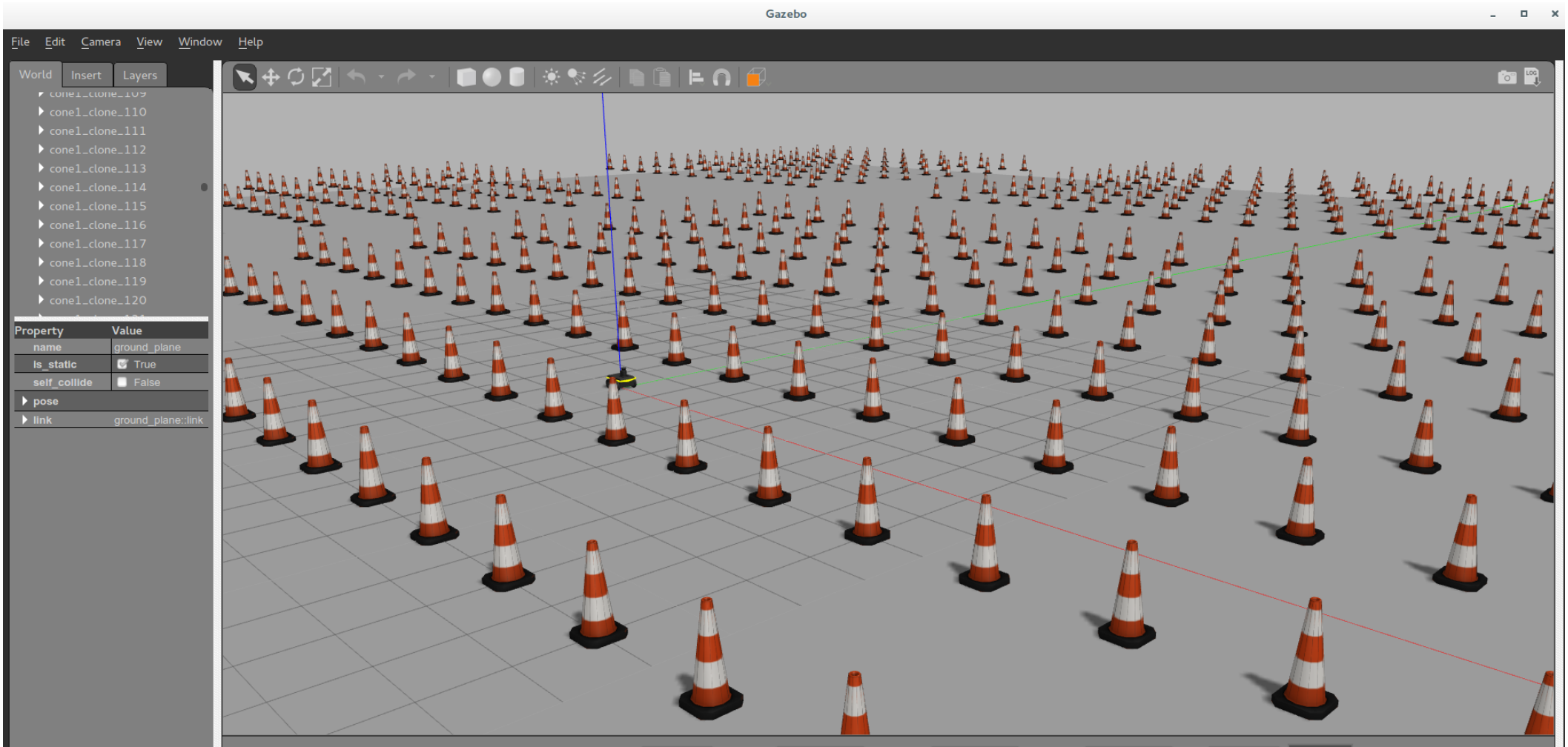
The effect of orchard heterogeneity on navigating an autonomous vehicle for agricultural tasks

Submits: Liron Shenkar

Supervisors Prof. Amir Degani , Omer Shalev



Project Structure and general info

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

- The AMCL 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.

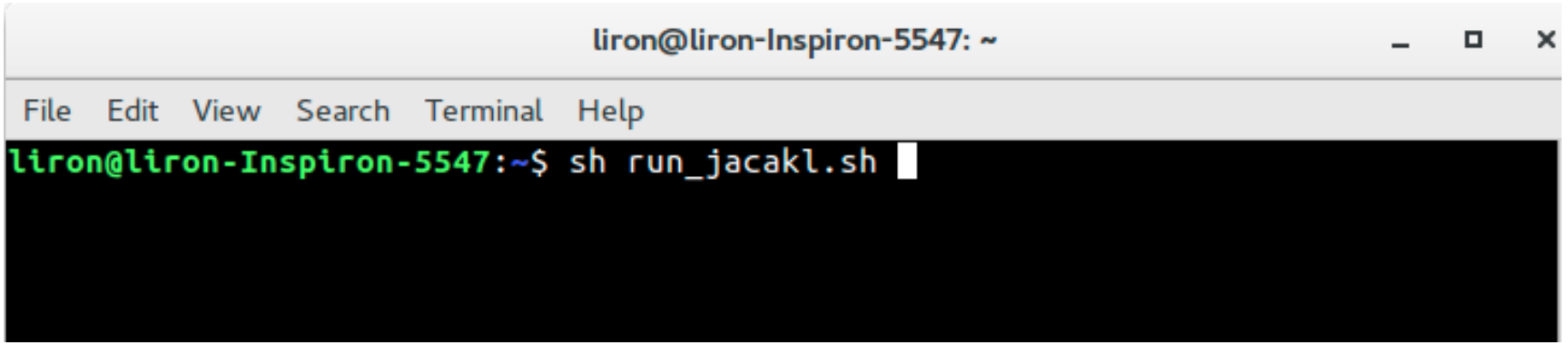
- ***The higher the homogeneity of the area-the harder it is to deliver good localization results.***

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

Project Structure and general info

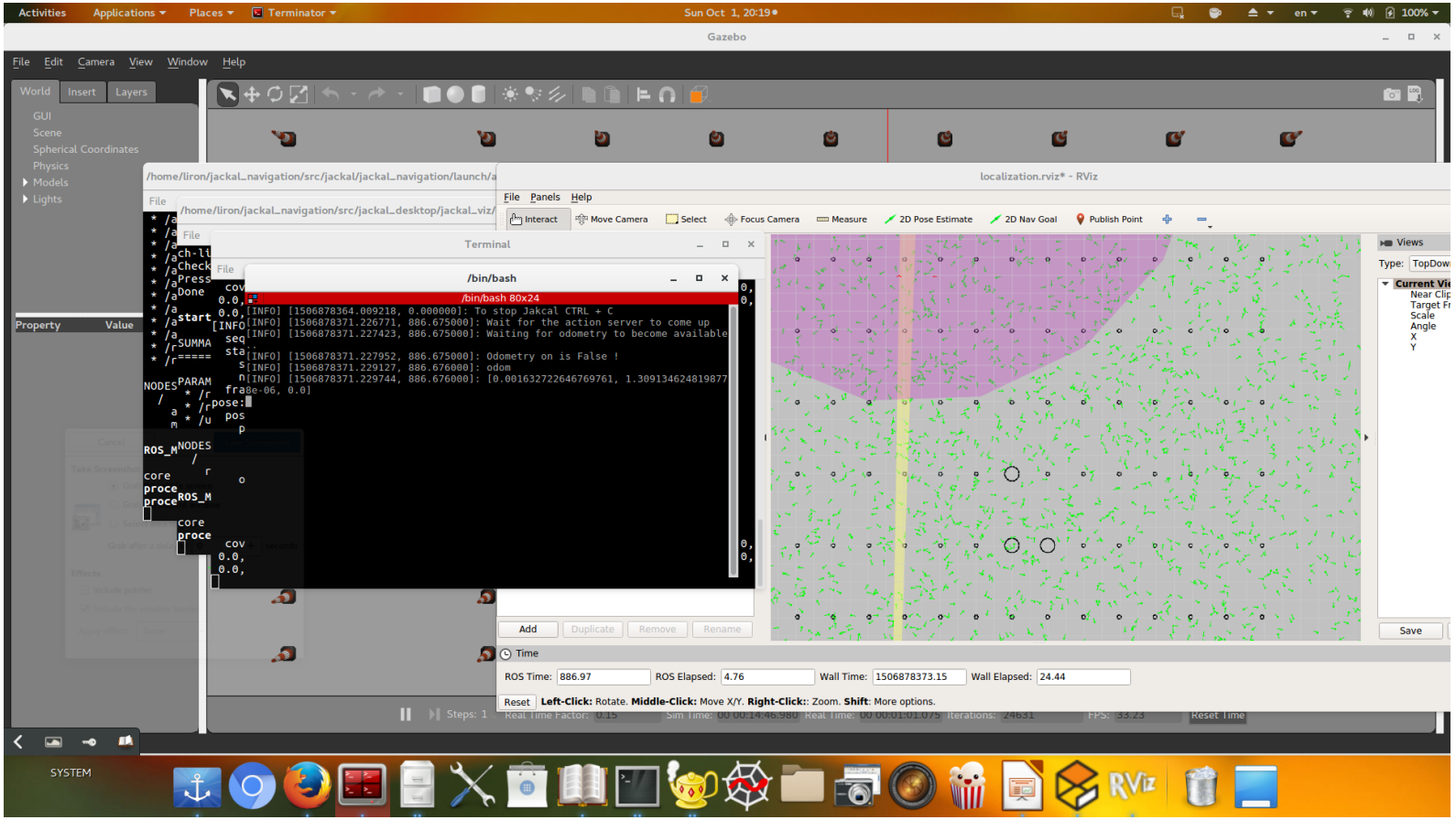
- All project files are in jackal_navigation dir, inside linux user's home directory.
- Directory contents in based on jackal navigation stack from CLEARPATH® website

  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:



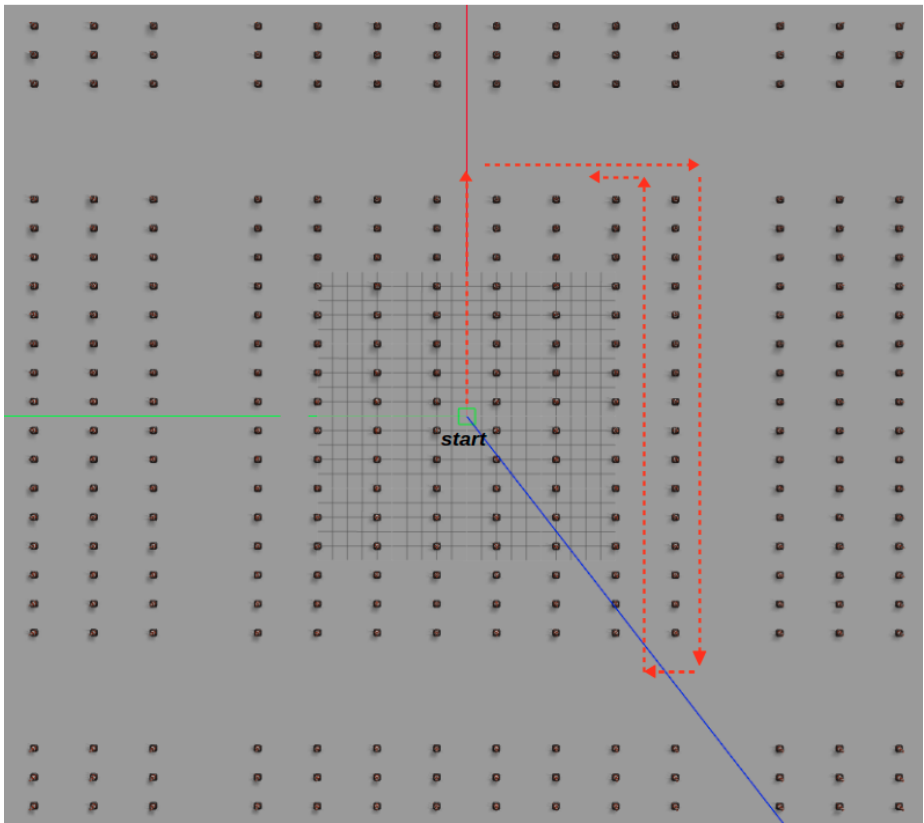
Project Structure and general info

- **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.



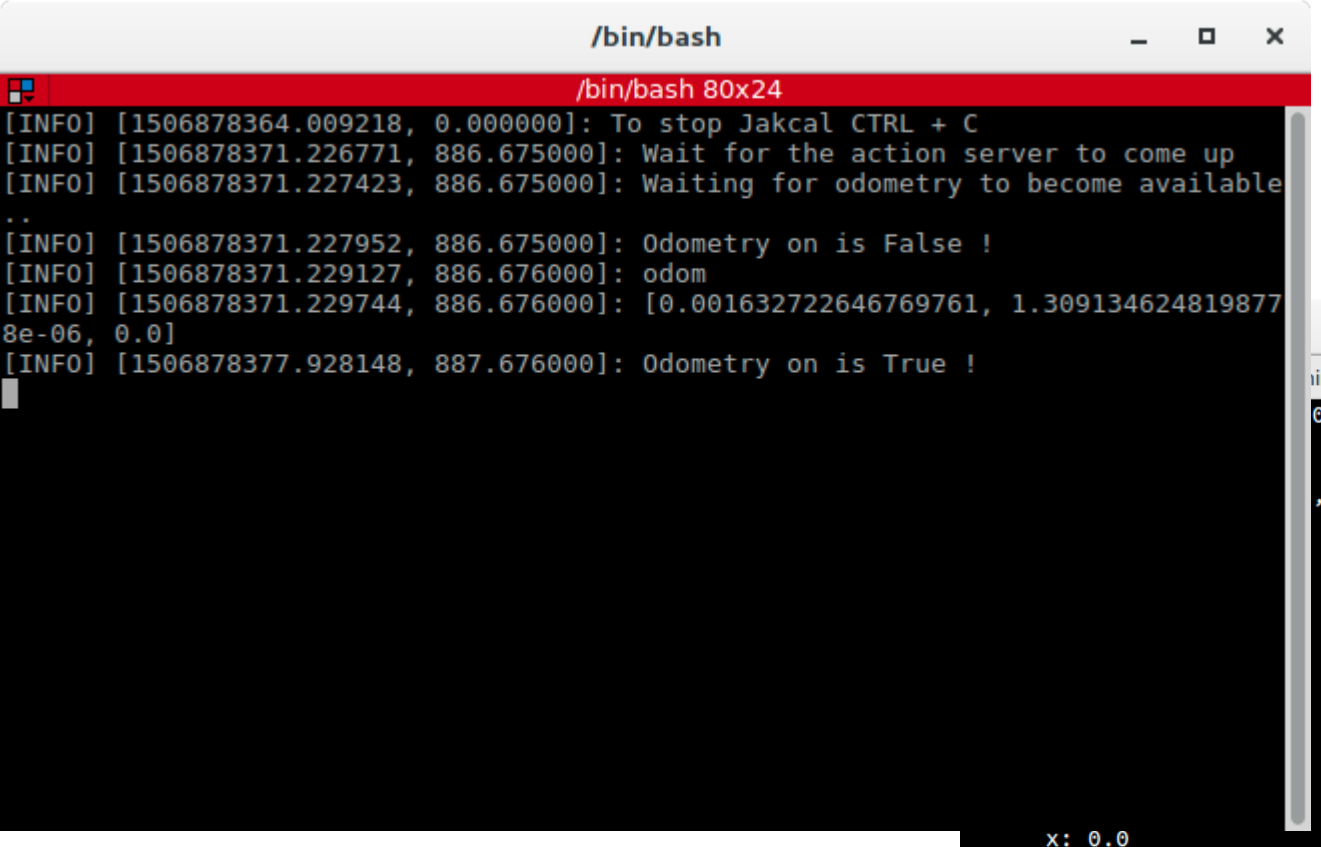
Project Structure and general info

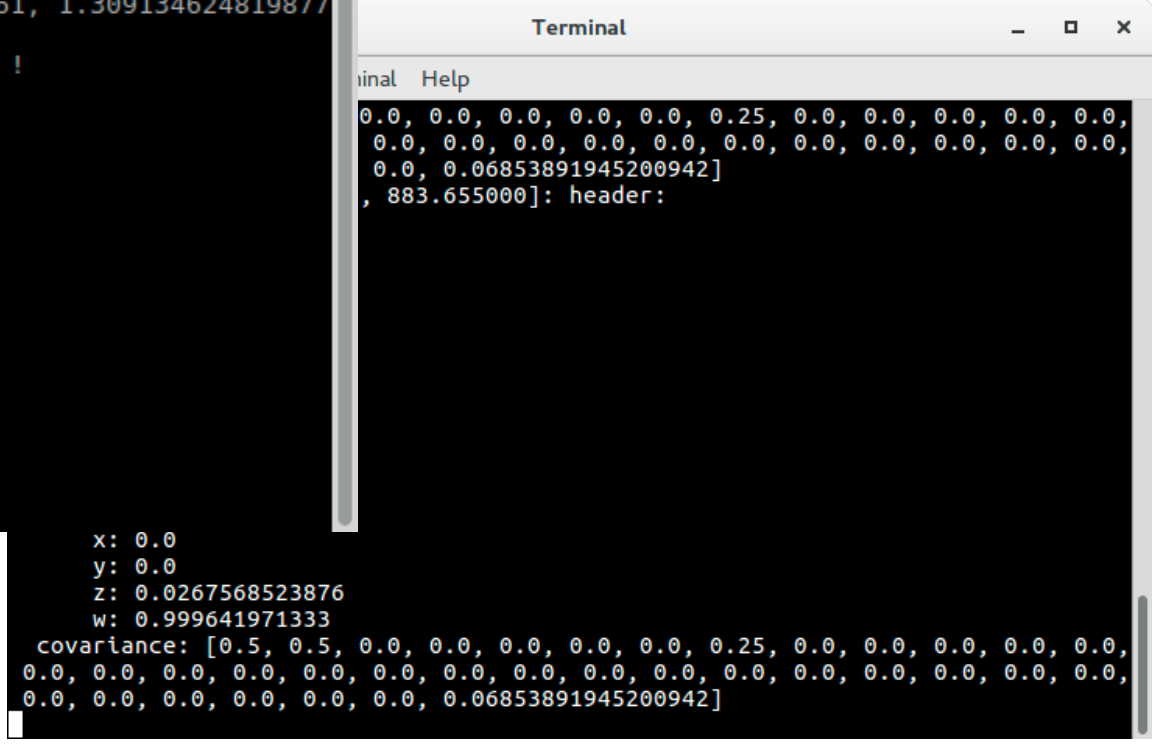
- **What happens when the run_jackal script is invoked (cont.)?**
- A script moves the robot in the path shown below- heading forward and rightwards, and than travel between the orchard's columns.
- Due to angular errors caused by odometry inaccuracy and timing Inaccuracies of the Gazebo simulator, the 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

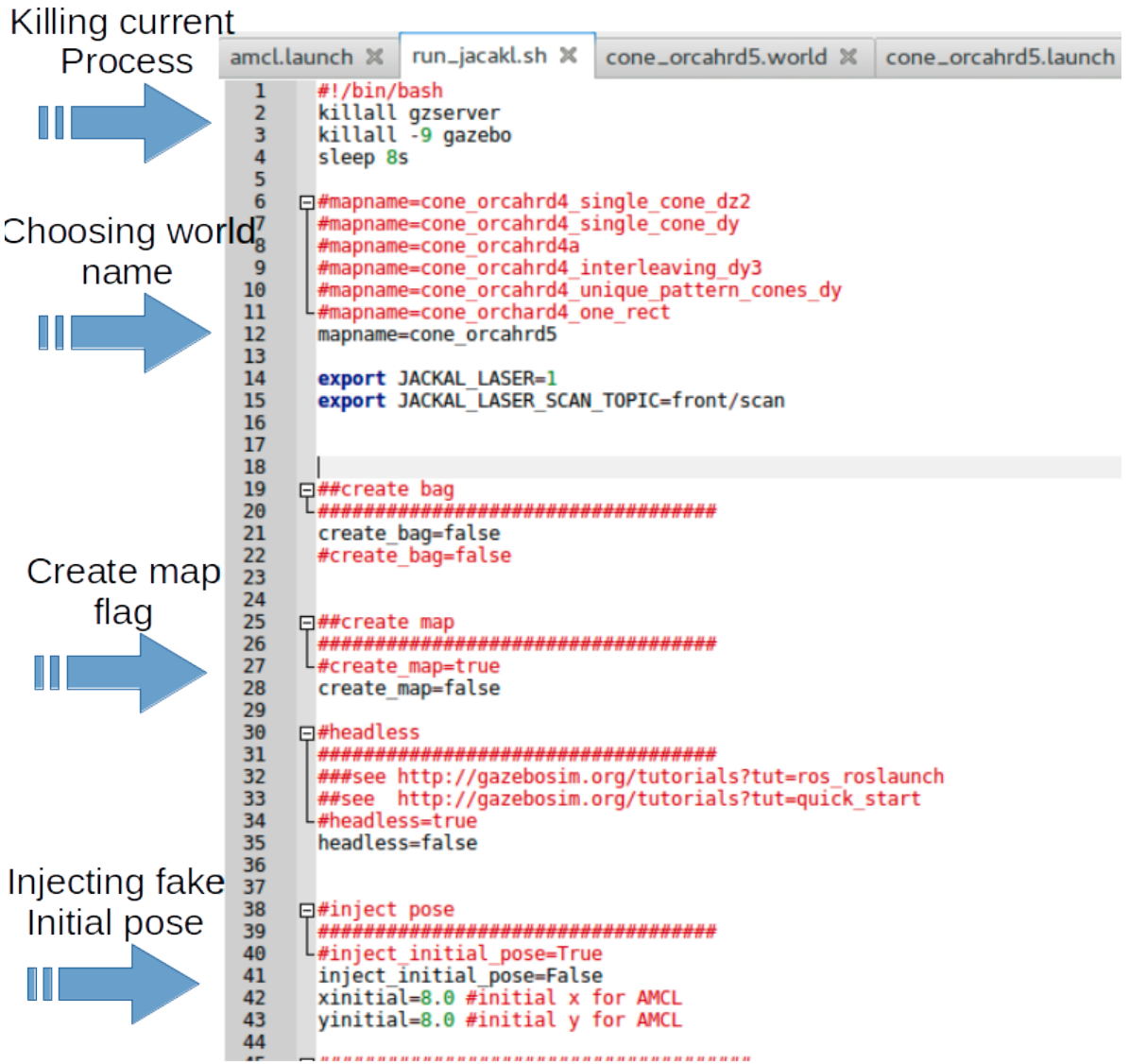
- **What happens when the run_jackal script is invoked (cont.)?**
- In order to stop the robot, one has to terminate (CTRL+C) the Terminator window seen below.
- In order to create maps (explained in the coming slides) , one has to terminate (CTRL+C) the terminal seen below.





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

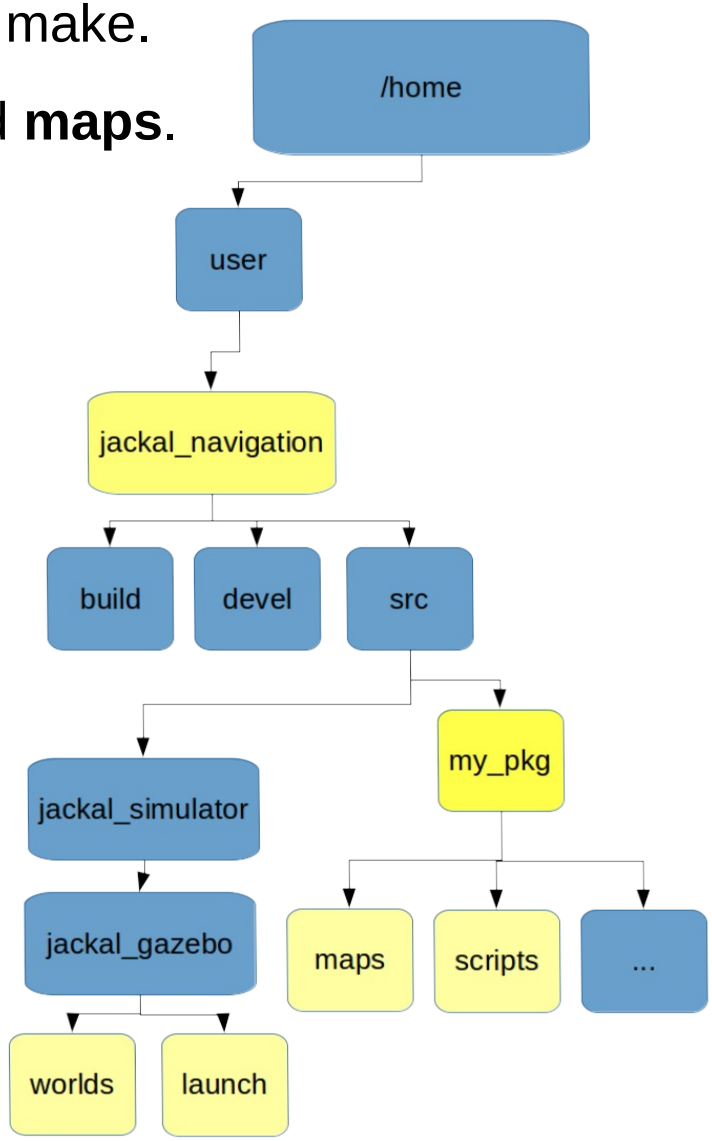
amcl.launch | run_jacakl.sh | cone_orcahrd5.world | cone_orcahrd5.launch | cone_orcahrd5.yaml | cone_orcahrd4.world | cone_

```
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"&
```

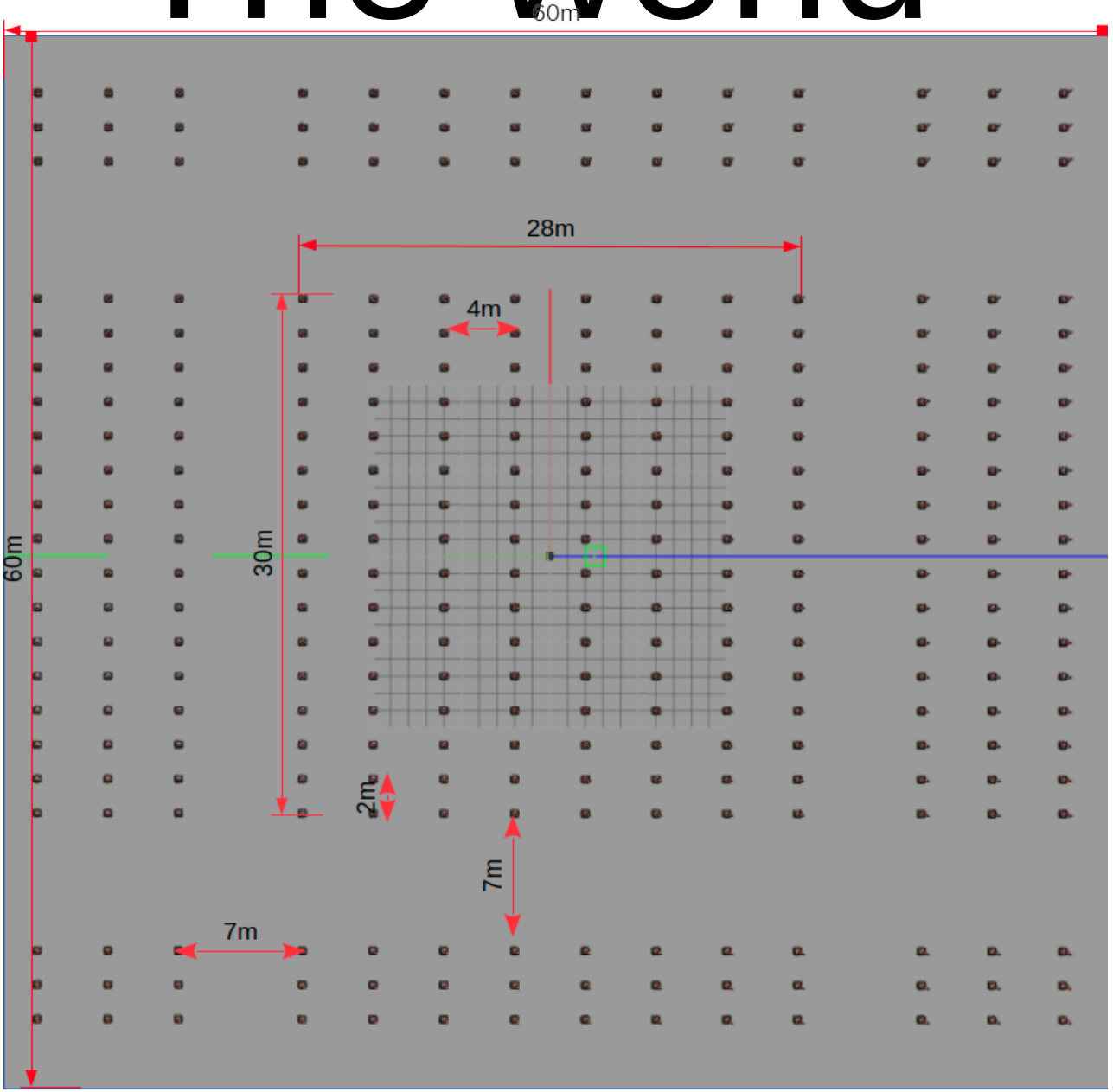
Project Structure and info

- 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*.



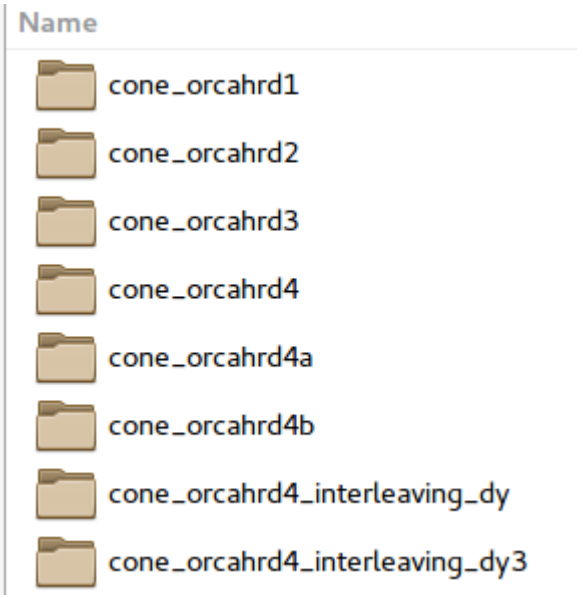
The world

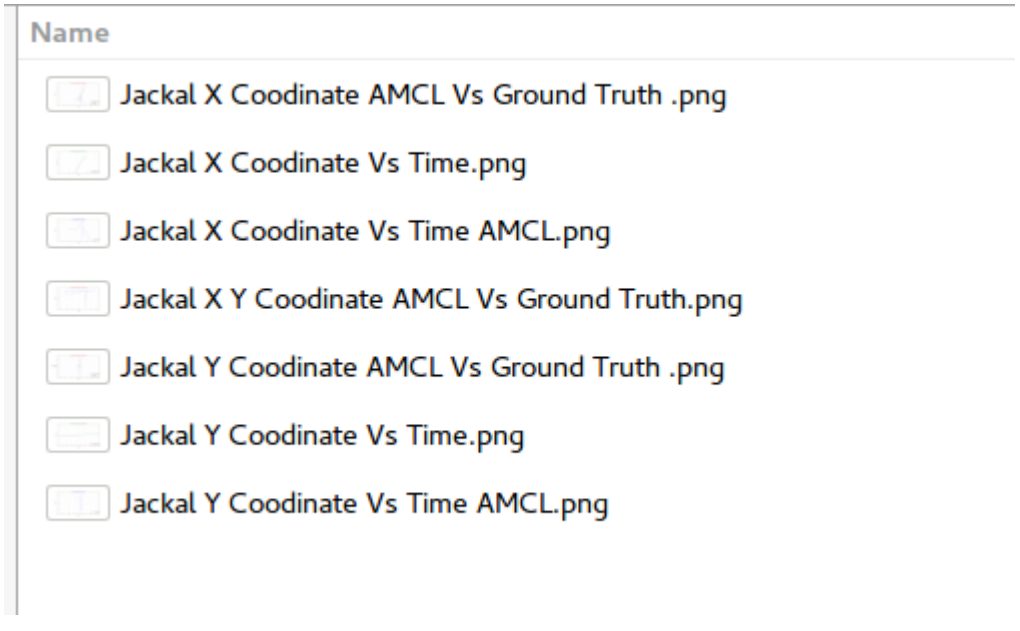


Project Structure and info

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.

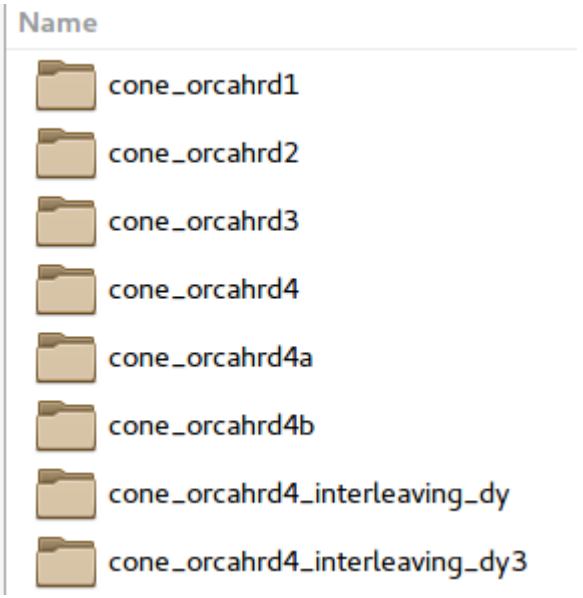


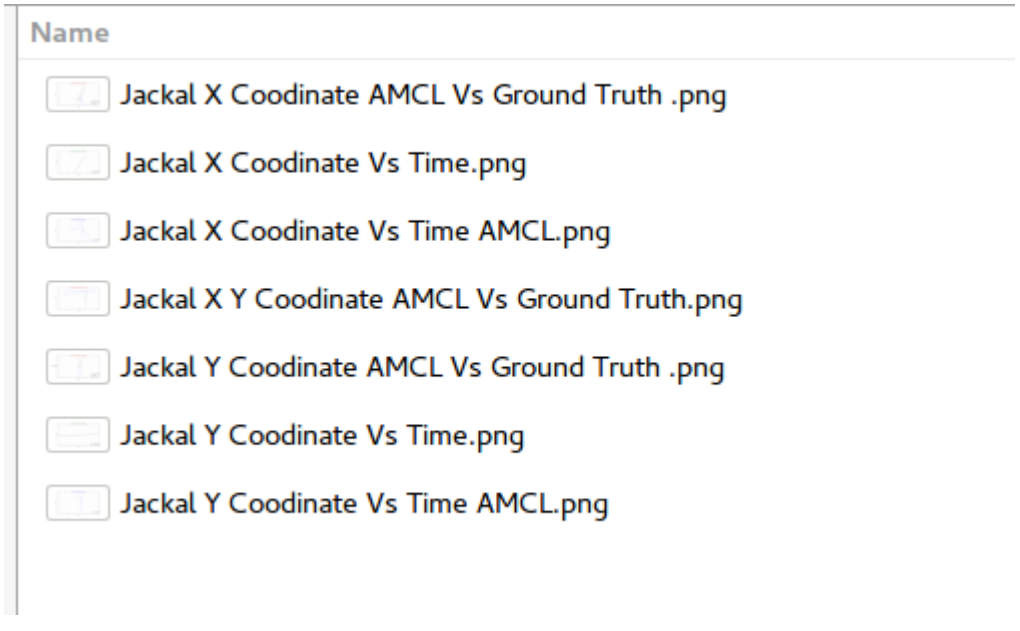


Project Structure and info

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)

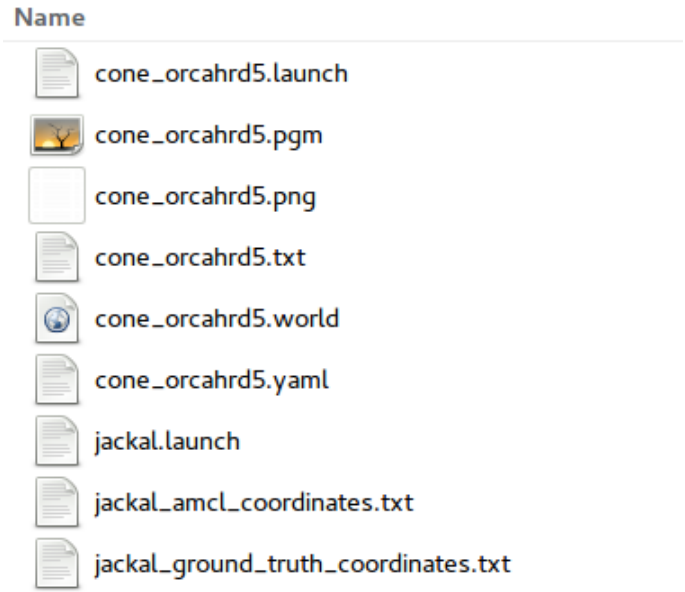


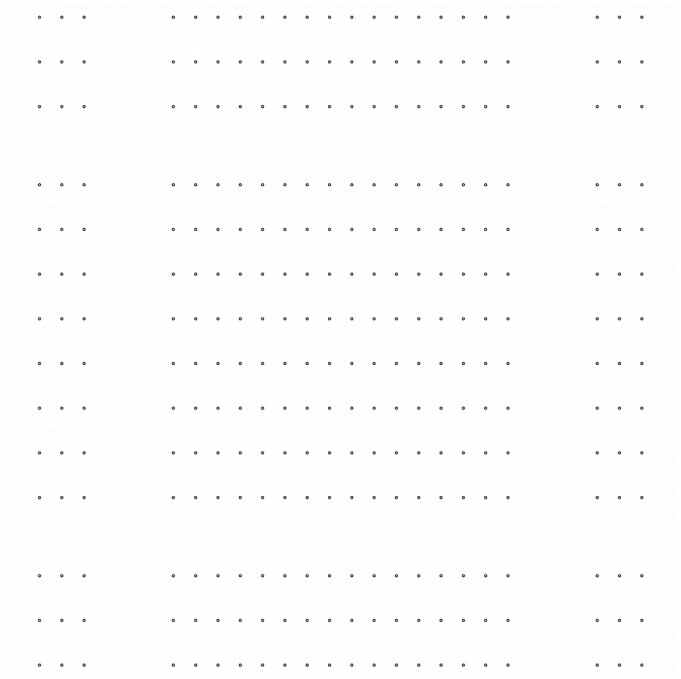


Project Structure and info

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.





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/*

Open ▾ | amcl.launch | ~/jackal_navigation/src/jackal/jackal_navigation/launch/include | Save

```
<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"/>
```

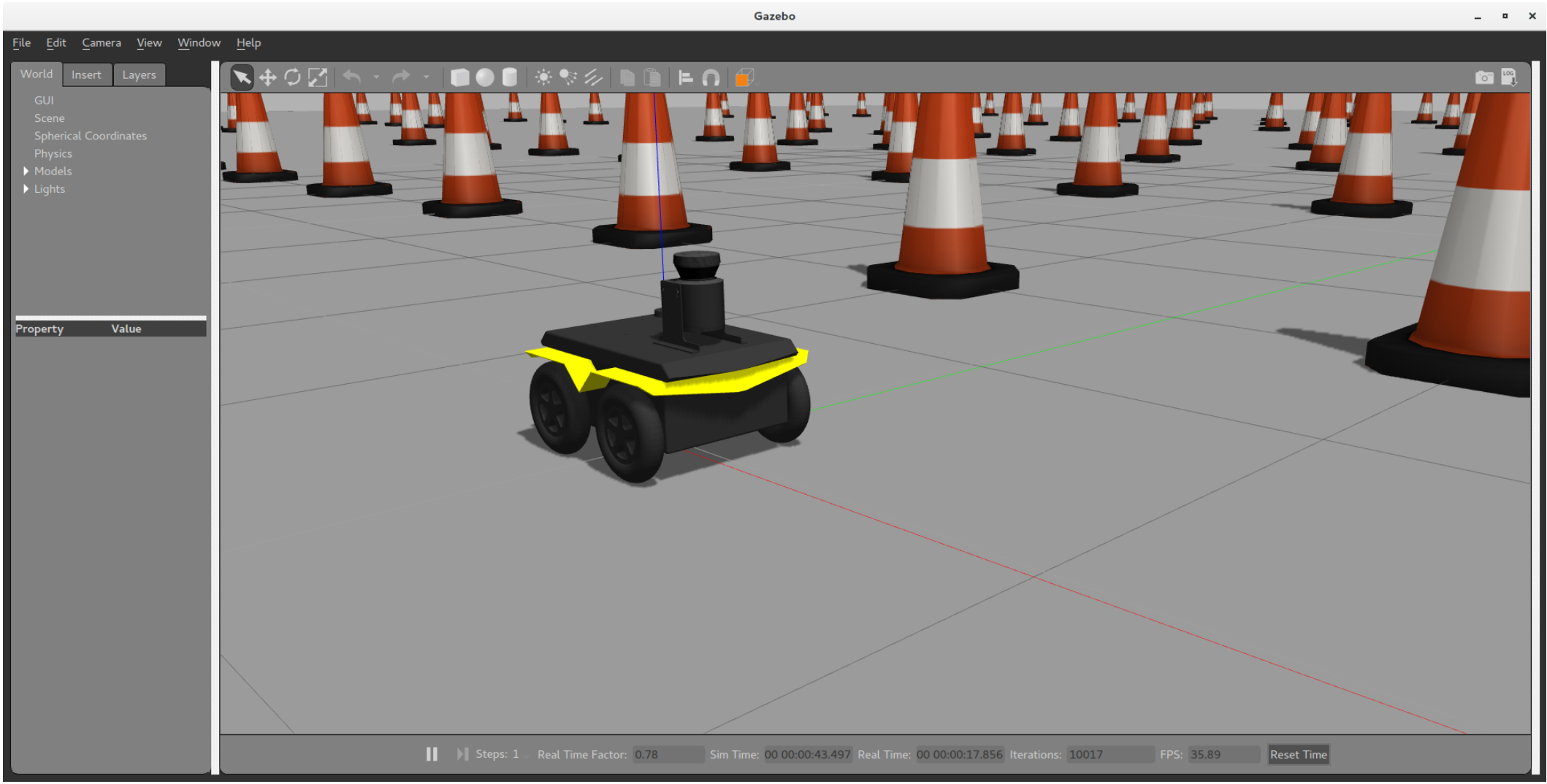
Plain Text ▾ | Tab Width: 8 ▾ | Ln 1, Col 1 | INS

Project Structure and info

Project location

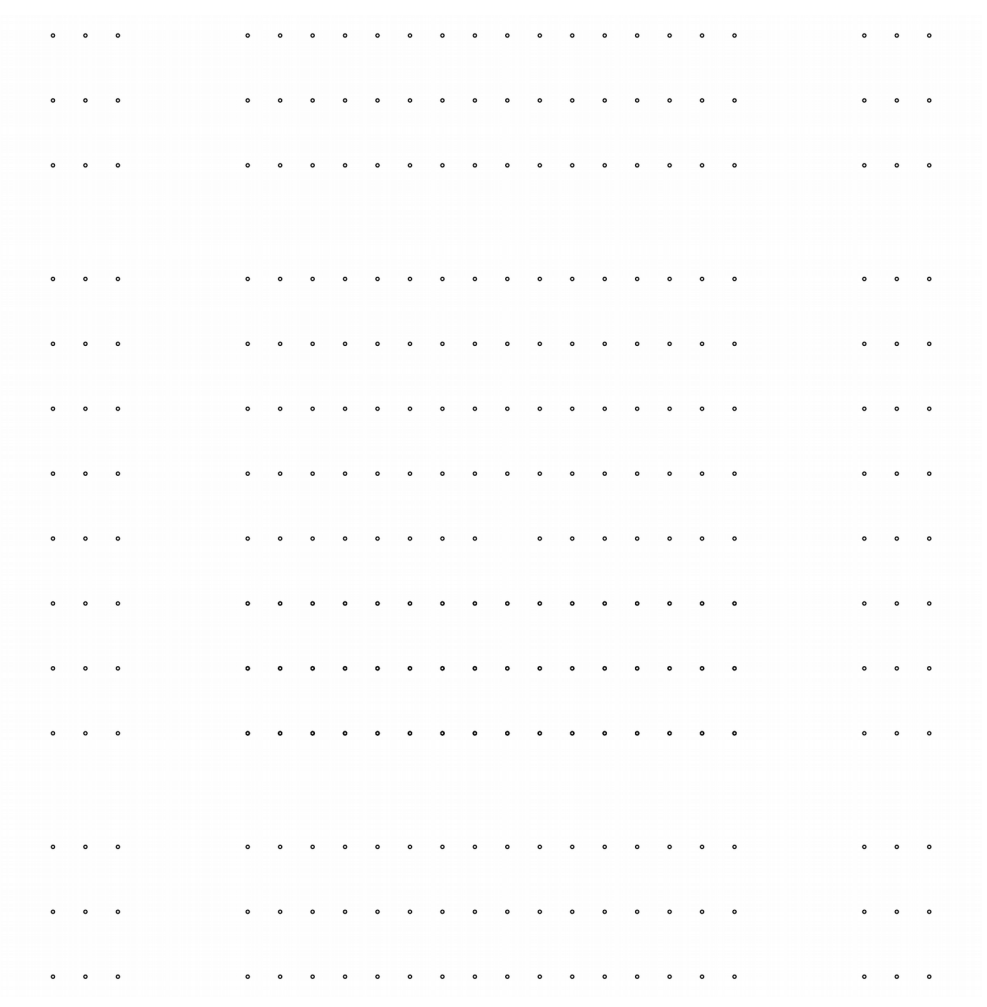
- *Github : https://github.com/sliri/my_pkg.git*
- One has to first install the jackal_navigation stack and then add my_pkg package using Catkin_make
- What appears on Github is only
  - my_pkg directory (with the scripts and maps)
  - package.xml & Cmakelists.txt that should be added to my_pkg folder
  - 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.

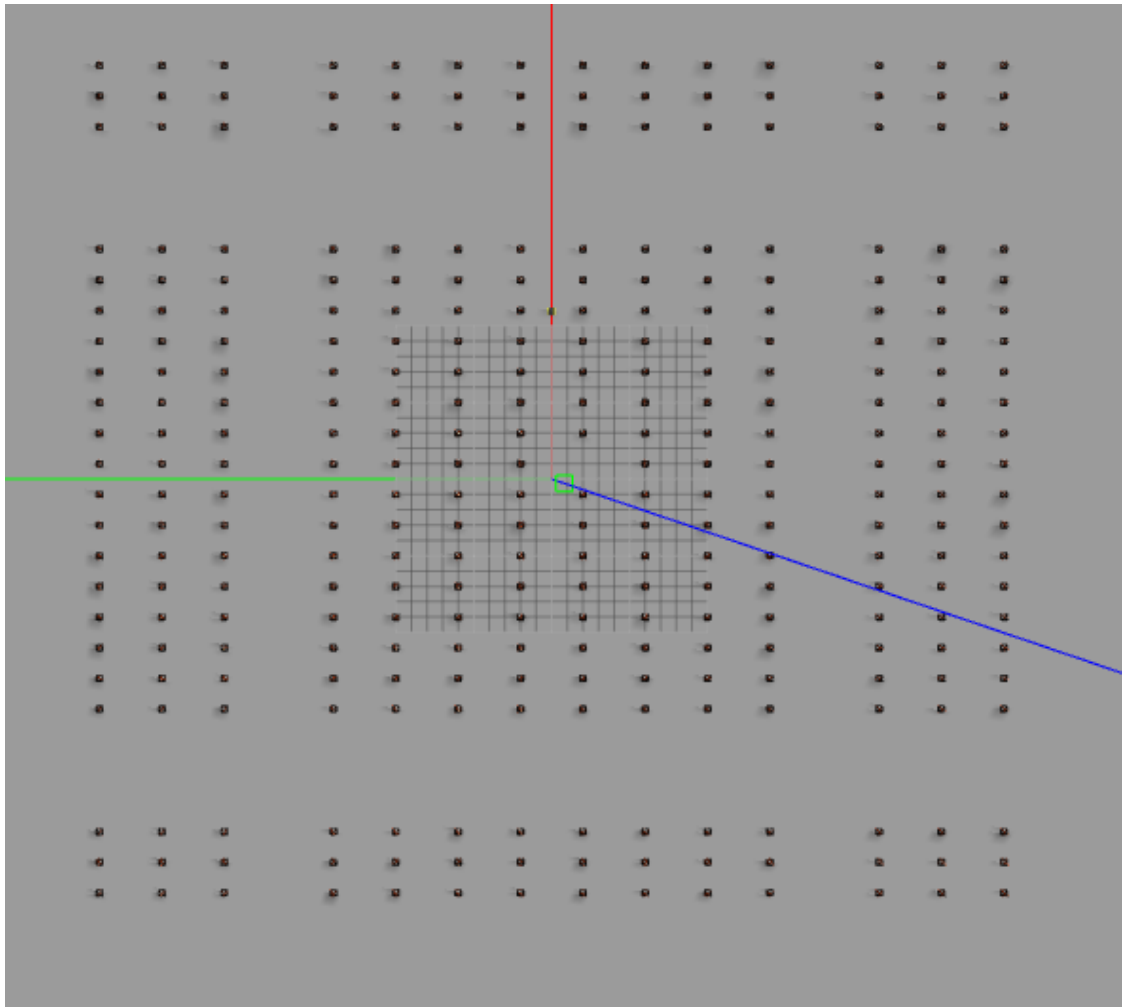
Experiments



Experiment #1 : single cone missing

- **Wold name**: cone_orcahrd5d
- **World Description:** single cone missing, near the origin.
- **Special notes**: Initial injected pose (8,8). 12,000 particles (max).
- **World map & image:**



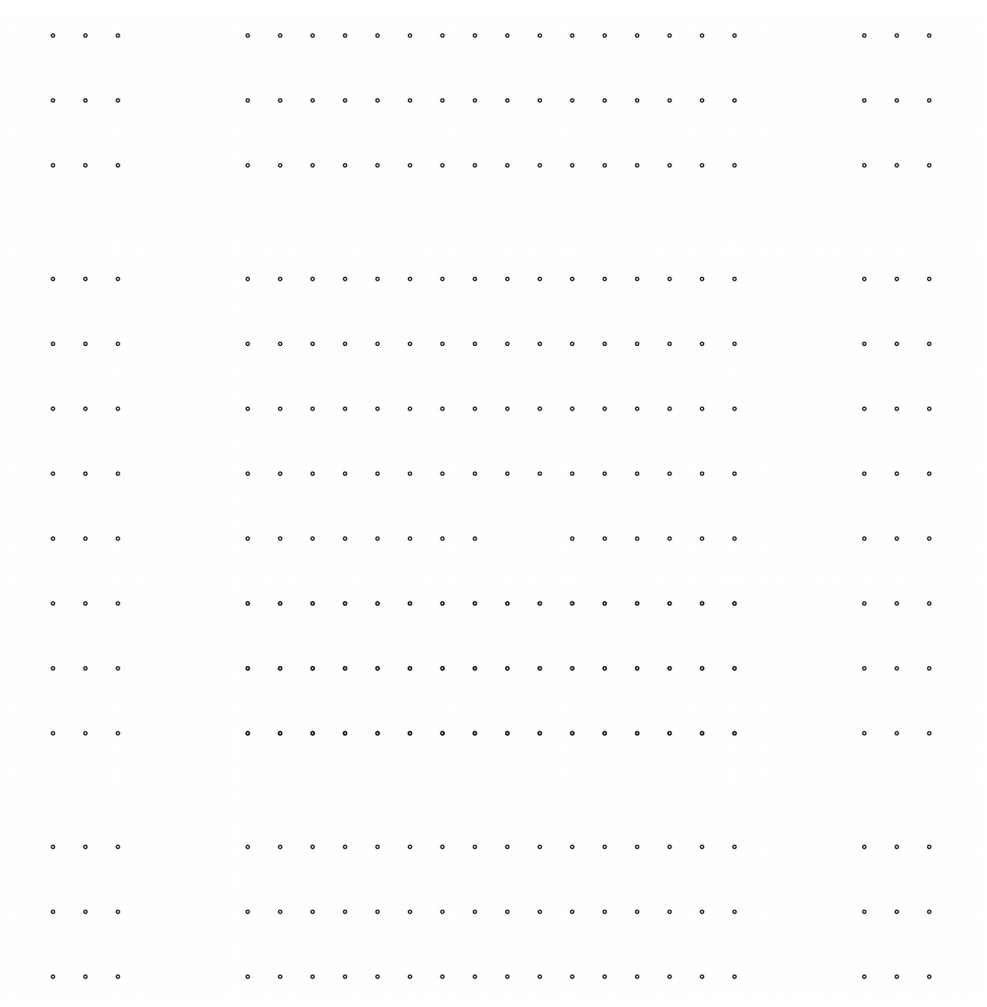


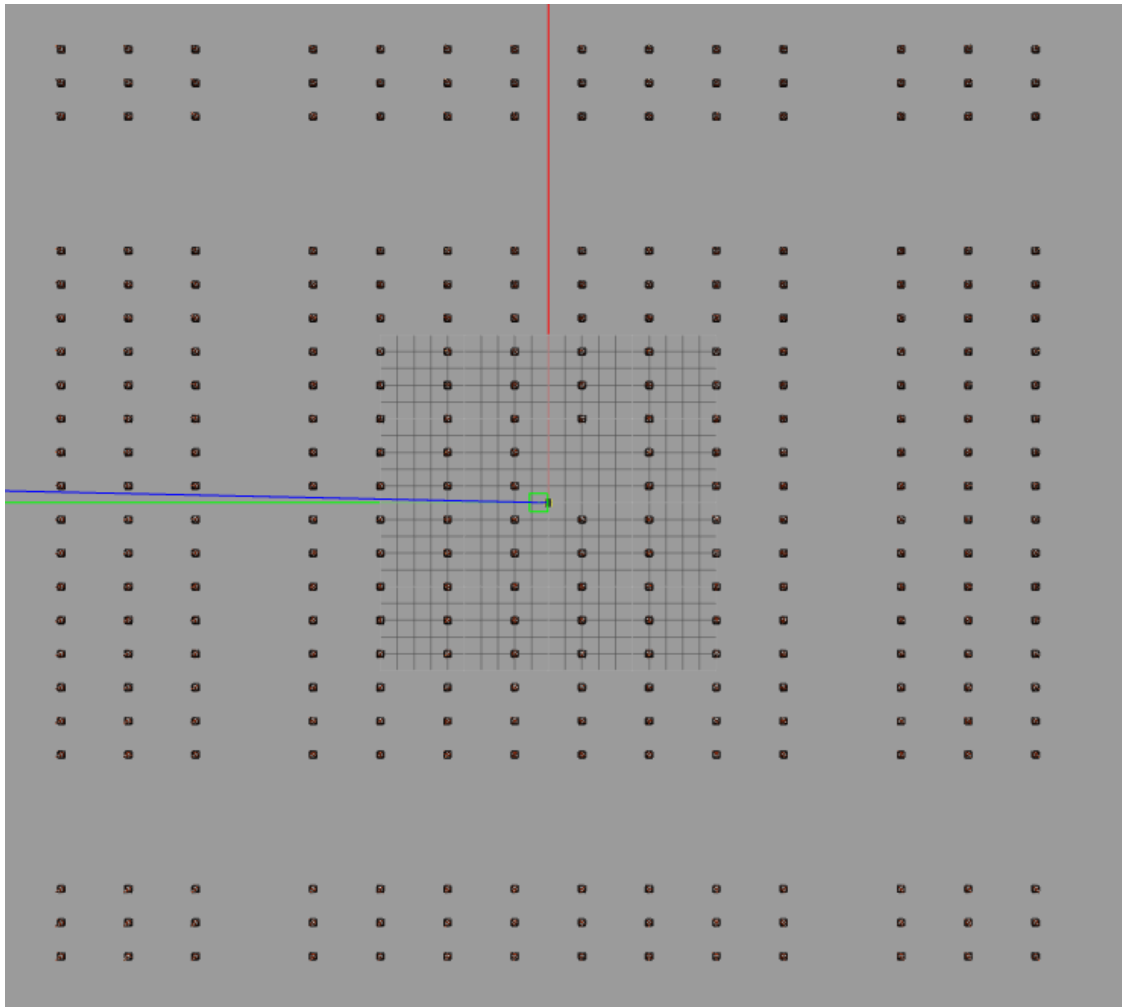
Experiment #1 : single cone missing

- **Results: 7** successive simulations- 0/10 (0%) success rate . I stopped there.
- Example success graph:-N.A
- There were other experiments with larger/smaller number of particles. None succeeded

Experiment #2 : two cones missing

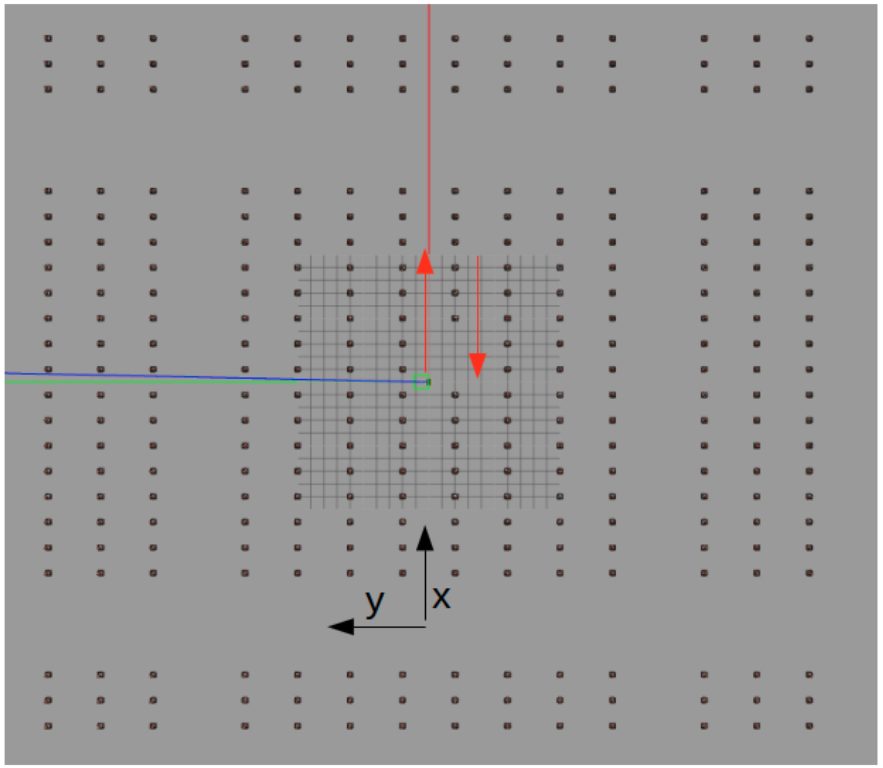
- **Wold name**: cone_orcahrd5b
- **World Description:** Two cones missing near the origin.
- **Special notes**: Initial injected pose (8,8). 12,000-30000 particles (max).
- **World map & image:**





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”.

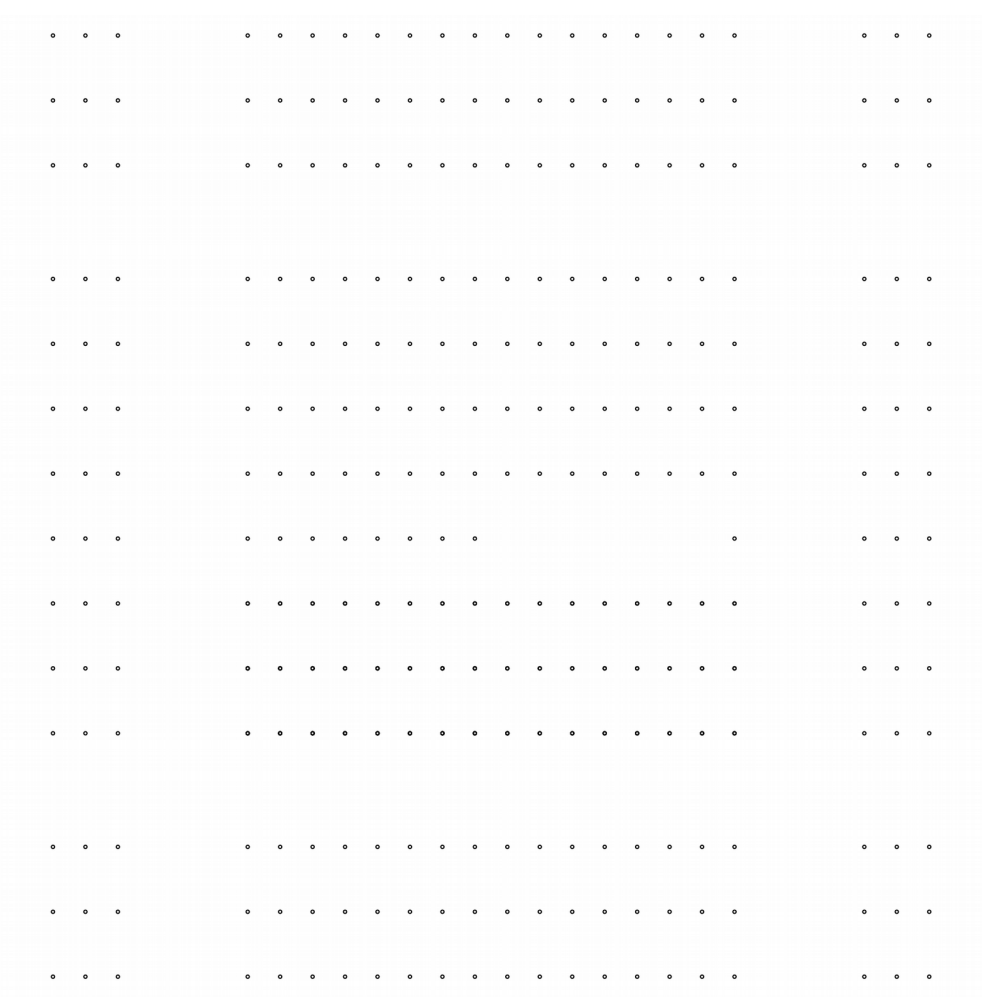


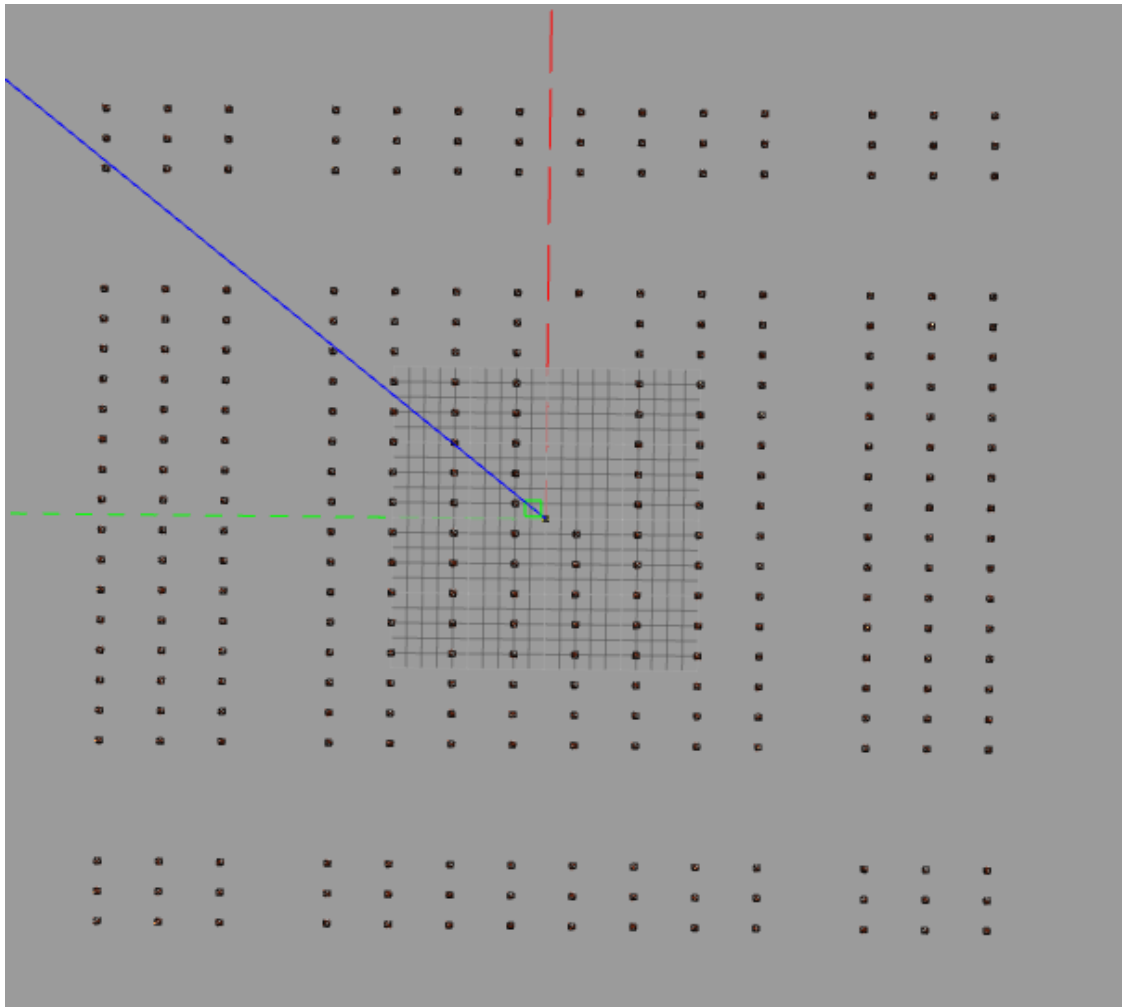
Experiment #2 : two cones missing

- **Results:** 10 successive simulations- **0**/22 (0%) success rate.
- Example success graph-None:

Experiment #3 : half row missing

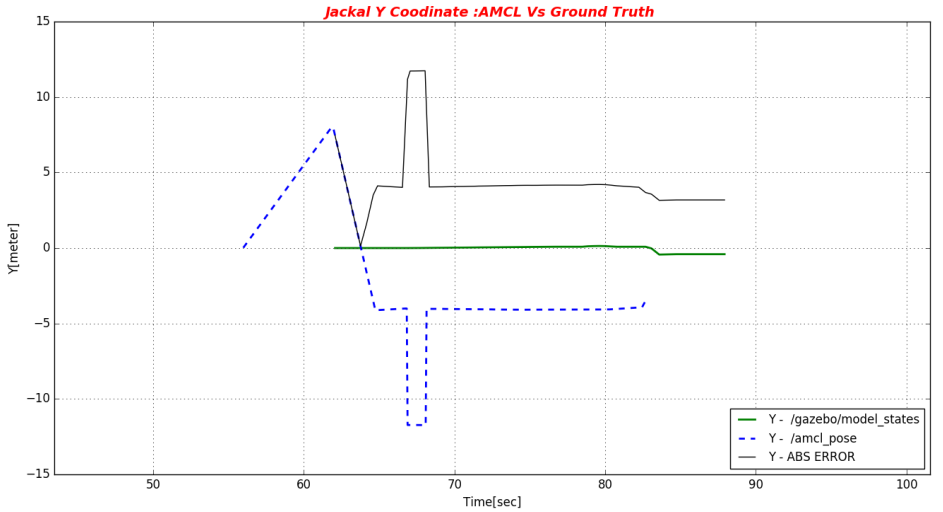
- **Wold name**: cone_orcahrd5c
- **World Description:** (nearly) half a row of cones missing.
- **Special notes**: Initial injected pose (8,8). 12,000 particles (max).
- **World map & image:**

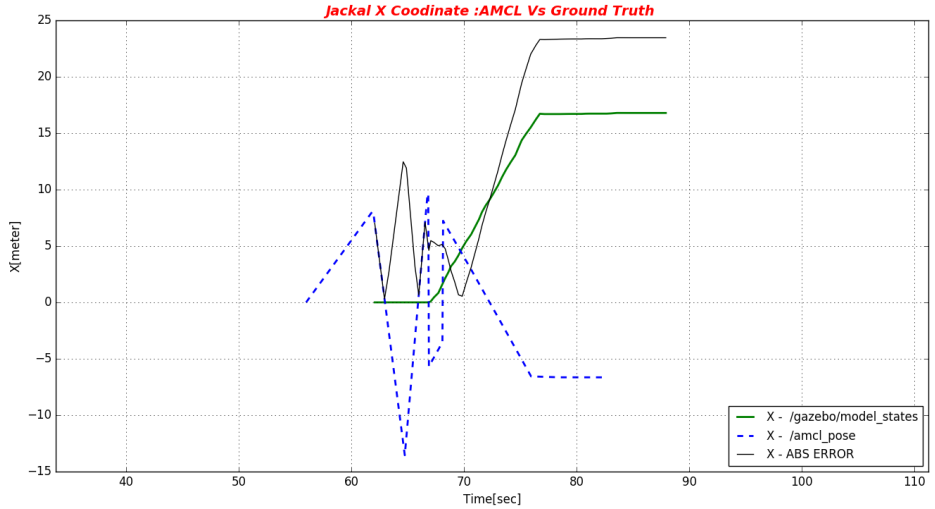




Experiment #3 : half row missing

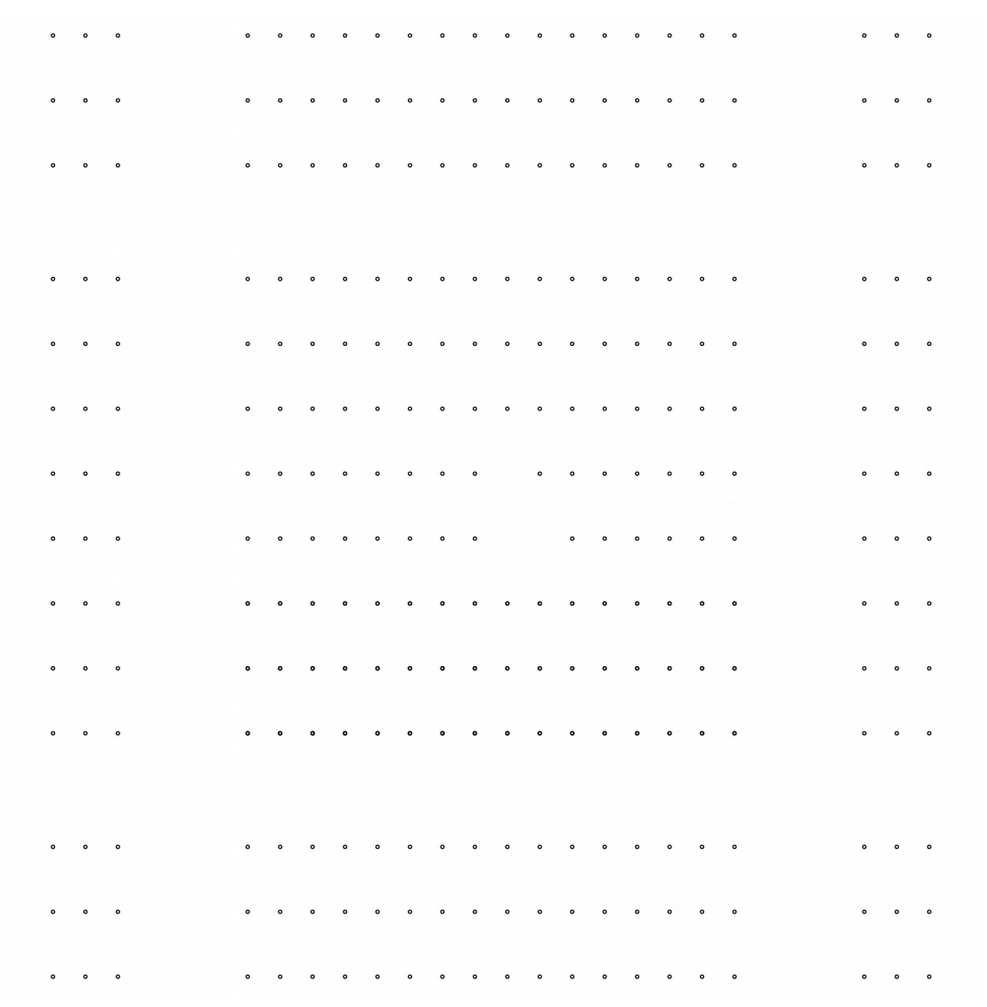
- **Results:** 10 successive simulations- 6/10 (60%) success rate .
- Example success graph:
- Note that this is an example of negative X-direction velocity and Y=-4 and not 0. which is considered valid No matters the X.

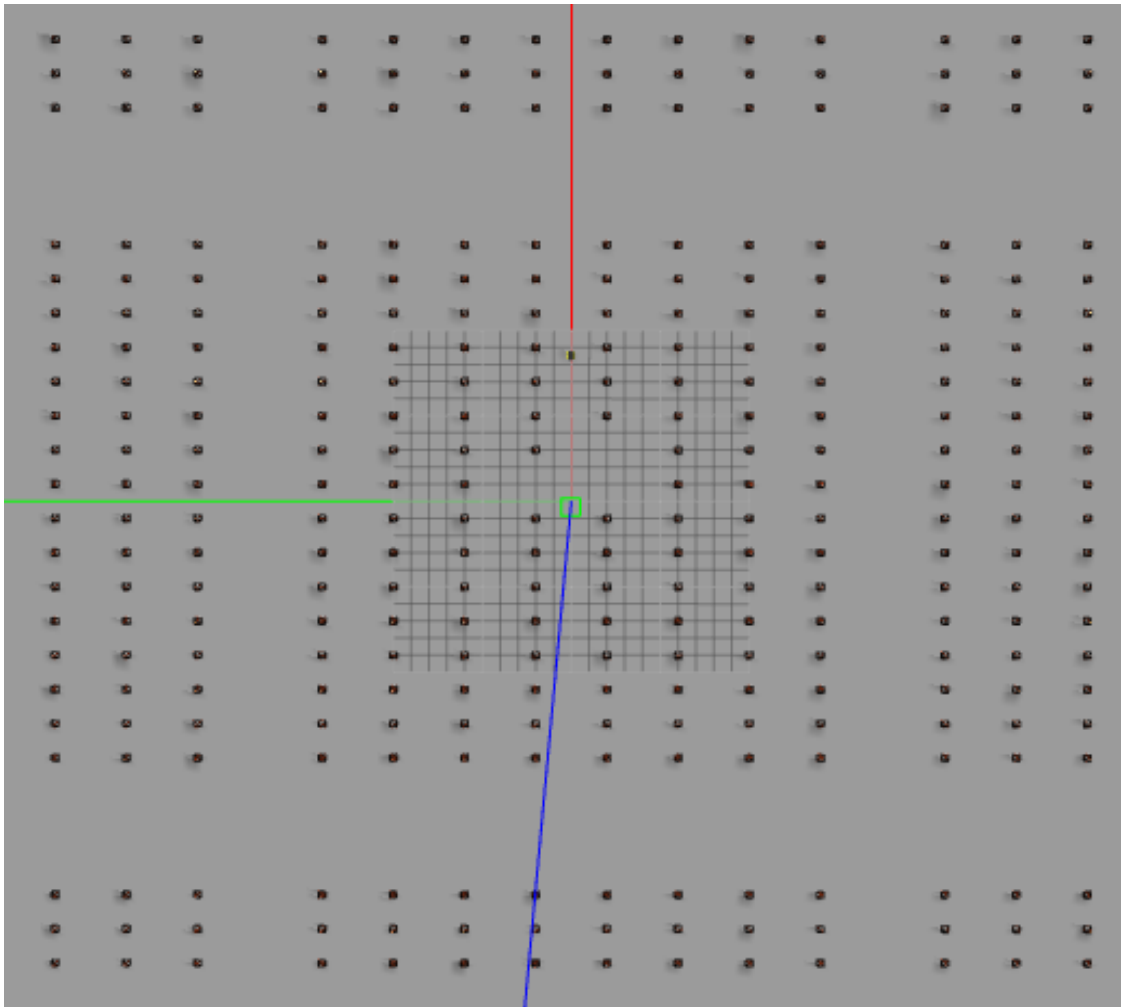




Experiment #4 : 3 cones missing both sides

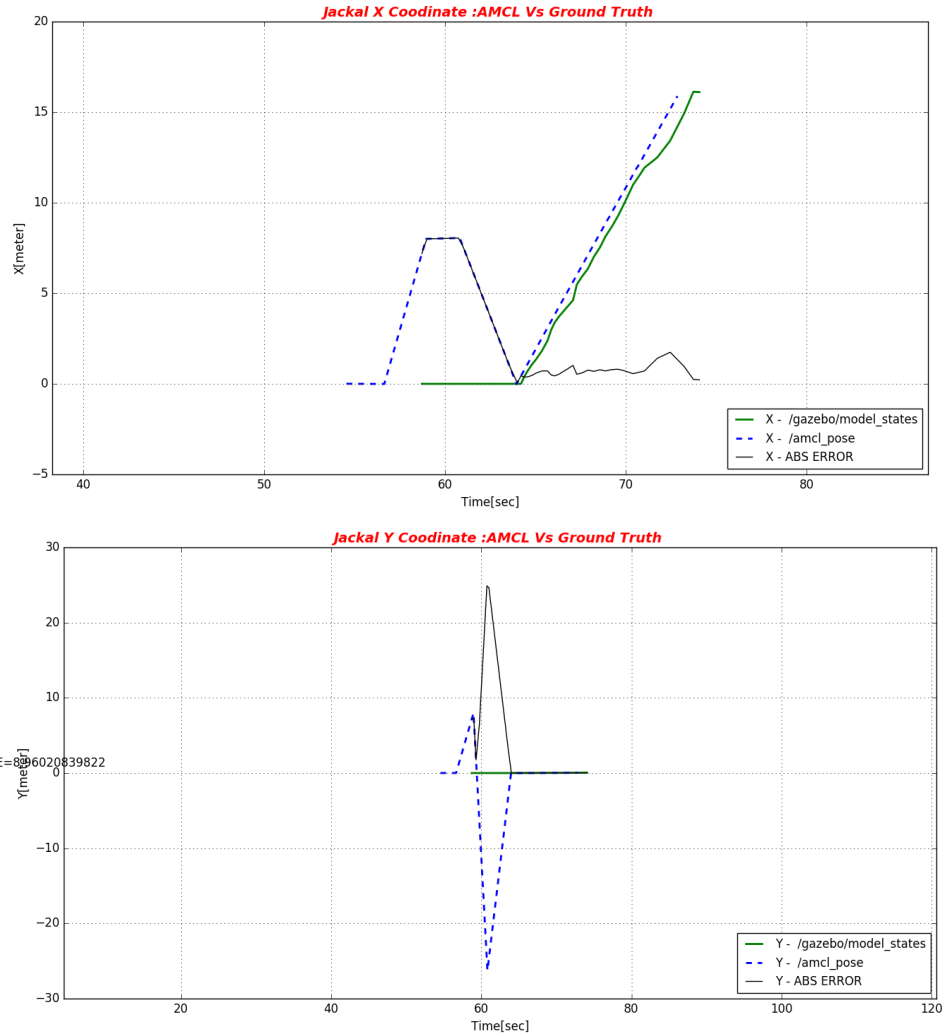
- **Wold name**: cone_orcahrd5e
- **World Description:** 3 cones missing near the origin, on both sides.
- **Special notes**: Initial injected pose (8,8). various number of particles tested .
- **World map & image:**





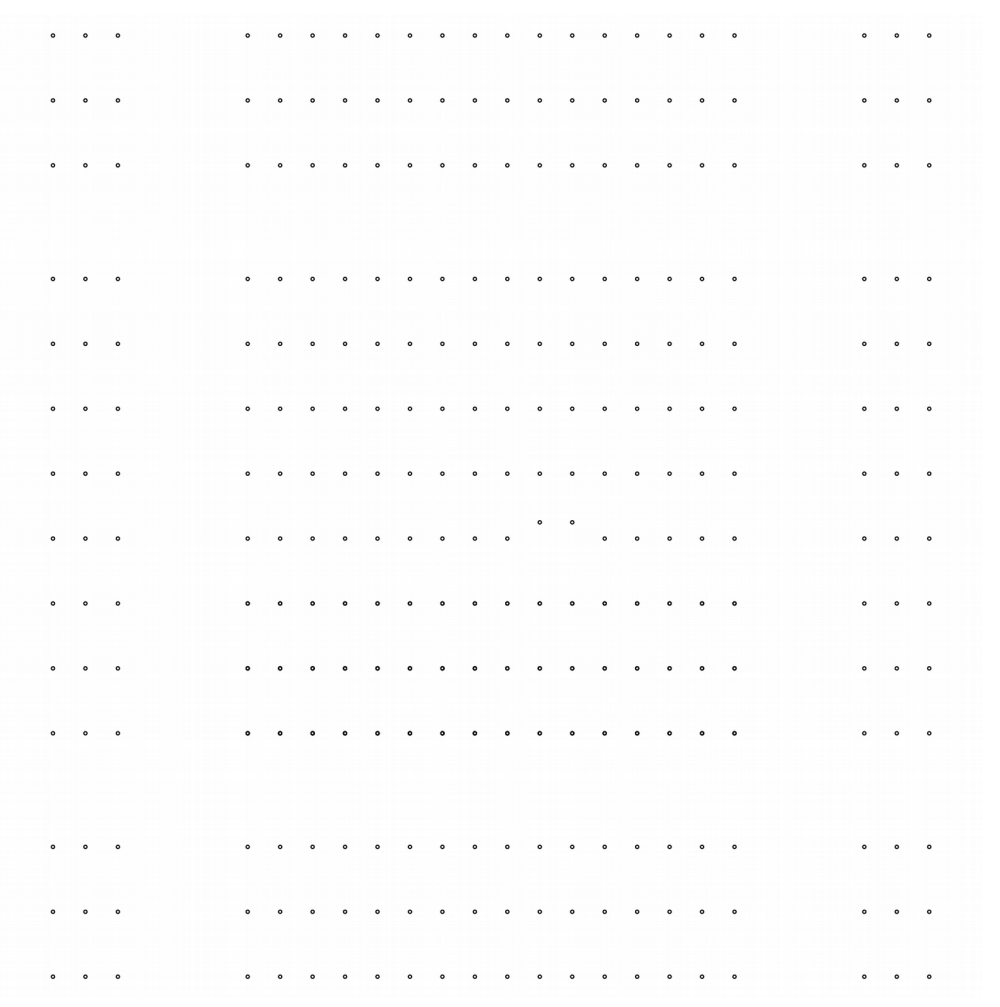
Experiment #4 : 3 cones missing both sides

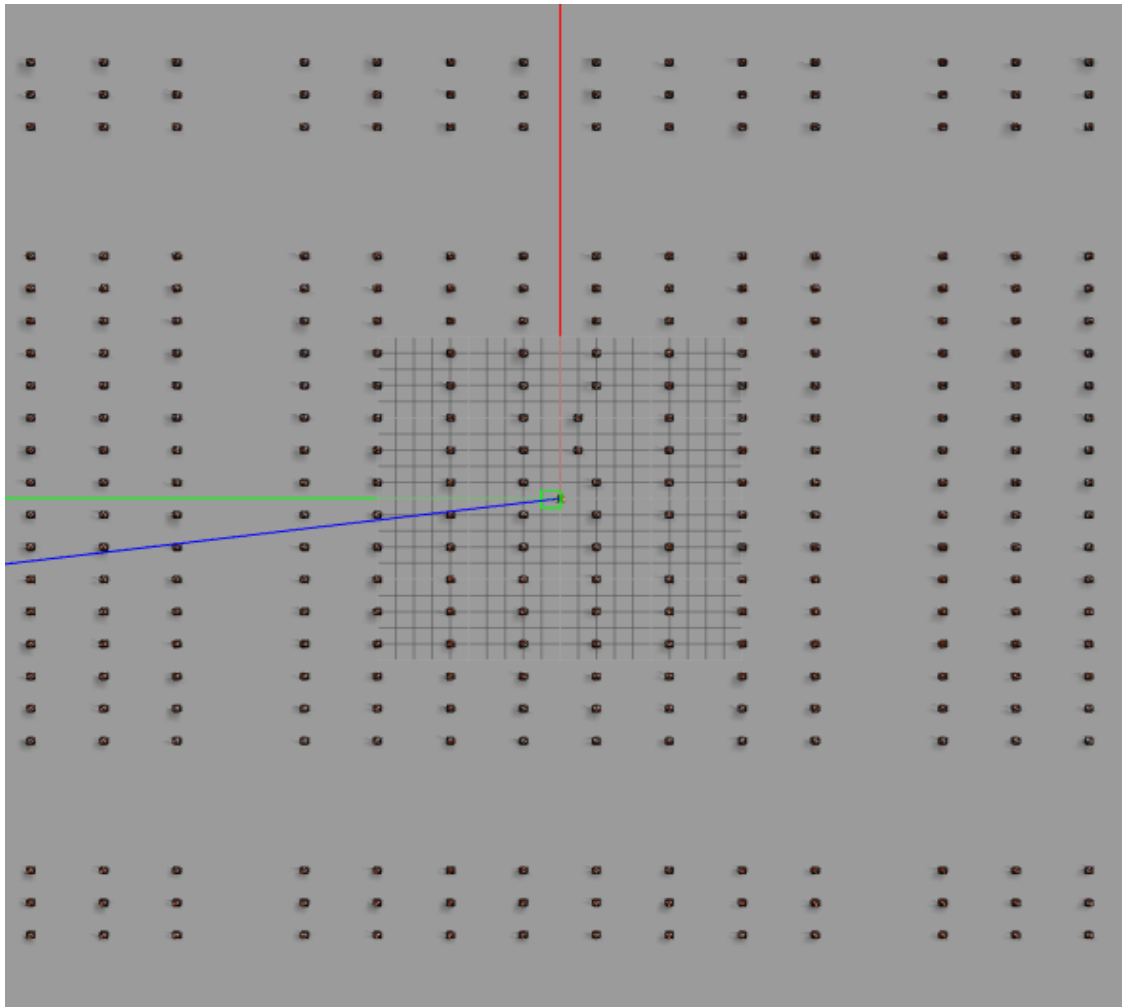
- Success rate- 4/6 with 18000 particles. Other numbers failed (30,000/12000) .
- **<u>The algorithm located the robot correctly in both axis.</u>**
- Example success graph:



Experiment #5 : two cones out of line

- **Wold name**: cone_orcahrd6b
- **World Description:** No cones missing, two cones are 1m ahead, near the origin.
- **Special notes**: Initial injected pose (8,8). various number of particles tested .
- **World map & image:**



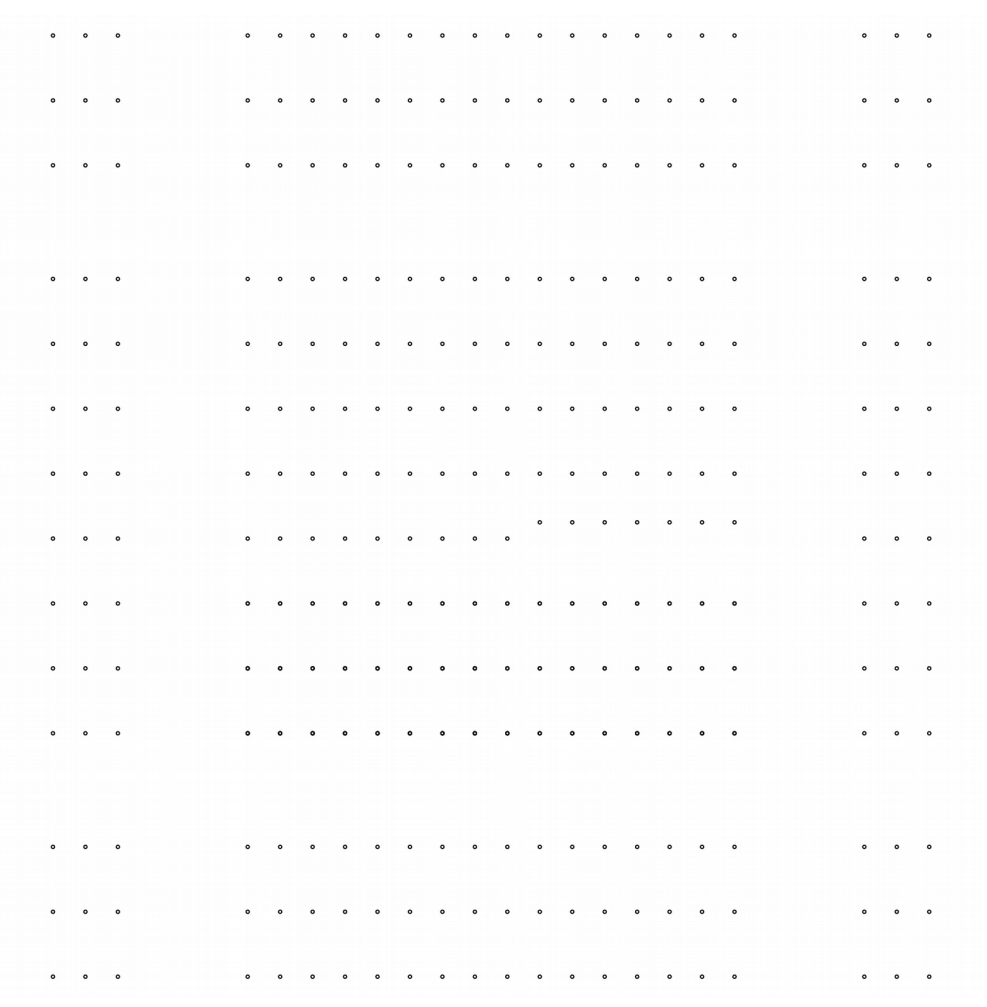


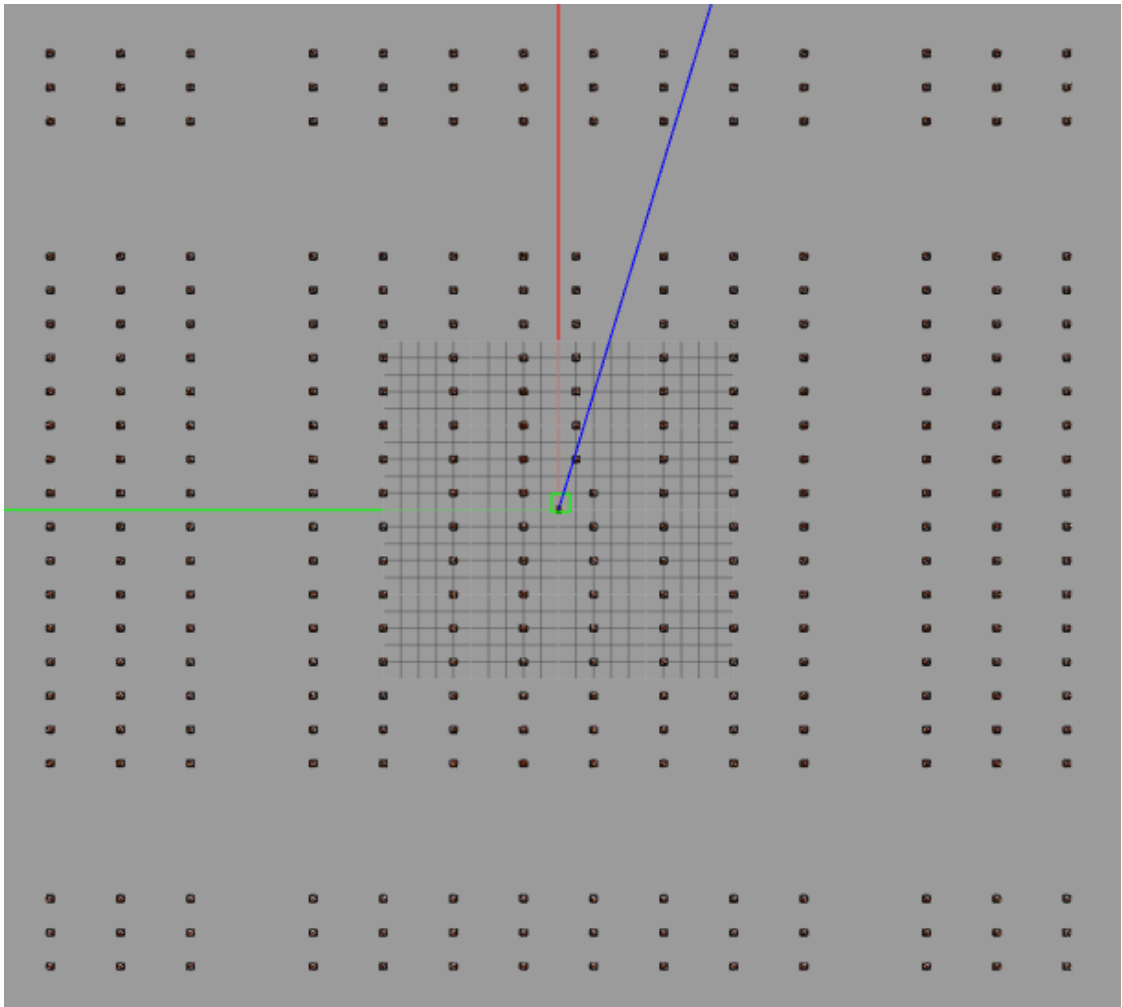
Experiment #5 : two cones out of line

- Results: 7 successive simulations- 0/7(0%) success rate . I stopped there. Many other experiments which were not saved showed the same result.
- Example success graph:-N.A
- Experiments within the interval of 8000-30000 particles were made.

Experiment #6 : half row of cones out of line

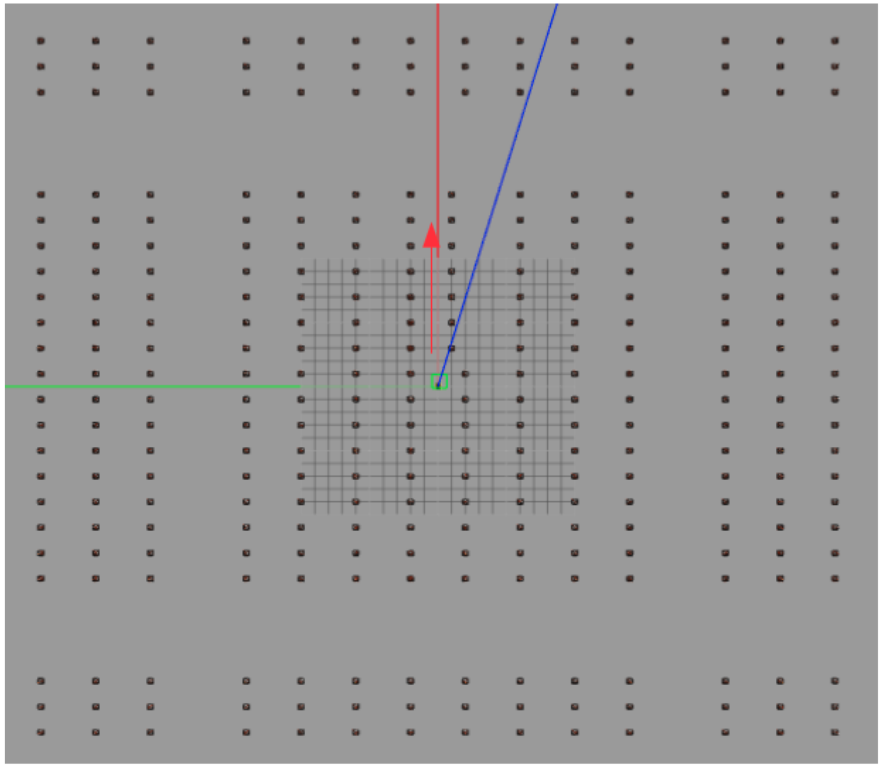
- **Wold name**: cone_orcahrd6b
- **World Description:** half a row of cones are 1m ahead.
- **Special notes**: Initial injected pose (8,8). various number of particles tested .
- **World map & image:**





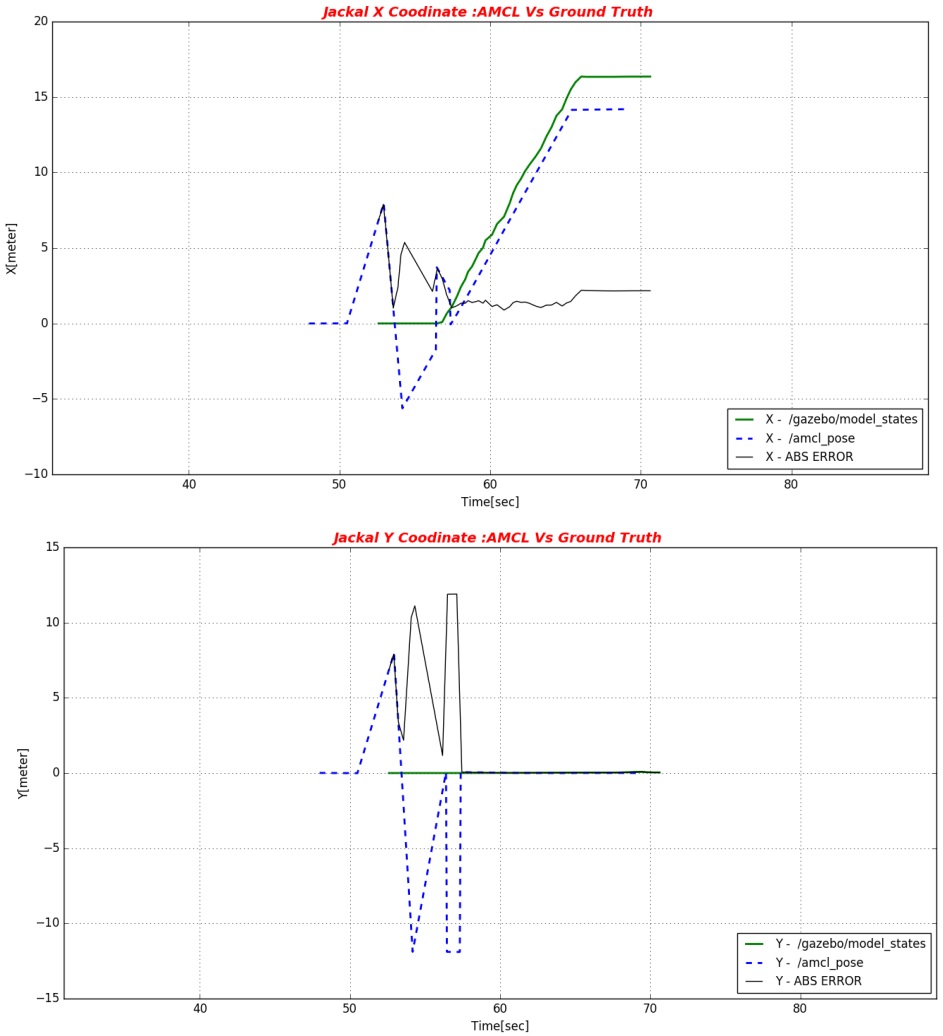
Experiment #5 : half row of cones out of line

- **Special note**: “Success” in this experiment is considered a localization solution somewhere inside the correct row (y=0) , while the robot is heading in the right (positive x) direction
- **Results**:
  - 30,000 particles- 7/8 succeeded (87.5%).
  - Less (12,000 and 20,000 particles)-None (0% succeeded).



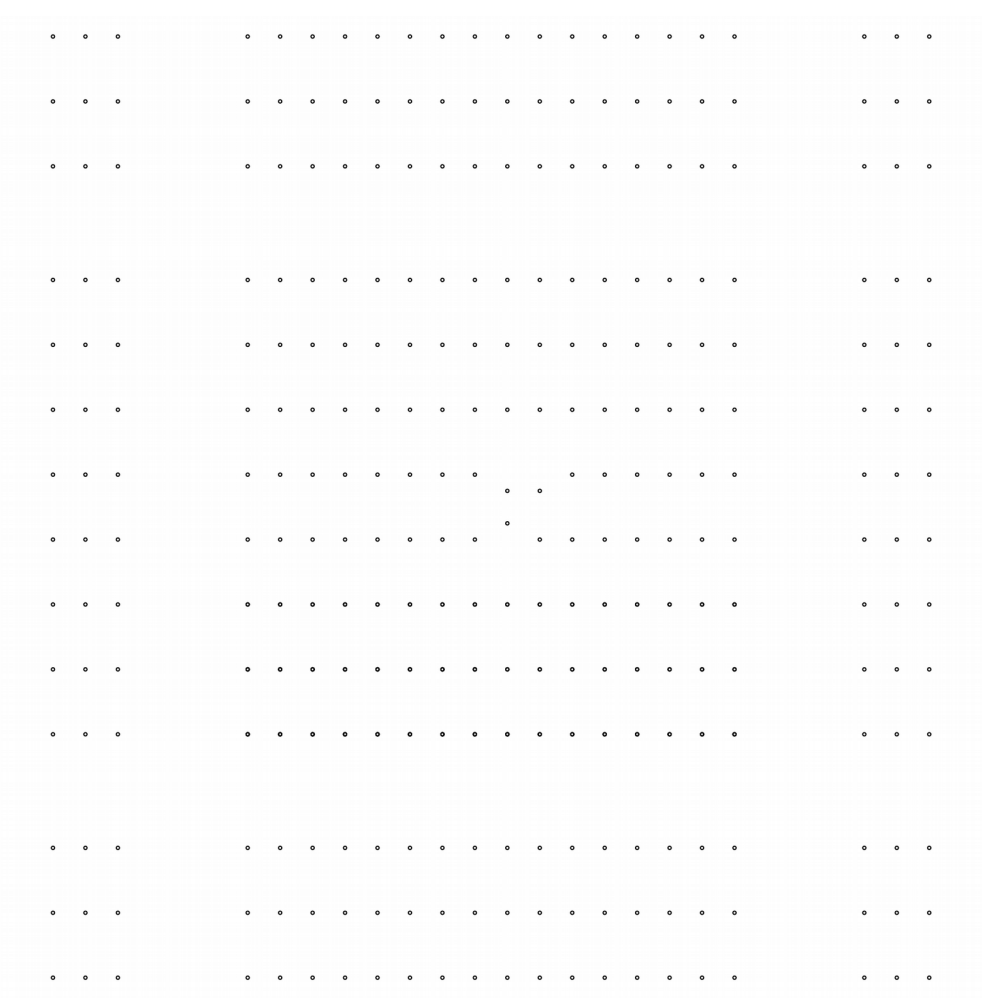
Experiment #6 : half row of cones out of line

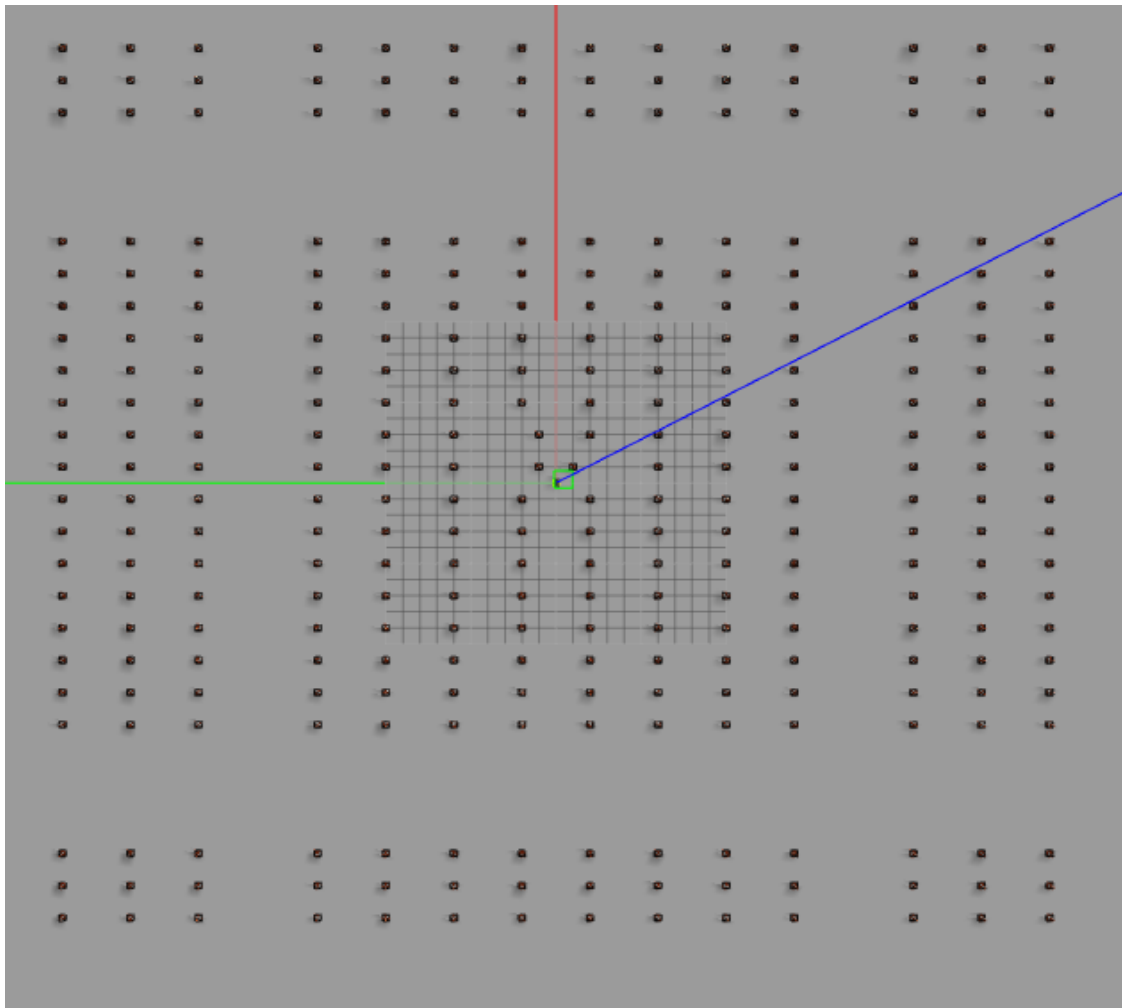
- Example success graph:



Experiment #7 : 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**: Initial injected pose (8,8). various number of particles tested .
- **World map & image:**



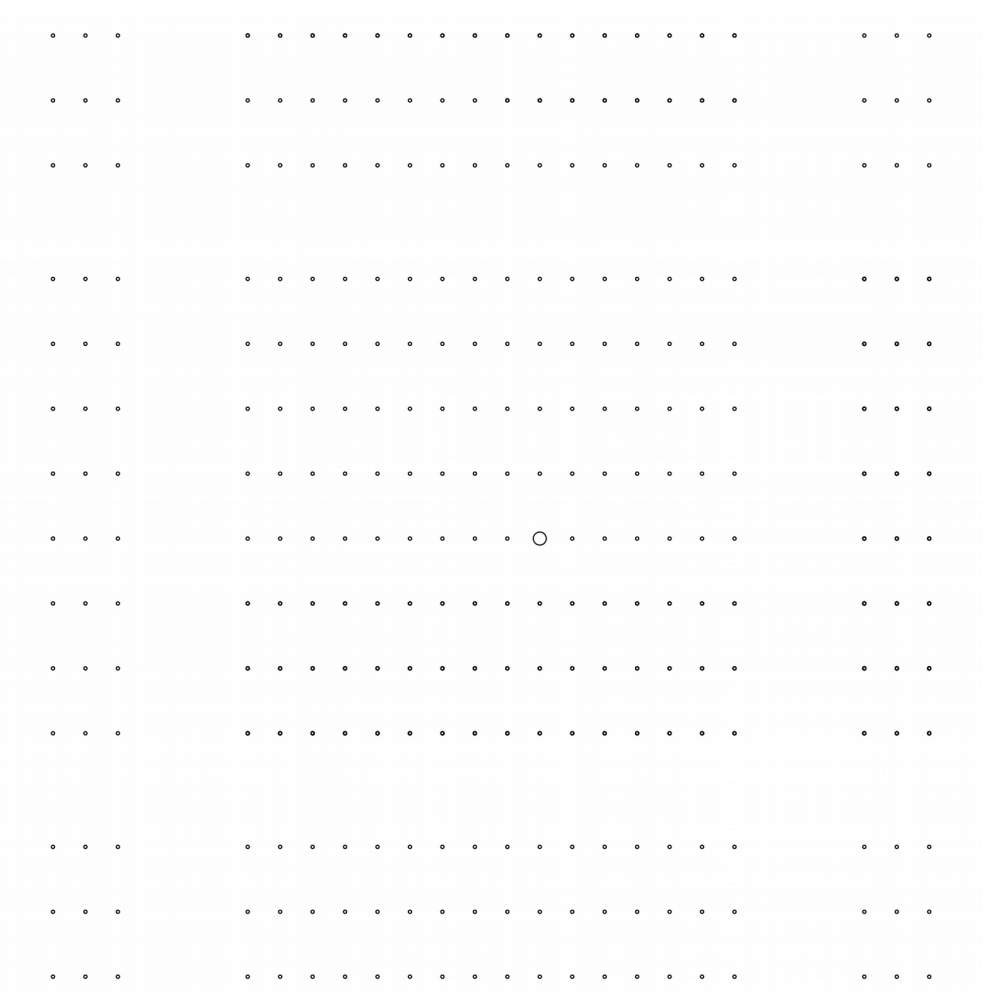


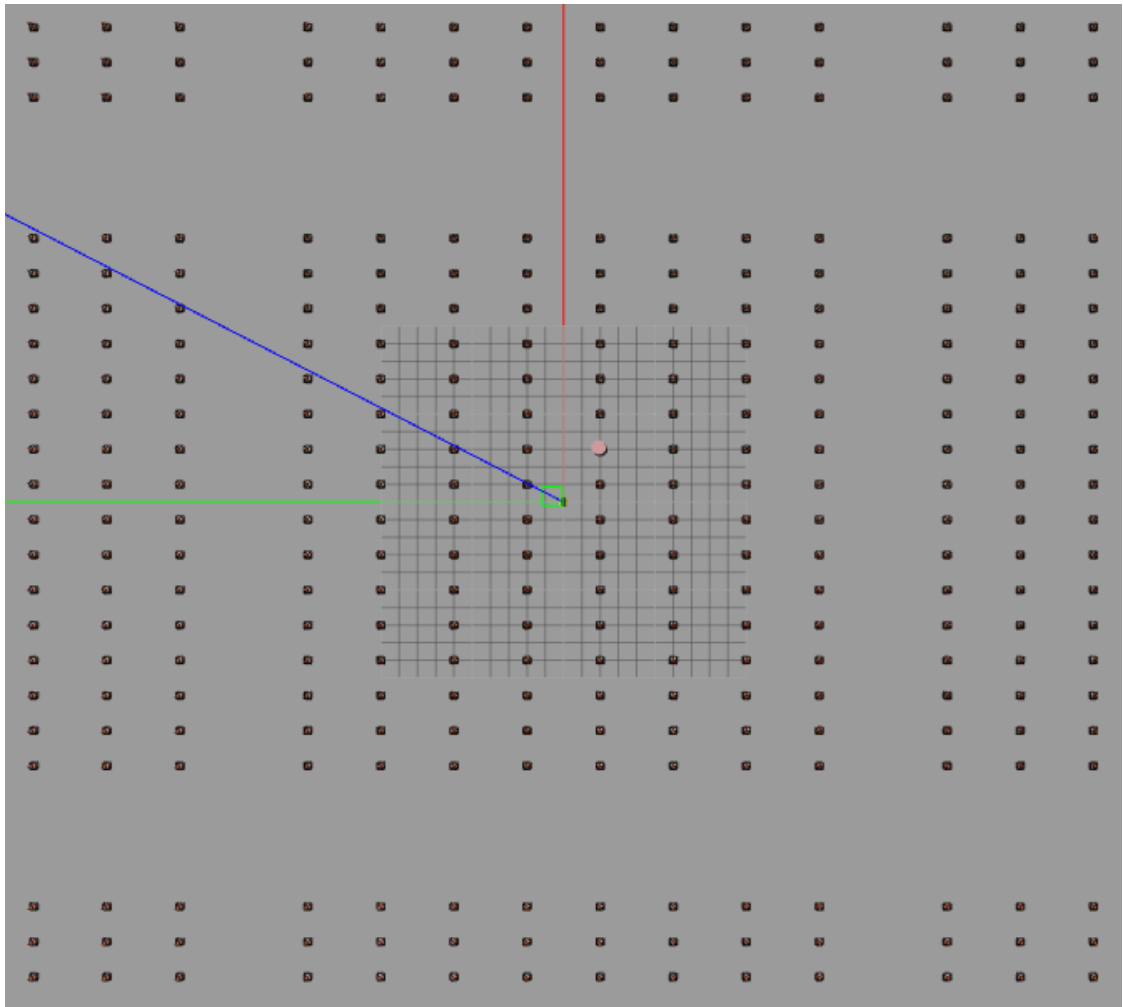
Experiment #7 : 3 cones out of line both sides

- 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.

Experiment #8 : Single cylinder

- **Wold name**: cone_orcahrd6c
- **World Description:** A single cylinder with radius=0.4m (4 times larger than a cone) near the origin.
- **Special notes**: Initial injected pose (8,8). various number of particles tested .
- **World map & image:**



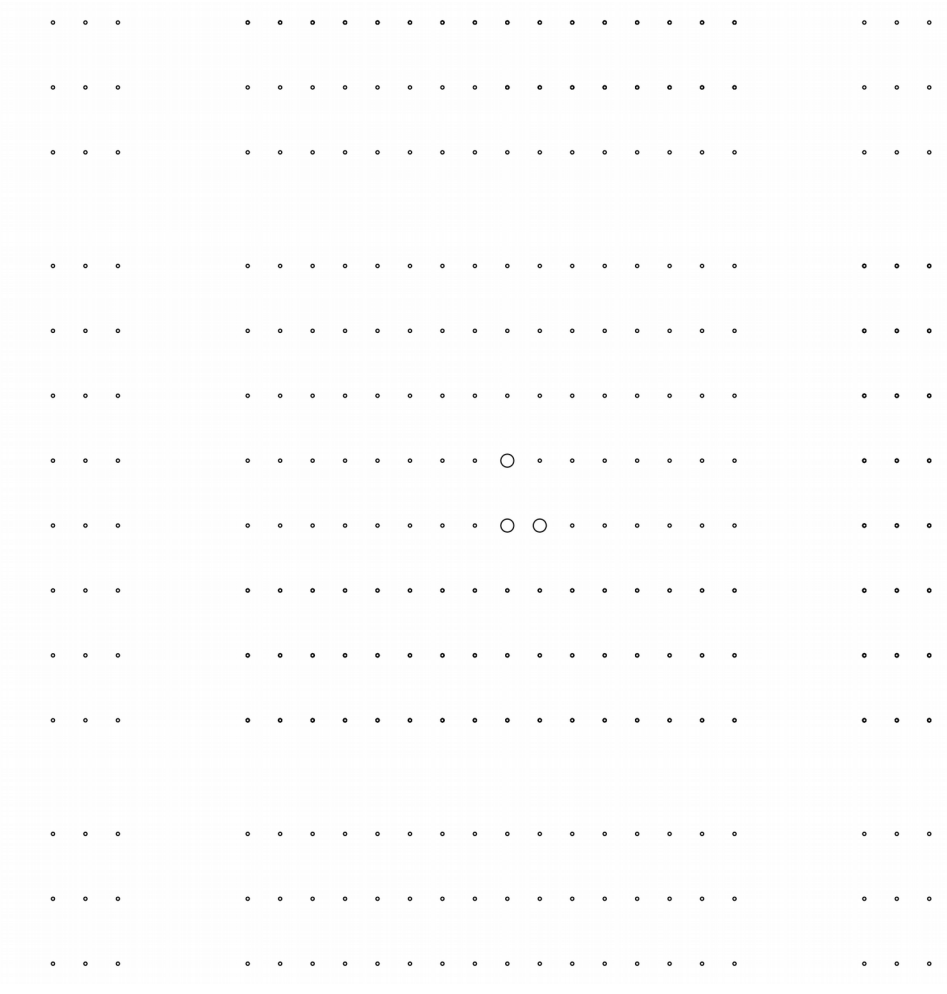


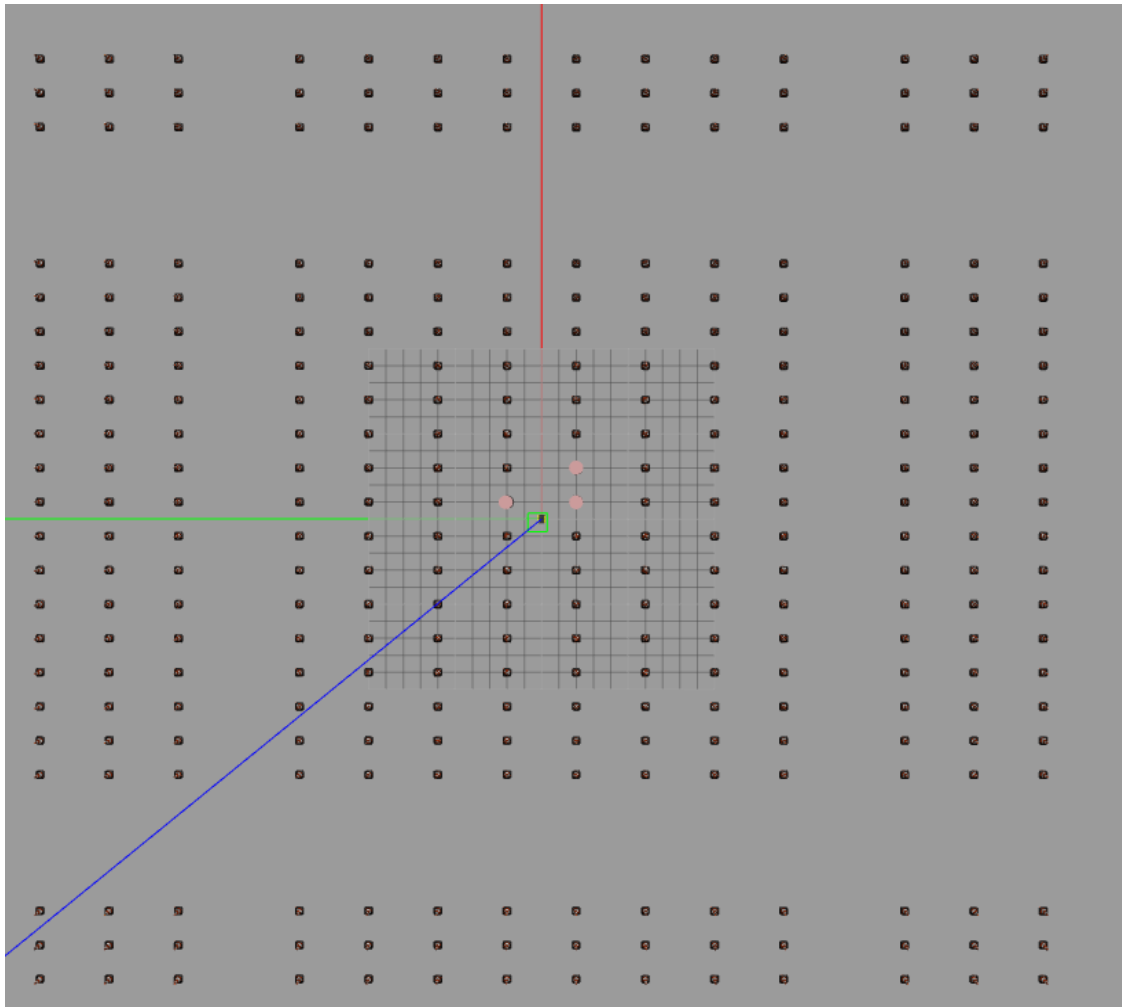
Experiment #8 : Single cylinder

- Results: 7 successive simulations- 1/7(14%) success rate . I stopped there.
- Experiments within the interval of 8000-30000 particles were made. Only the 30,000 particles simulations succeeded, once. Even that was with the “mirror” solution – meaning y=-4 instead of y=0 row.

Experiment #9 : 3 cylinders both sides

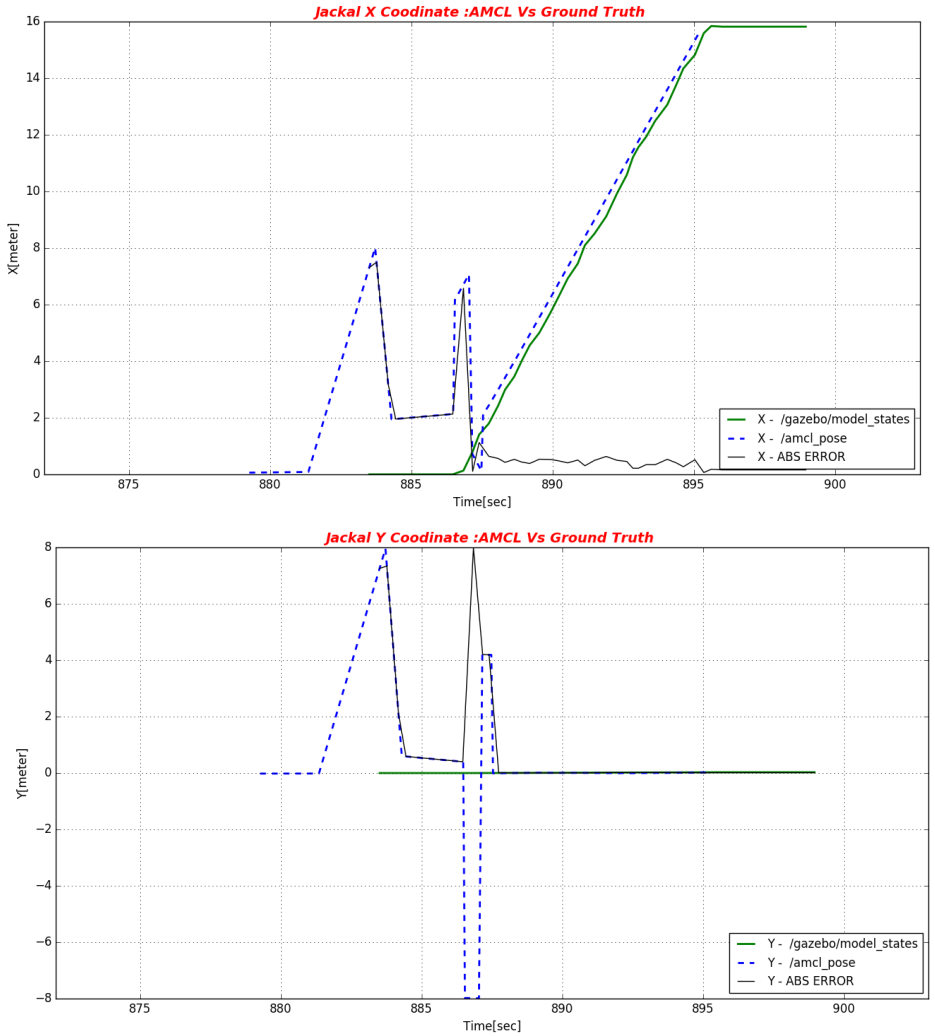
- **Wold name**: cone_orcahrd6d
- **World Description:** 3 cylinders with radius=0.4m (4 times larger than a cone) near the origin.
- **Special notes**: Initial injected pose (8,8). various number of particles tested .
- **World map & image:**





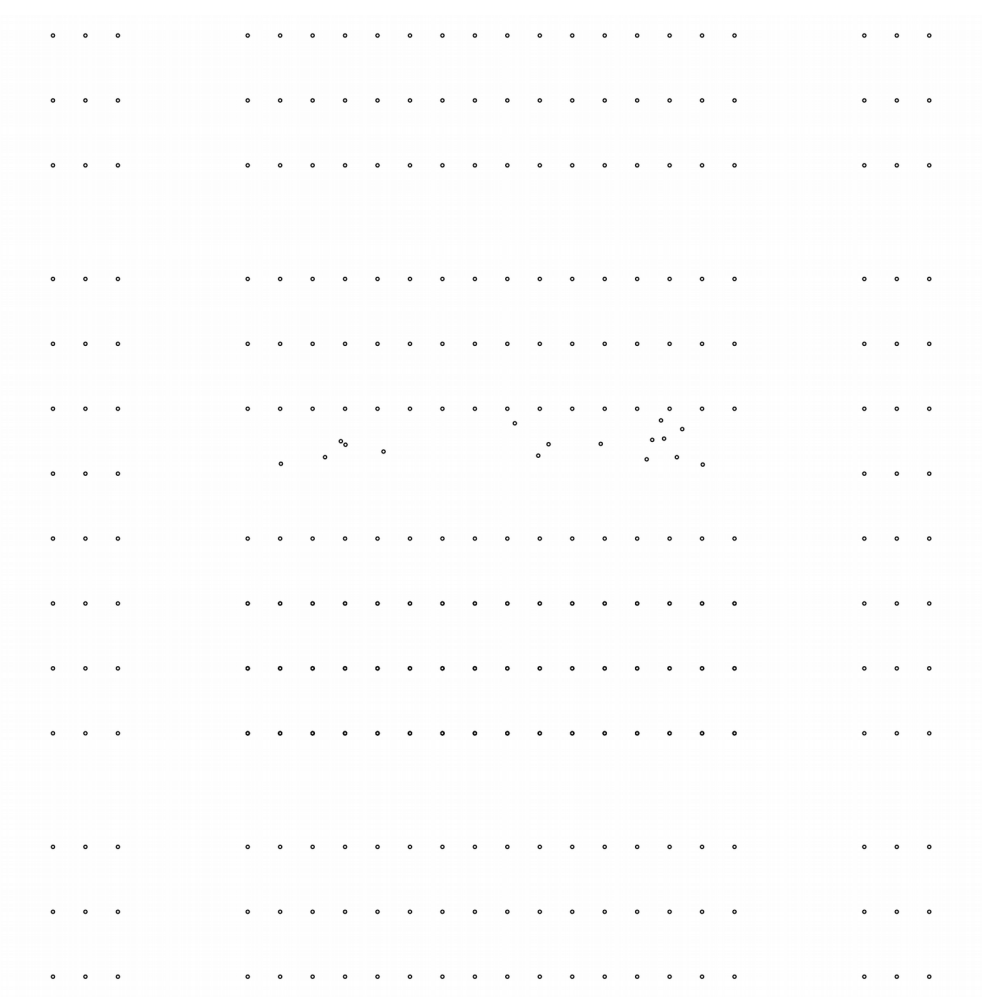
Experiment #9 : 3 cylinders both sides

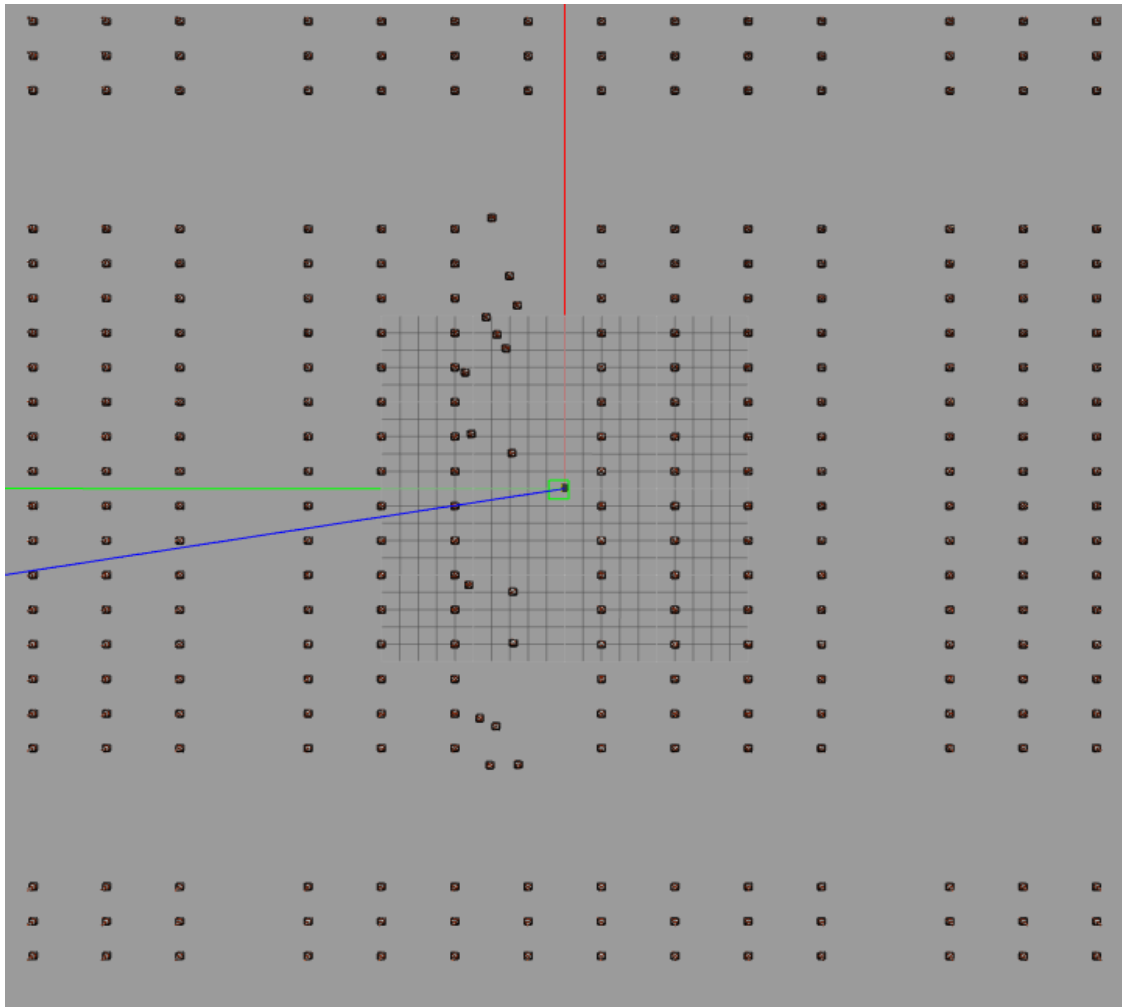
- Results: 5 successive simulations- 5/5 (100%) success rate with 3000 particles.
- 20,000 particles-1/3 success (33%). Using less particles turned to be worse.
- **The algorithm located the robot correctly in both axis**
- Example success graph:



Experiment #10 : random row of cones

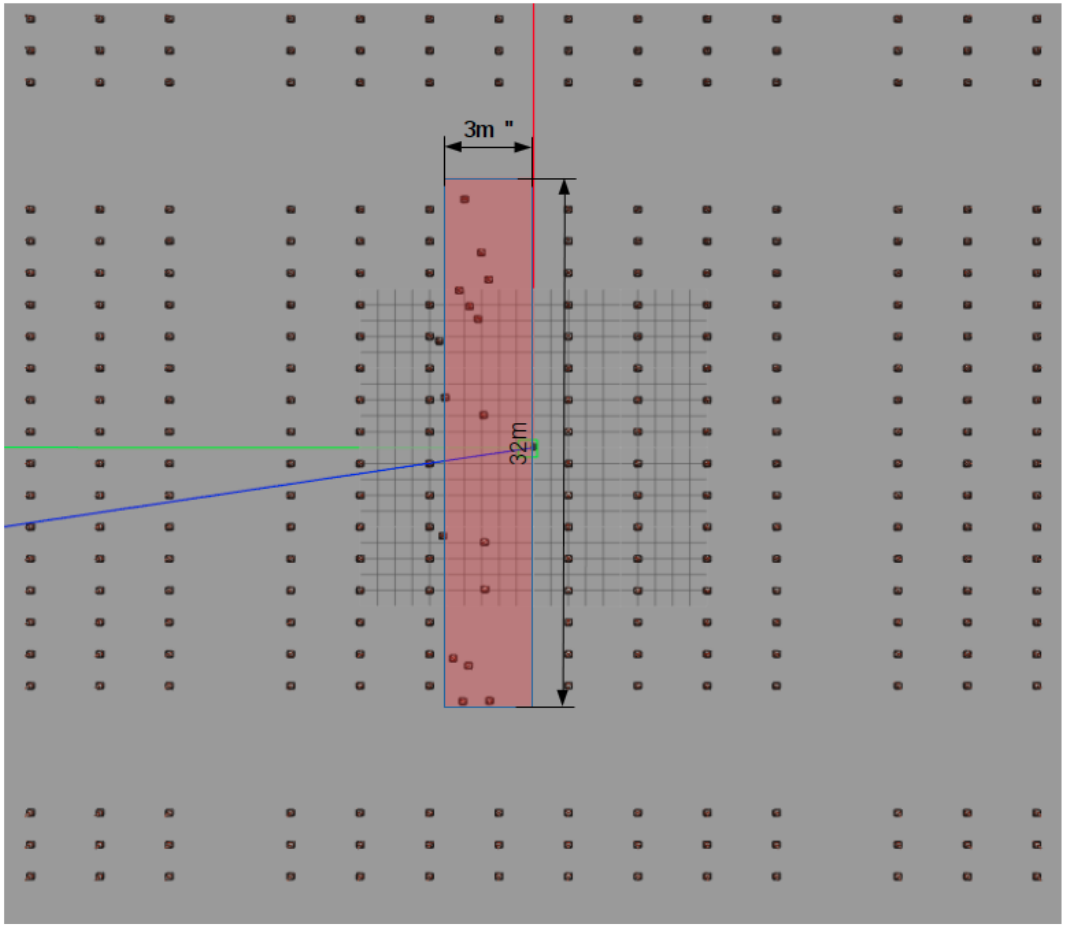
- **Wold name**: cone_orcahrd7a
- **World Description:** a row of cones ordered randomly within a rectangle of 30[m]X3[m].
- **Special notes**: Initial injected pose (8,8). 30000 particles tested .
- **World map & image:**





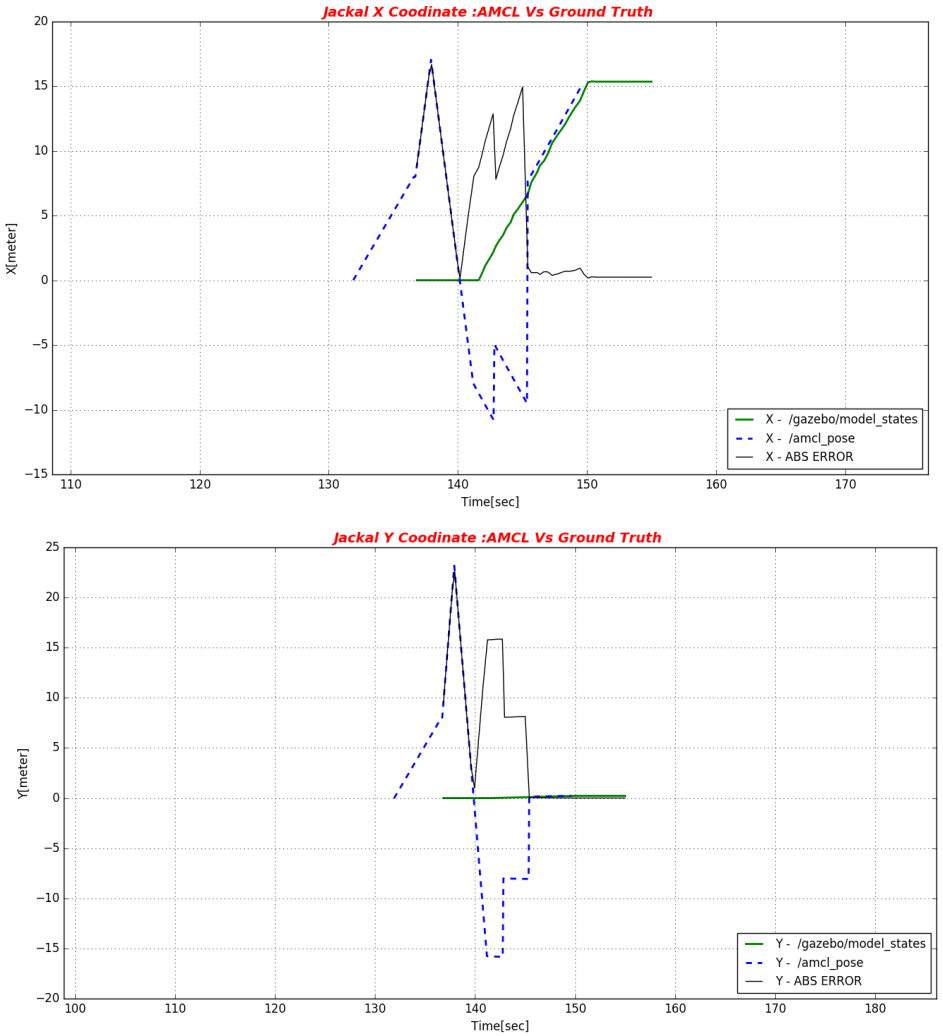
Experiment #10 : random row of cones

- **Wold name**: cone_orcahrd7a
- **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.



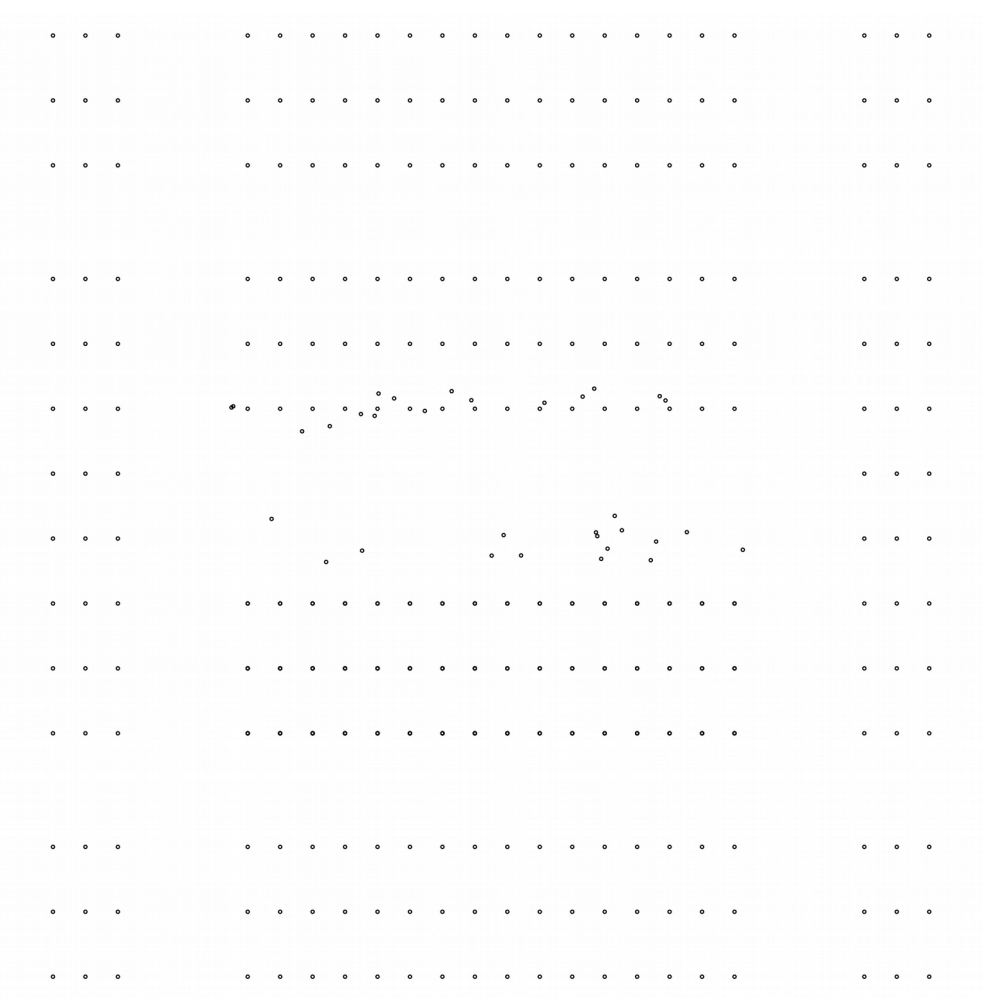
Experiment #10 : random row of cones

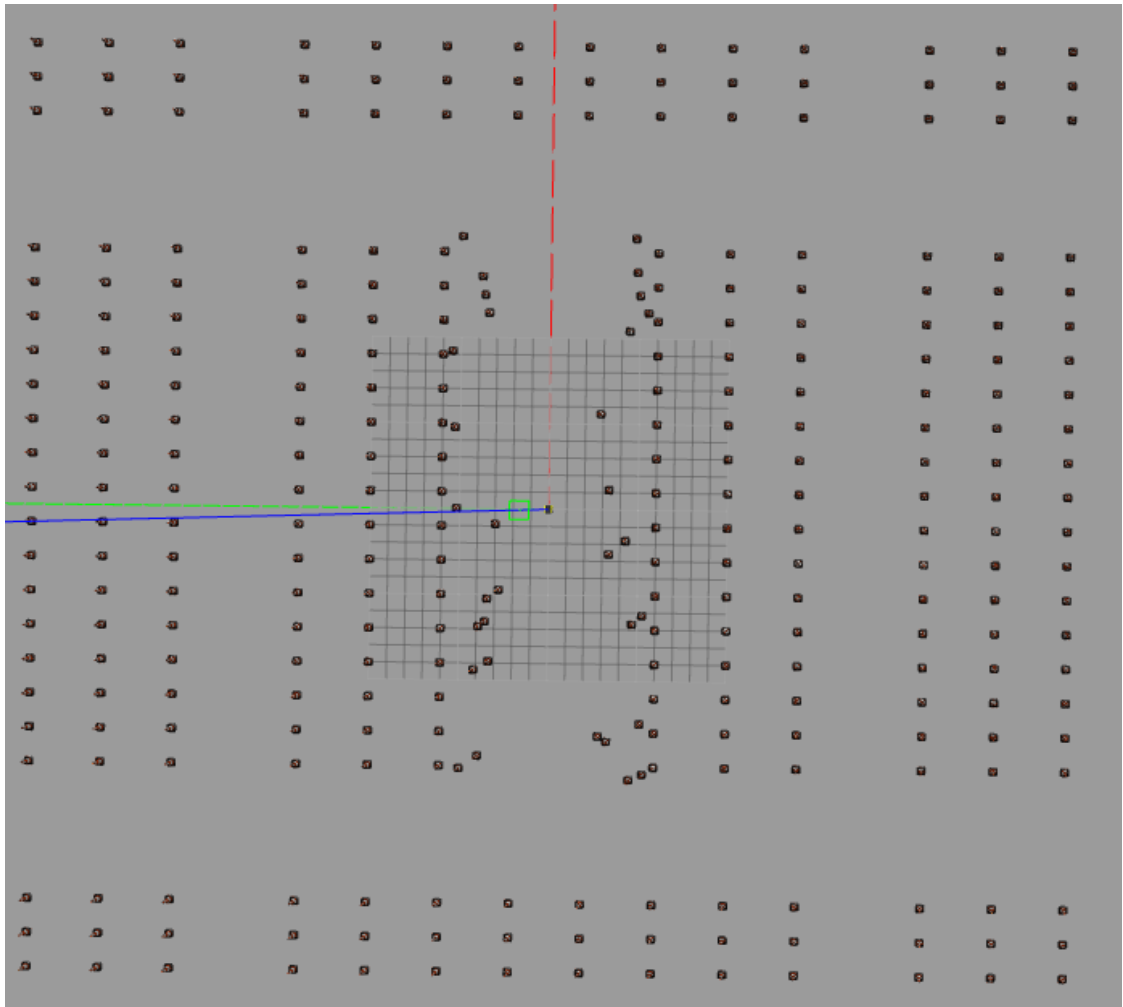
- Results: 8 successive simulations- 4/10 (40%) success rate with 30000 particles.
- **<u>The algorithm located the robot correctly in both axis</u>**
- Example success graph:



Experiment #11 : two random rows of cones

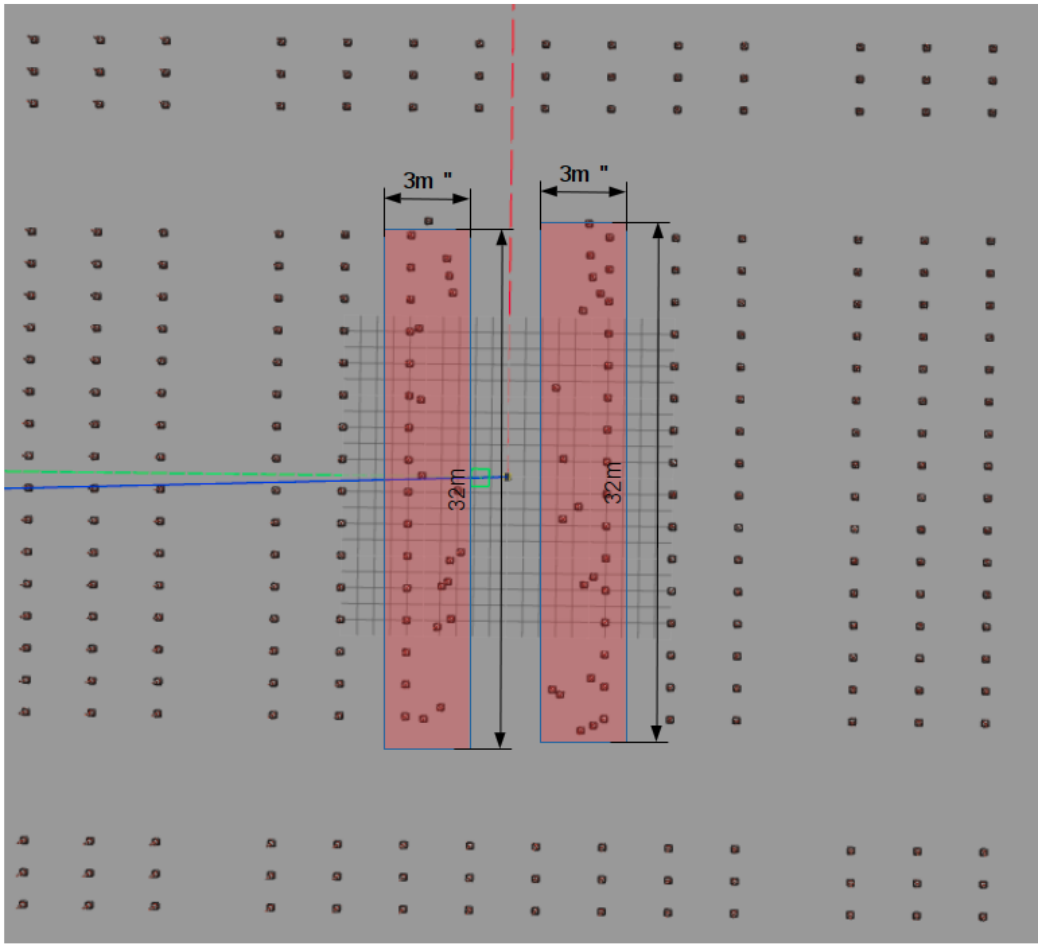
- **Wold name**: cone_orcahrd7b
- **World Description:** two row of cones ordered randomly within a rectangle of 30[m]X3[m] on both sides of the origin.
- **Special notes**: Initial injected pose (8,8). 30000 particles tested .
- **World map & image:**





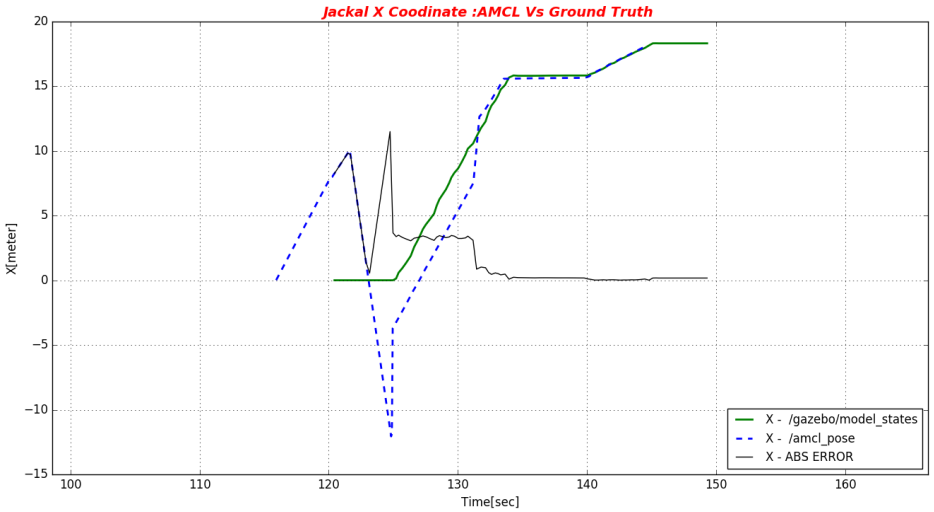
Experiment #11 :two random rows of cones

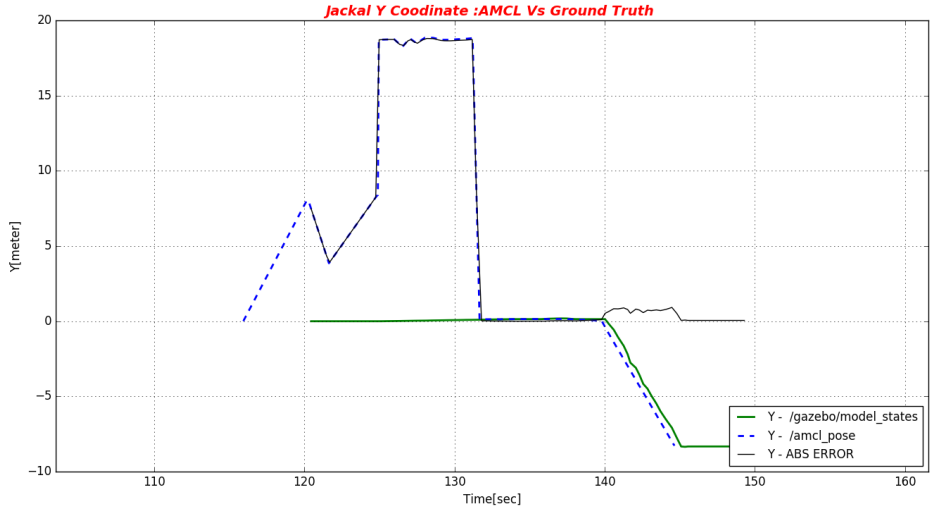
- **Wold name**: cone_orcahrd7b
- **Special notes:** The “random row” was created using gazebo 7 “population of models” feature. See http://gazebosim.org/tutorials?tut=model_population&cat=
- As a result, each experiment has been carried out on a **different** world, using a different **new** map created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



Experiment #11 : two random rows of cones

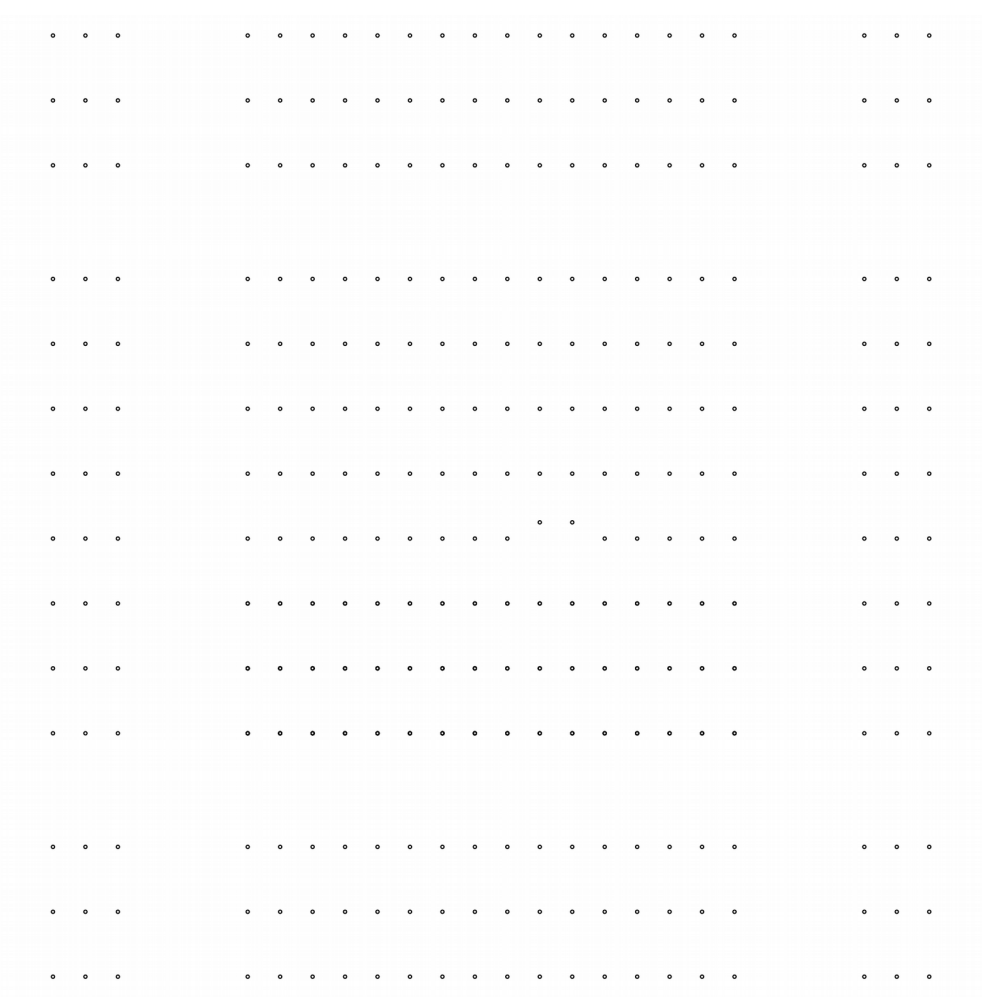
- Results: 10 successive simulations- 8/10 (80%) success rate with 30000 particles.
- **<u>The algorithm located the robot correctly in both axis</u>**
- Example success graph:

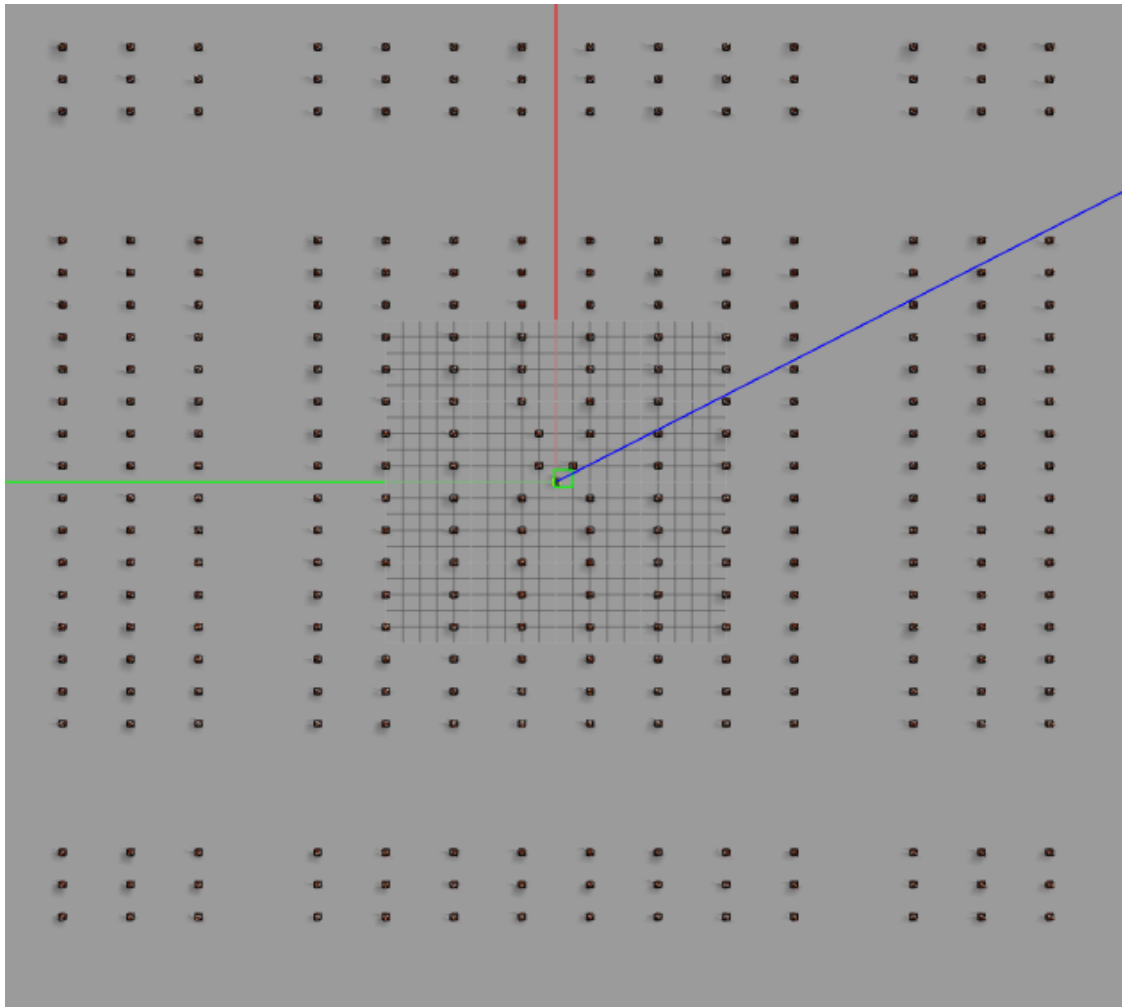




Experiment #12 : hand driven

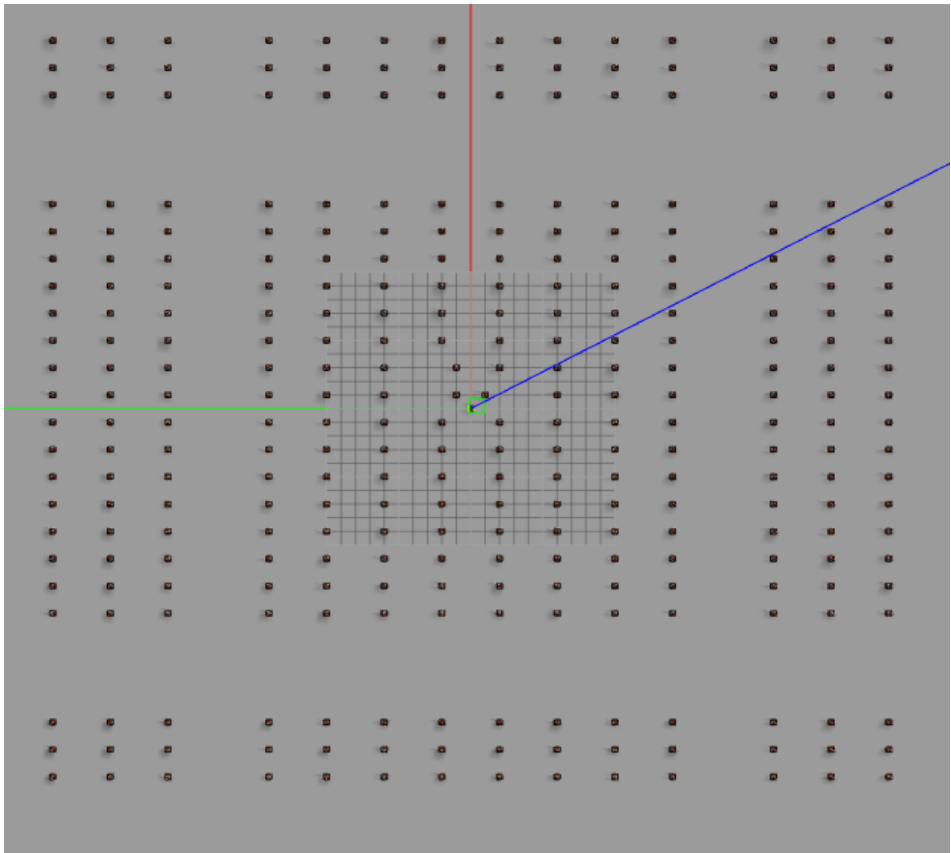
- **Wold name**: cone_orcahrd6e
- **World Description:** 3 cones are 1m ahead, on both sides near the origin.
- **Special notes**: Initial injected pose (8,8). 30000 particles tested .
- **World map & image:**





Experiment #12 : hand driven

- **Wold name**: cone_orcahrd6e
- **World Description:** 3 cones are 1m ahead, on both sides near the origin.
- **Special notes**: I tried to repeat experiment #7 , but this time with teleop-twist-keyboard commands instead of the programmed route. I tried to travel with the jacakl around the origin, where the “out-of order” cones are most visible, in order to get better results than the null results of the original experiment.
- Results- 0/5 (0%) success. I stopped there.



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 constant diameter, 0.2m.
- **Special notes**:

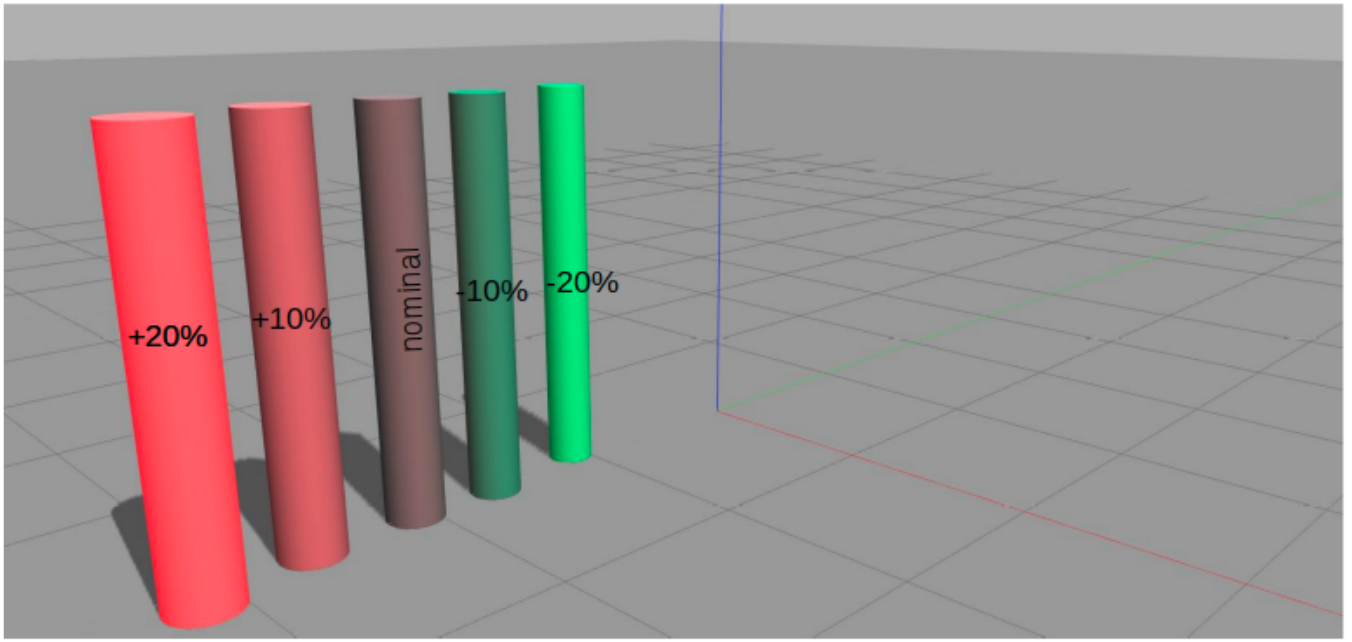
  The trees were simulated using 5 models of cylinders :

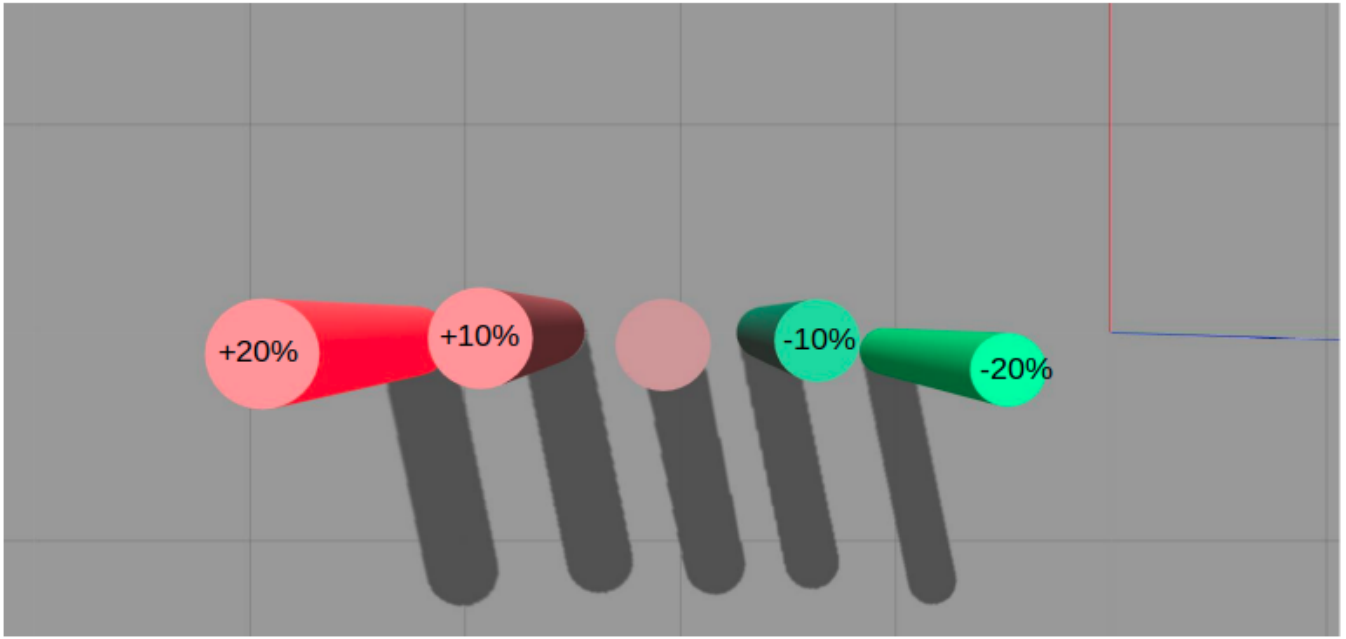
  - Nominal tree with a diameter of 0.28m
  - +- 10 % of the nominal diameter (0.308m , 0.252m)
  - +- 20 % of the nominal diameter (0.224m , 0.336m)
- Each model was colored in a different color in gazebo, in order for me to distinguish between them easily. The colors of do not influence the ground truth map 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

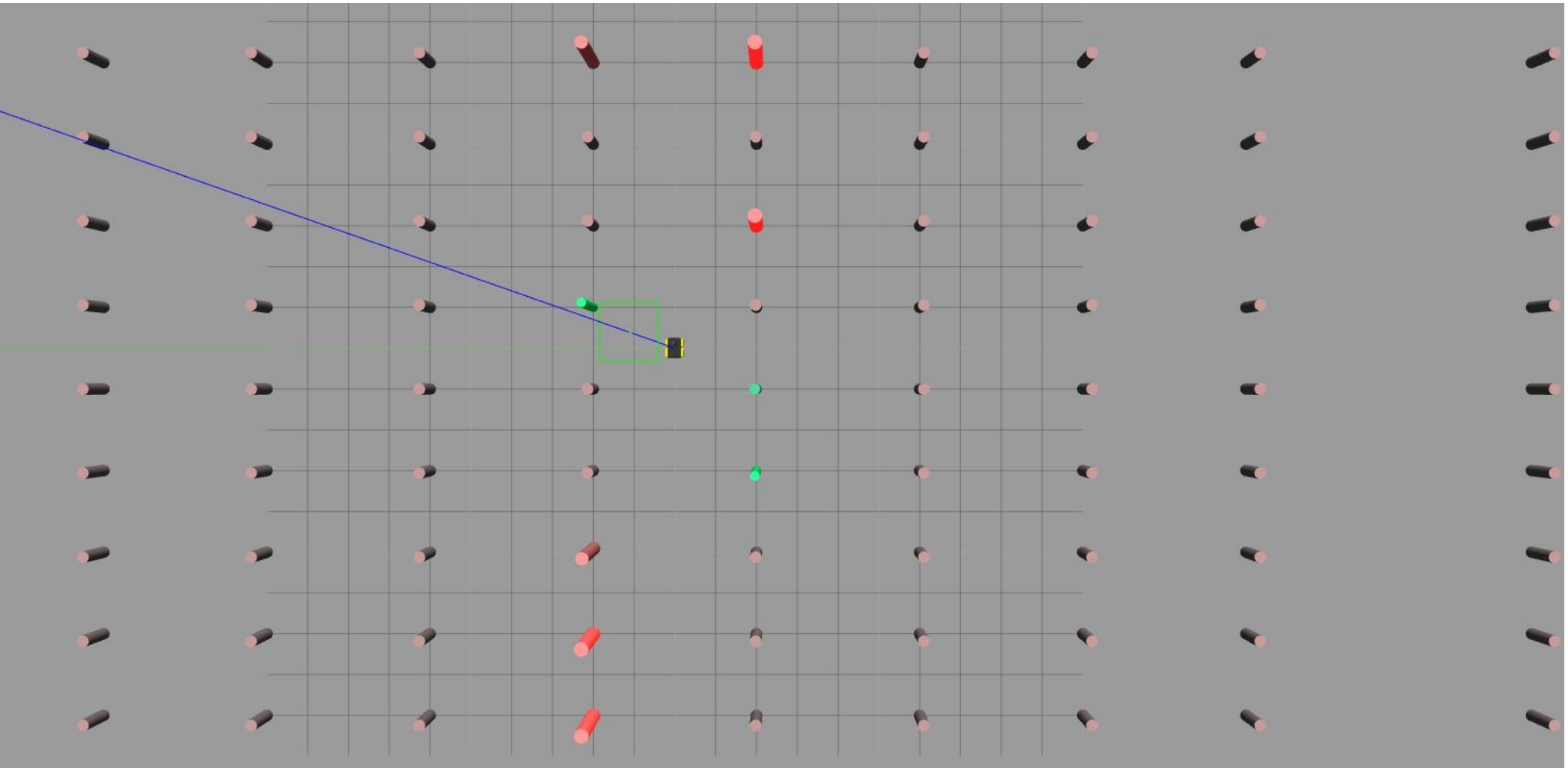
- The cylinders models:





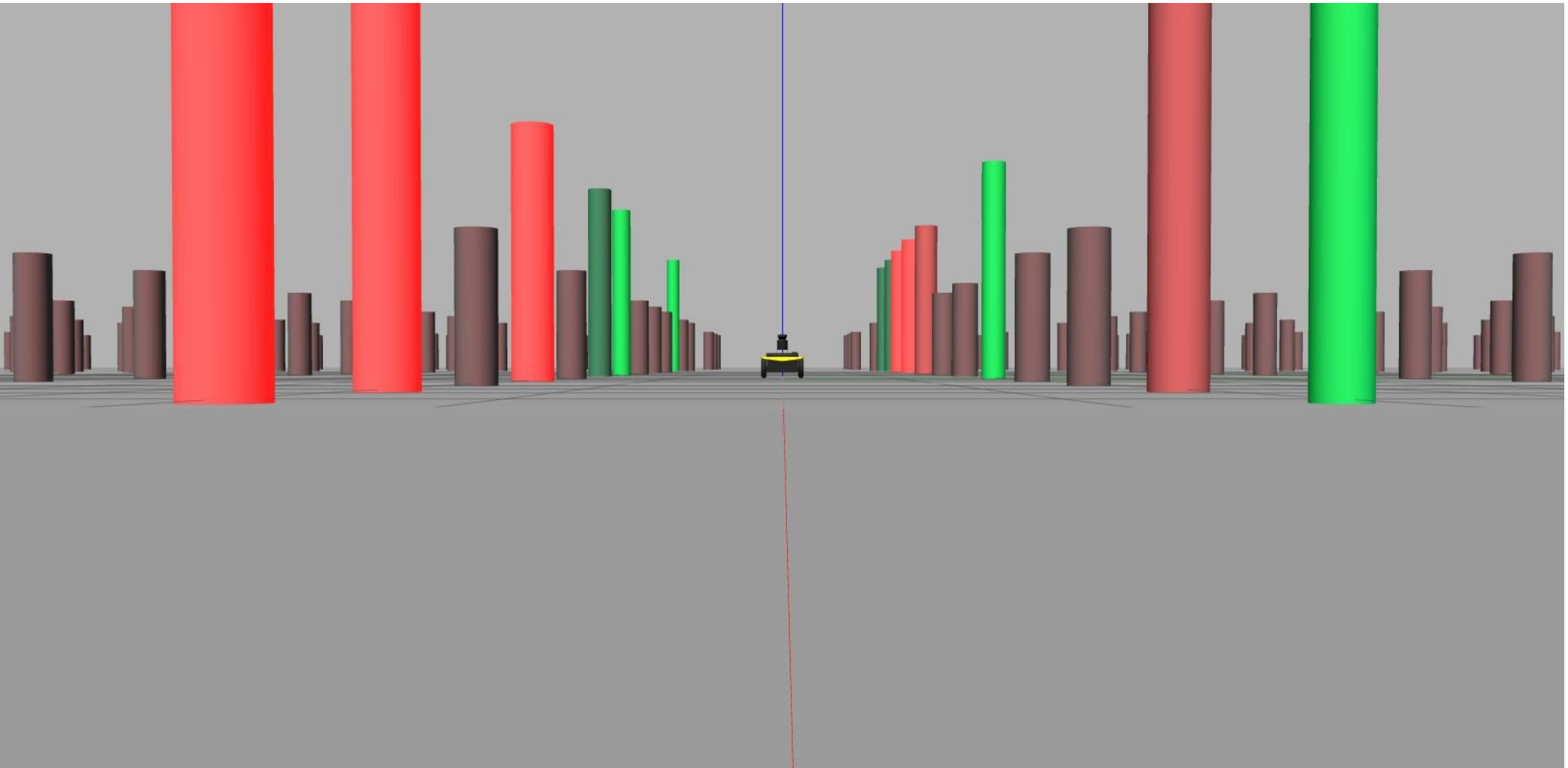
World of Cylinders

- A world example:



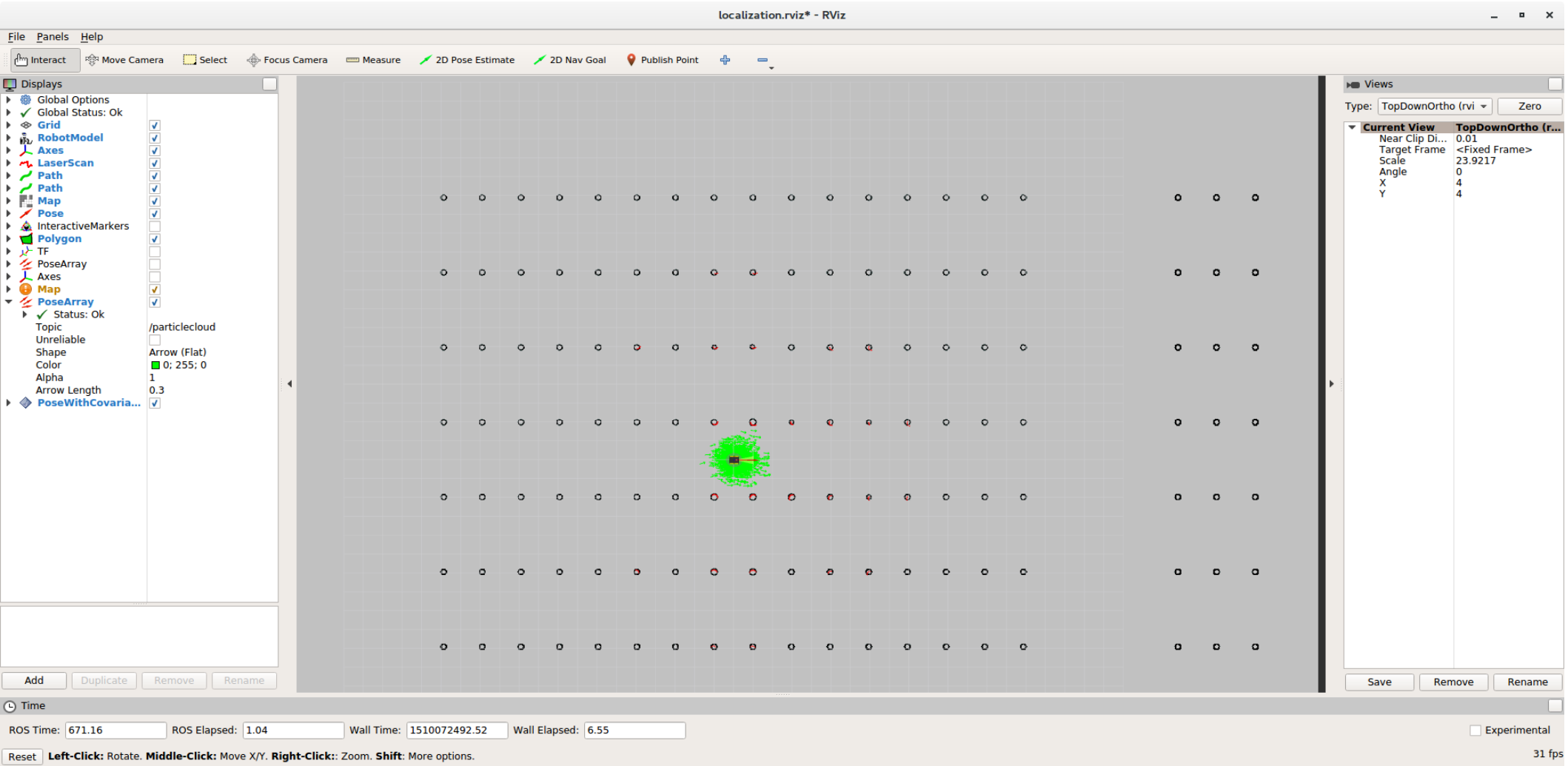
World of Cylinders

- 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.



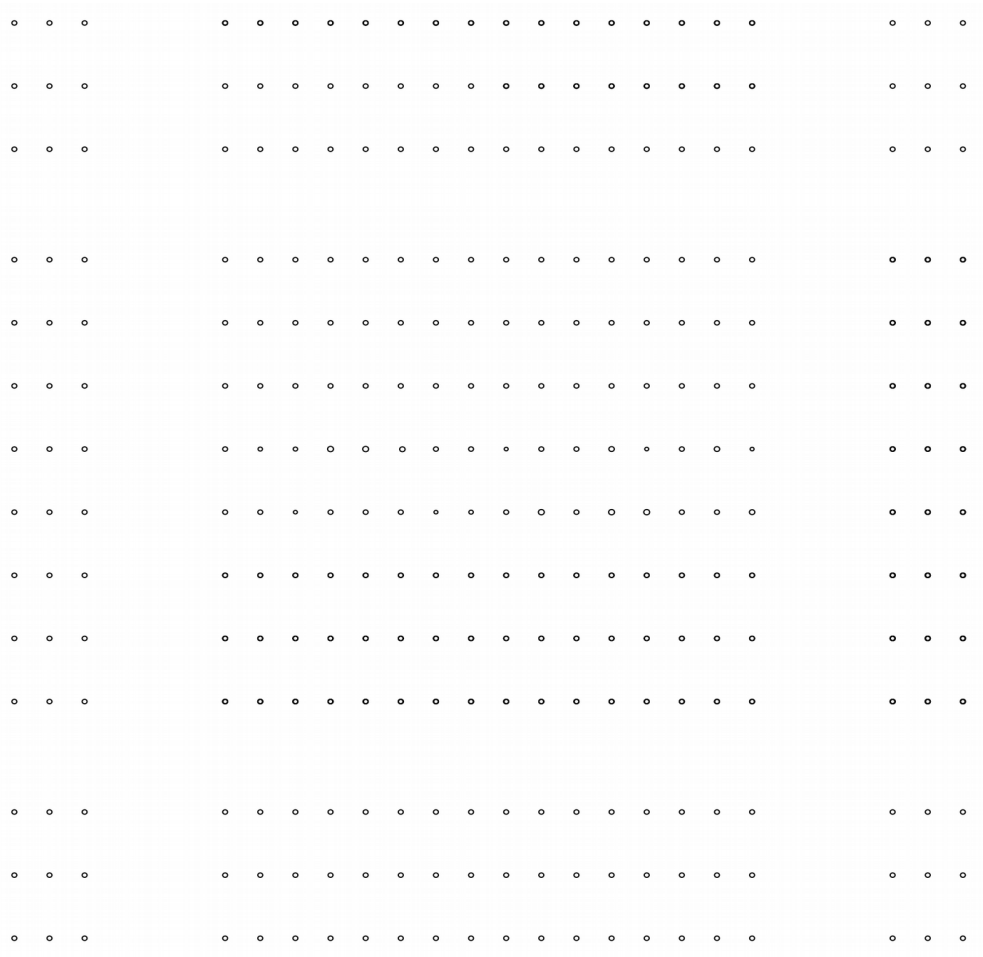
World of Cylinders

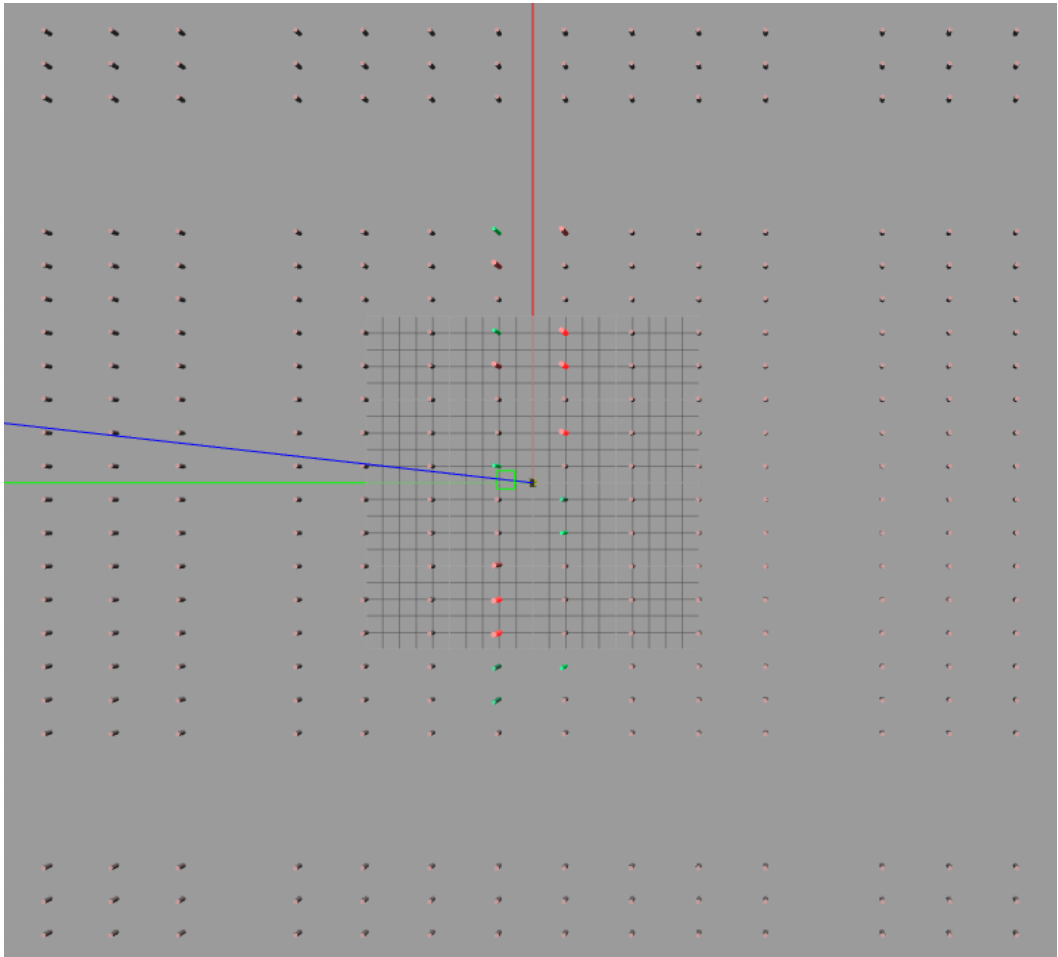
- A Rviz window with map example. One can see the various diameters of the cylinders.



Experiment #1A : World of cylinders #1

- **Wold name**: cone_orcahrd8b
- **World Description:** mixture of cylinders, on both sides near the origin. No placement variance.
- **Special notes**: Initial injected pose (8,8). 30000 particles tested
- **World map & image:**



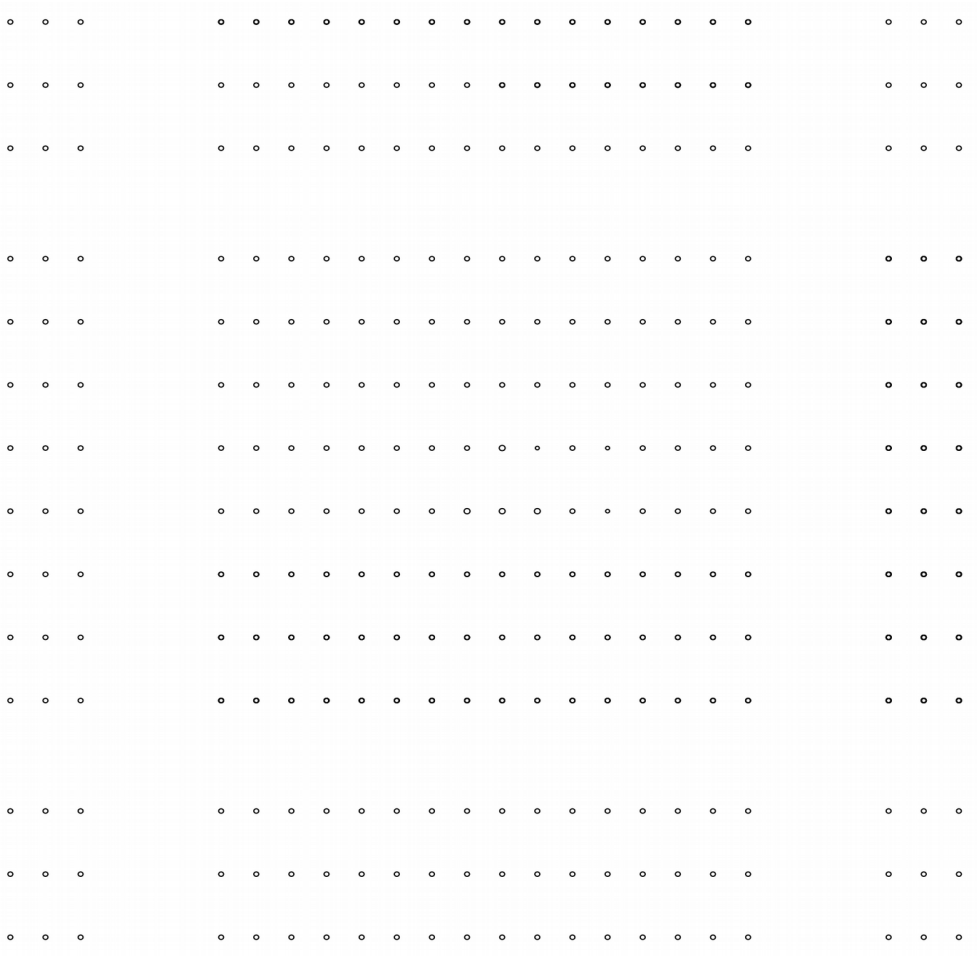


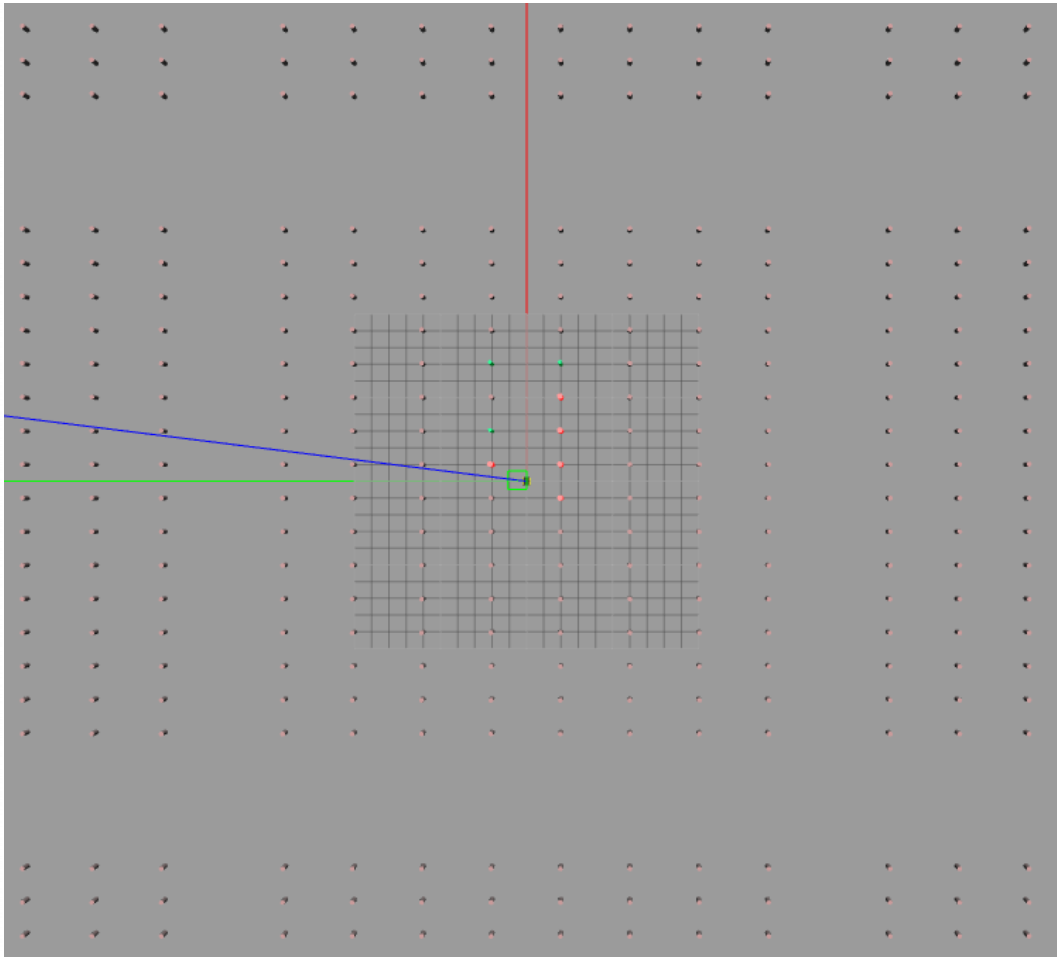
Experiment #1A : World of cylinders #1

- 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 #2A : World of cylinders #2

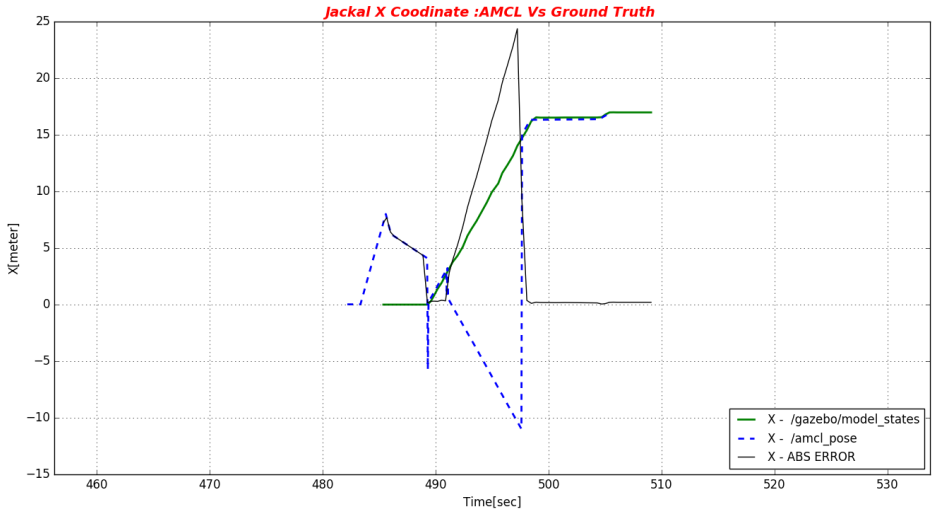
- **Wold name**: cone_orcahrd8e
- **World Description:** mixture of cylinders, on both sides near the origin. No placement variance.
- **Special notes**: Initial injected pose (8,8). 30000/4000 particles tested
- **World map & image:**





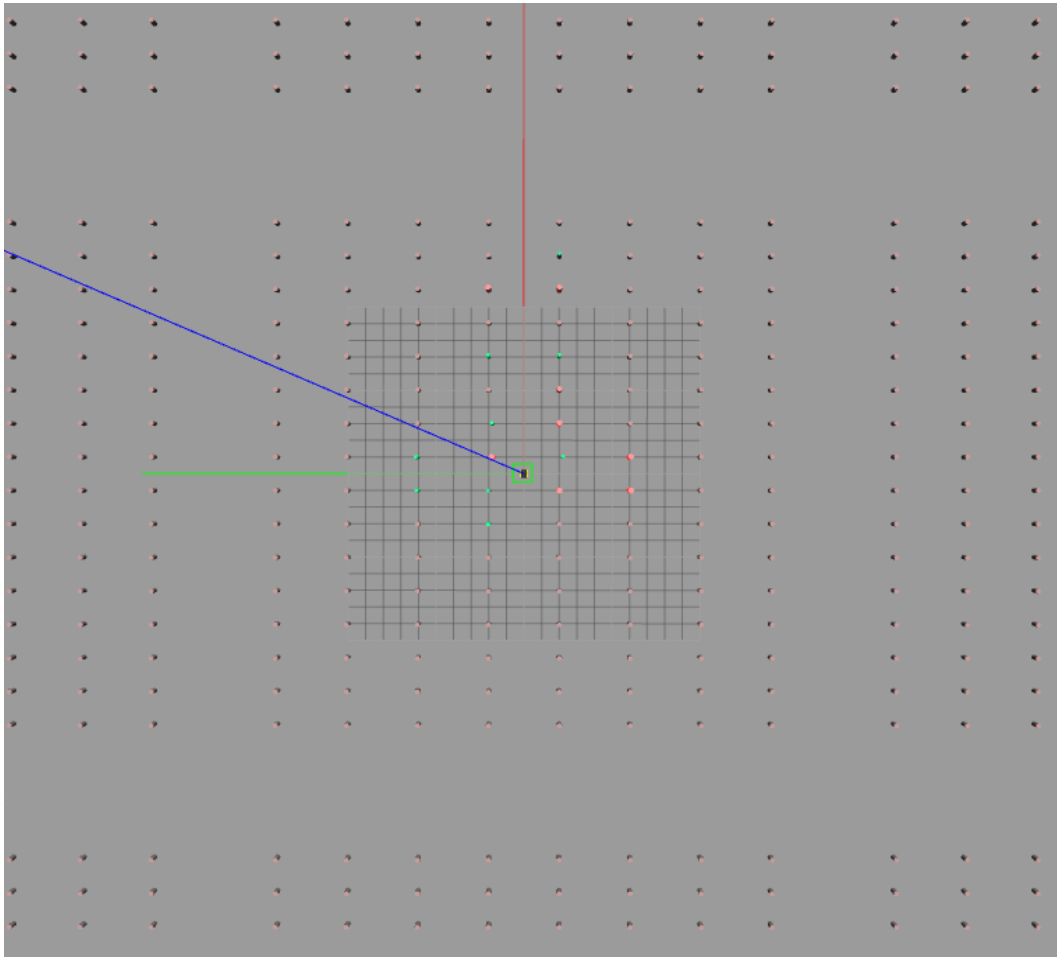
Experiment #2A : World of cylinders #2

- 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)



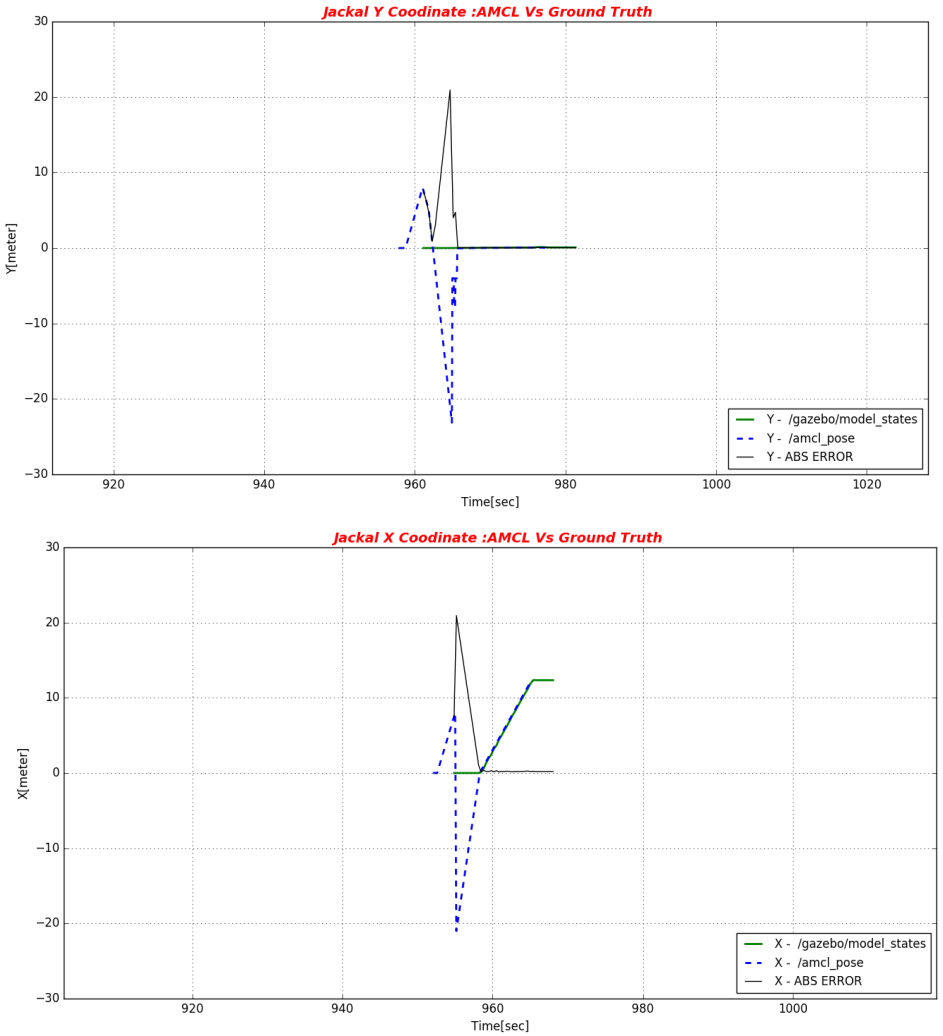
Experiment #3A : World of cylinders #3

- **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**: Initial injected pose (8,8). 30000/4000 particles tested
- **World map & image:**



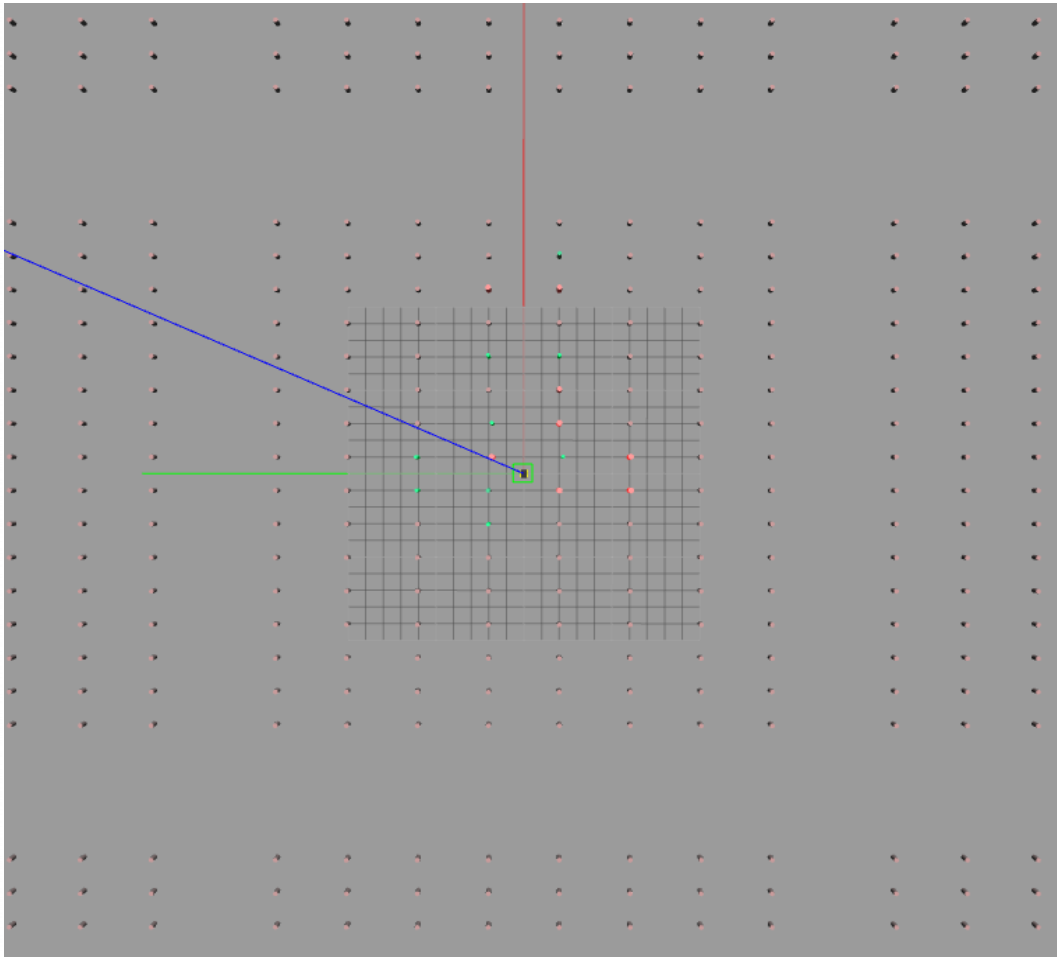
Experiment #3A : World of cylinders #3

- 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



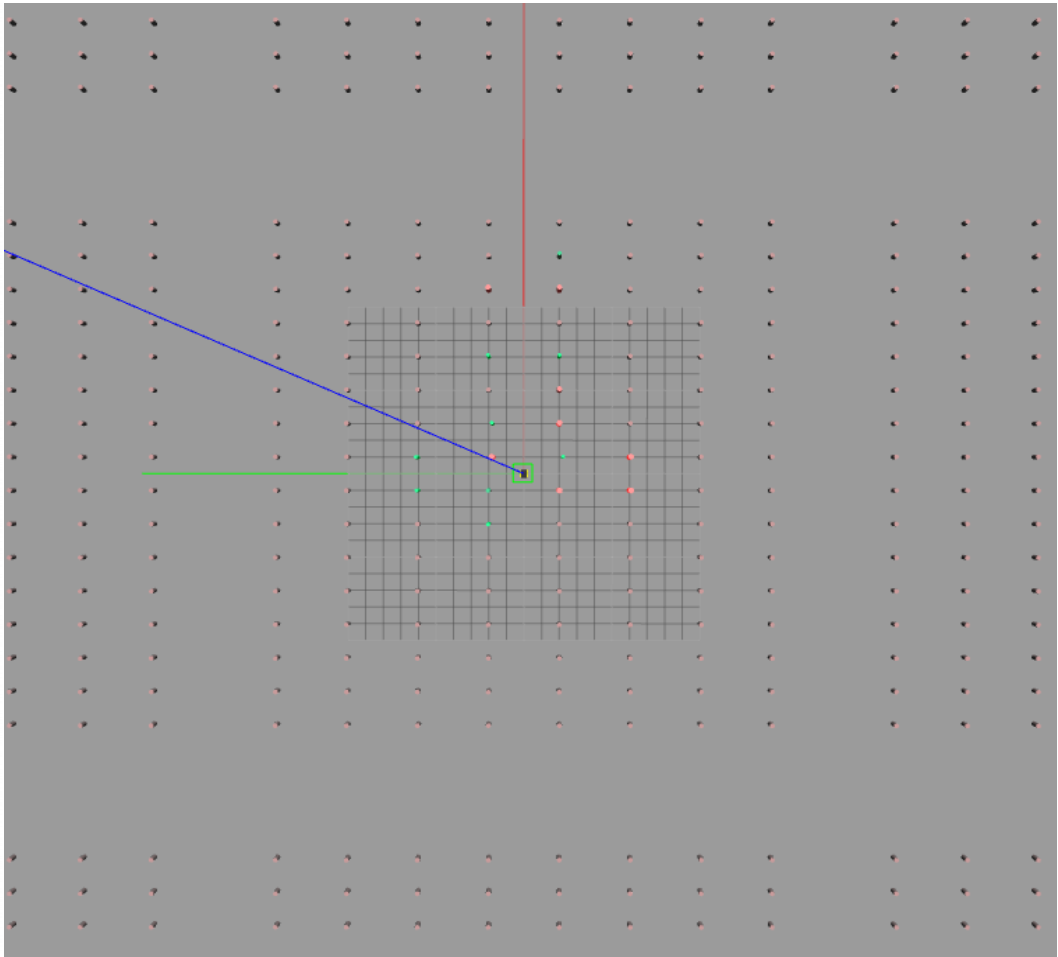
Experiment #4A : World of cylinders #4

- **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**: Initial injected pose (8,8). 30000/4000 particles tested **-injection of initial pose effect tested.**
- **World map & image:**



Experiment #4A : World of cylinders #4-Initial pose injection test

- **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**: Initial injected pose (8,8). 30000/4000 particles tested **-<u>injection of initial pose effect tested.</u>**
- **World map & image:**

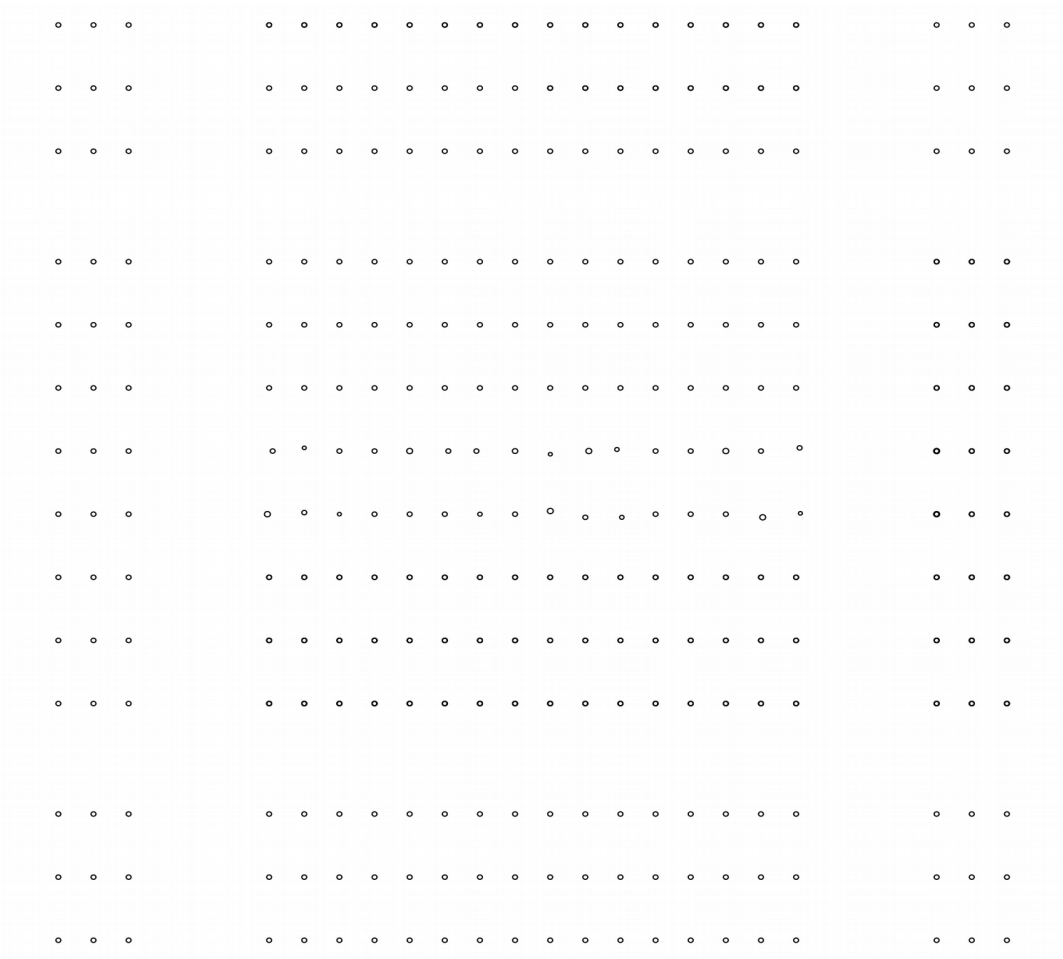


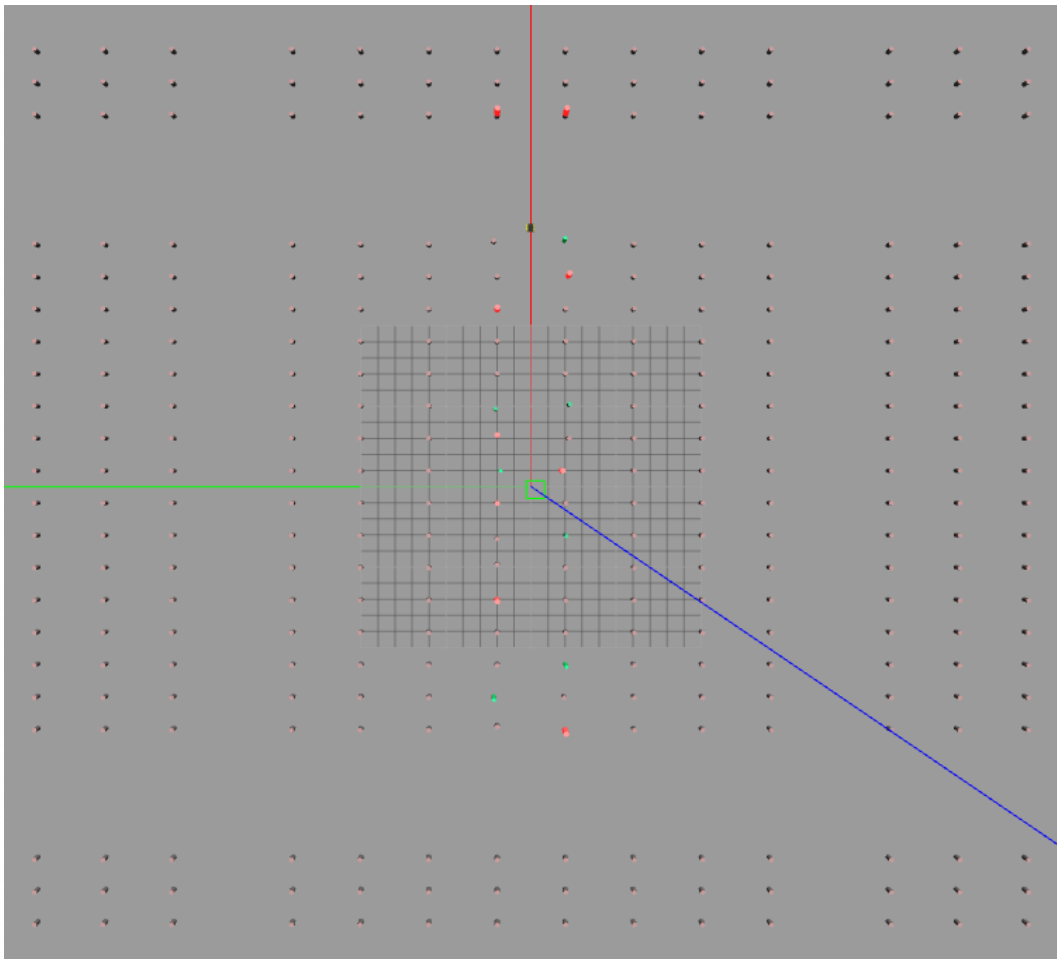
Experiment #4A : World of cylinders #4-Initial pose injection test

- Results:
  - **No initial pose injected**: 5 successive simulations with 30,000 particles – 0/5 (0%) success rate(in both axis) **.**
  - **Injection of initial pose (0,0)** : 5 successive simulations with 30,000 particles – 2/5 (40%) success rate in finding the true Y coordinate only. Nearly identical to the result experiment 3A with initial pose (8,8)-4/8 (50%)
  - **Injection of initial pose (0,0)** : 5 successive simulations with 40,000 particles – 4/5 (80%) success rate in finding the true X coordinate only. Nearly identical to the result experiment 3A with initial pose (8,8)-6/8 (75%)

Experiment #5A : World of cylinders #5-Initial pose injection test

- **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 on the rows to the left/right from the origin.
- **Special notes**: Initial injected pose (8,8). 30000/4000 particles tested -This experiment is similar to 4A **The purpose was just simulating another world.**
- **World map & image:**





Experiment #5A : World of cylinders #5-Initial pose injection test

- Results:
  - **Injection of initial pose (8,8) :**5 successive simulations with 30,000 particles – 0/5 (0%) success rate with regular initial pose (8,8).
  - **Injection of initial pose (8,8) :**5 successive simulations with 40,000 particles – 0/5 (0%) success rate with regular initial pose (8,8).
  - **Injection of initial pose (0,0)** : 5 successive simulations with 30,000 particles – 0/5 (0%) success rate.
  - **Injection of initial pose (0,0)** : 4 successive simulations with 40,000 particles – 0/4 (0%) success rate. I stopped there.

Conclusions and insights

- The first thing which appears immediately , even before analyzing the results in detail, is that 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 and sometimes to erroneous results. This may happen 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.
- **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

- **A single cone/two cones were never enough**. No matter whether it's a missing cone, a cone out-of place , or a cylinder instead of a cone (which may simulate a much larger cone)- the results of the AMCL where very poor. **The algorithm needs more heterogeneity than a single/two cones. This holds for any number of particles tested. The landmark should be visible for some minimum time during the convergence period .**
- **A landmark should be a distinguishable one. Positive landmarks are better than negative ones.** A missing cone or cones are usually not as good landmark as a cylinder. The experiment of 3 missing cones lead to poorer results than 3 cylinders placed in the same positions.
- **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 9 yielded 0% success. Even with an entire "random row" (experiment 10) the results were quite poor (40%), despite the large heterogeneity.
- **3 cones may introduce a larger heterogeneity than an entire row of cones.** When an entire row of cones was missing/out of place - this is indeed a distinguishable landmark which is seen for a long period of time. But still, If there are two valid solutions (because of symmetry) surprisingly this situation sometimes lead to a totally wrong one. The amount of heterogeneity is not just counting how many cones are "out of order" but how strongly they force a single, valid solution.
- **The algorithm does not cope very well with 2 symmetrical valid solutions, nor it does so with ambiguity in a single axis**. The "missing row" and "out of order row" experiments (#3 & #6) the results were not very good, considering that it was a "relaxed", single- dimensional problem- (only a valid y coordinate should be found) . On the other hand, when the 3 cones are placed in such way that cause a strong homogeneity in the area – a valid solution was usually found.

Conclusions and insights

- **Heterogeneity on both sides is twice as better**. Experiments 10 and 11 shows that when the robot senses unique a pattern (i.e.- heterogeneity) on both sides the results of the localization are about twice as good. The “single random row” experiment introduced enough heterogeneity for a single solution of the localization algorithm, yet the results (as stated before) were quite poor (~40% success). Adding another random row on the opposite side
- **“Hand drive” doesn’t seem to help.** Experiment 12 (hand driven jackal) showed no improvement compared to the case of a deterministic pre programmed route. Moving the robot around randomly did not yield better results.
-

Conclusions and insights

- **“Natural” Heterogeneity in the cylinder’s diameter alone is not enough.** From experiments 1A and 2A 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 3A 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). Running the algorithm with 30,000 particles leaded to null results of experiments 1A and 2A but succeeded (to some extent) in experiment 3A. Yet, even with 40,000 particles (where my PC was sweating) – a valid location solution in both coordinateS was never found despite the larger heterogeneity introduced by moving the cylinders from their original positions. Experiment 5A is similar to 3A and 4A, but yielded null results. It shows that the even the partial success of experiment 3A 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.
- **Adding particles may compensate to a lack of heterogeneity.** Going up to 40,000 particles was a desperate step , but it improved the success rate of experiment 3A. As stated before ,It is still a little bit problematic to fit the number of particles to the world. Why 30,000 particles leaded to a good Y-coordinate solution but 40,000 particles leaded to a good X-coordinate solution is still a mystery to me.

Conclusions and insights

- **The Initial pose injection value does not effect the global localization of AMCL.** The results of experiments 4A and 5A shows clearly that injecting a far, “fake” initial pose is just as good as injecting the true initial position (0,0). This holds for 30,000 particles and 40,000 particles. This result should not surprise, since when calling the global localization service, the particle cloud, which was just around the robot’s area, immediately covers the entire map, as the algorithm seeks the robot’s location all around the map. This is the true meaning of “global”- the (assumed) initial pose/hypothesis is ignored, and the search is based on the map and the laser scan only. **On the other hand, It is not understood to me why no injection of initial pose at all does not yield the same results.**
- **One important note**- when I did not call the global localization service and used only local search, the initial pose & the covariance effected the result a lot , of course. But local search is not the scope of this work.