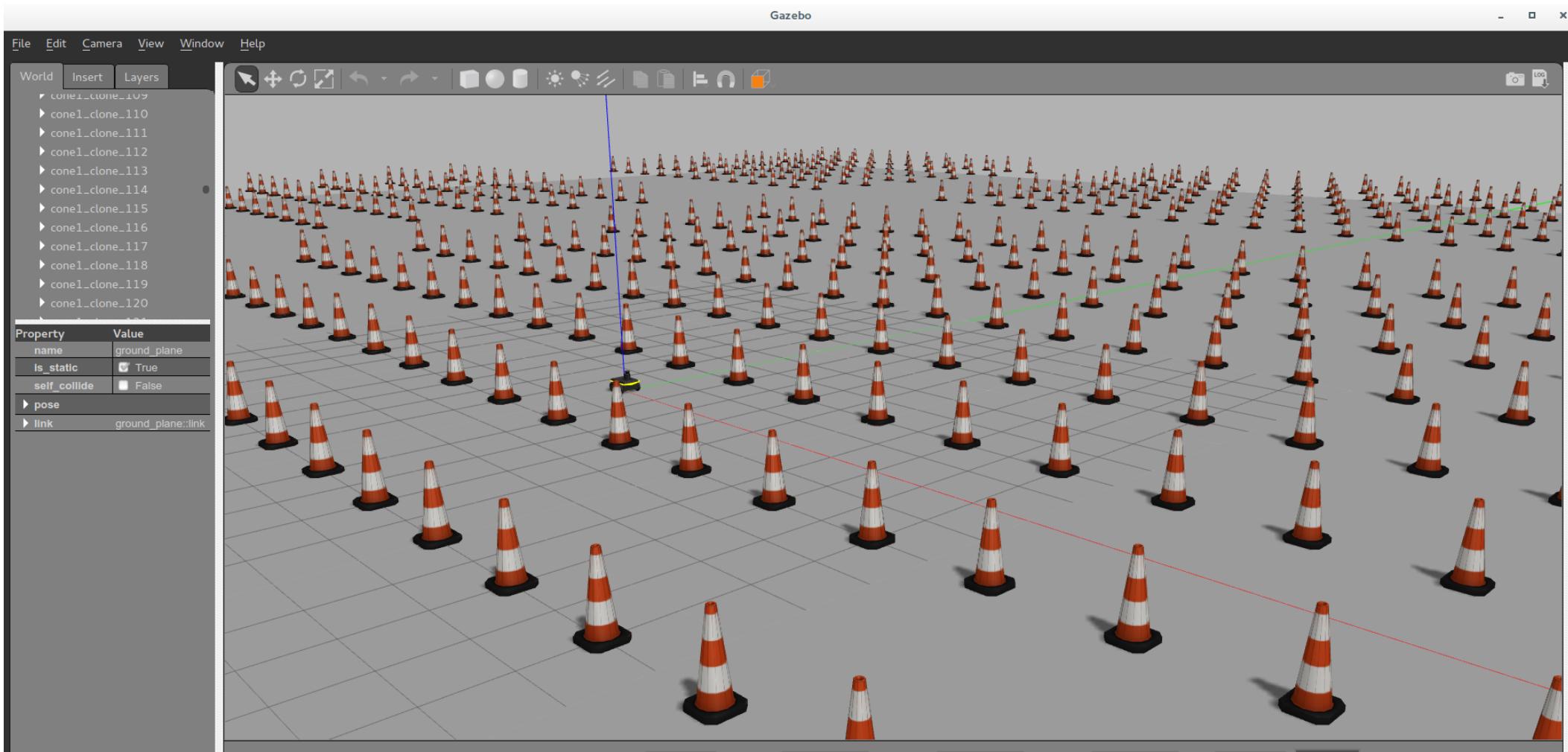


# The effect of orchard heterogeneity on navigating an autonomous vehicle for agricultural tasks

Submits: Liron Shenkar

Supervisors Prof. Amir Degani , Omer Shalev



## Project Structure and general info

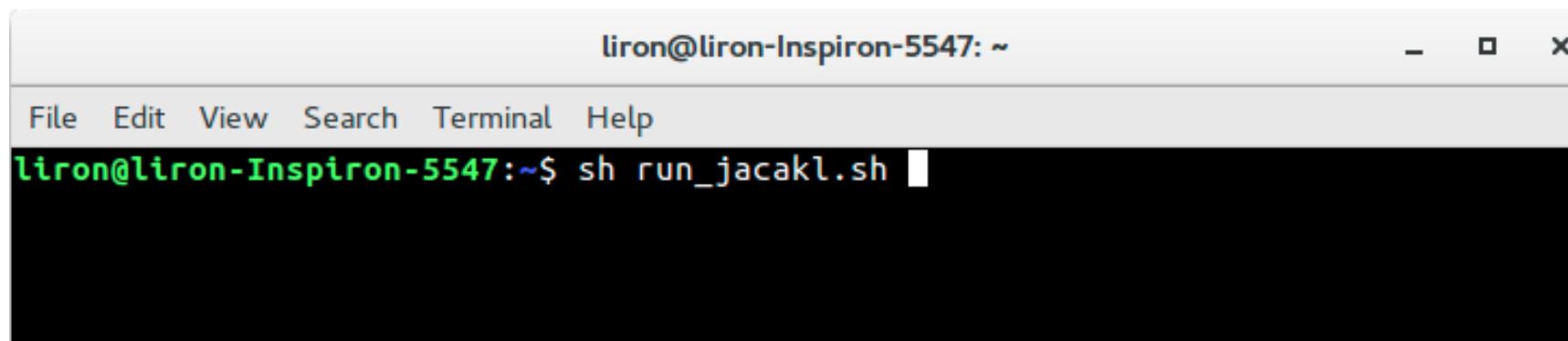
***Project Goal: To examine the performance of AMCL global localization algorithm under various heterogeneities of an orchard of cones.***

- 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

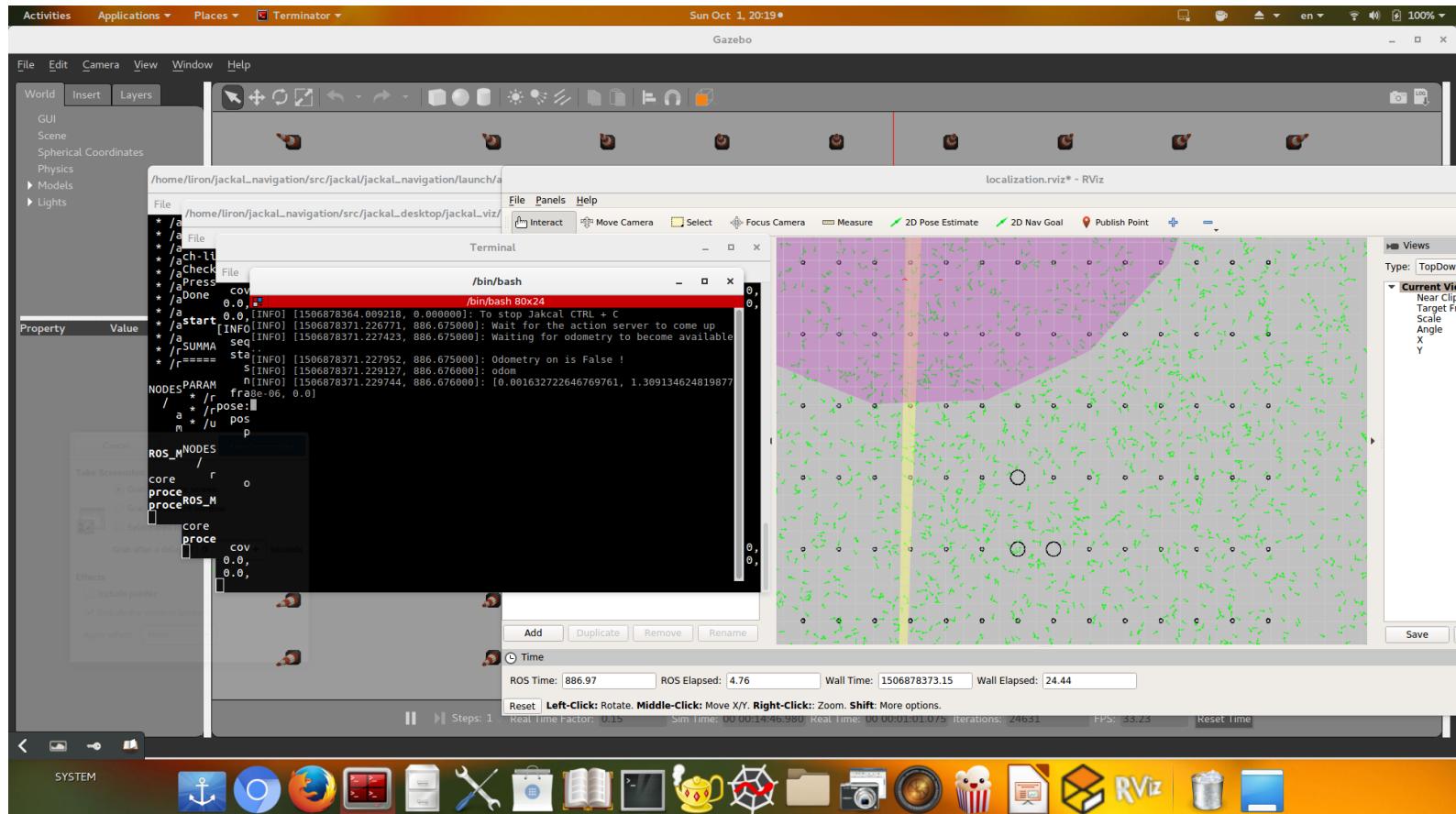
- All project files are in jackal\_navigation dir, inside linux user's home directory.
- Directory contents in based on jackal navigation stack from CLEARPATH® website  
<https://www.clearpathrobotics.com/assets/guides/jackal/simulation.html> downloaded from source
- Project is running on **UBUNTU 16 (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 scripts uses it.
  - Running example:



A screenshot of a terminal window titled "liron@liron-Inspiron-5547: ~". The window has standard Linux window controls (minimize, maximize, close) at the top right. The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal itself is black with white text. At the top of the terminal, the command "liron@liron-Inspiron-5547:~\$ sh run\_jacakl.sh" is visible, with the cursor positioned after the command. The rest of the terminal window is blank, showing only the terminal prompt and the command entered.

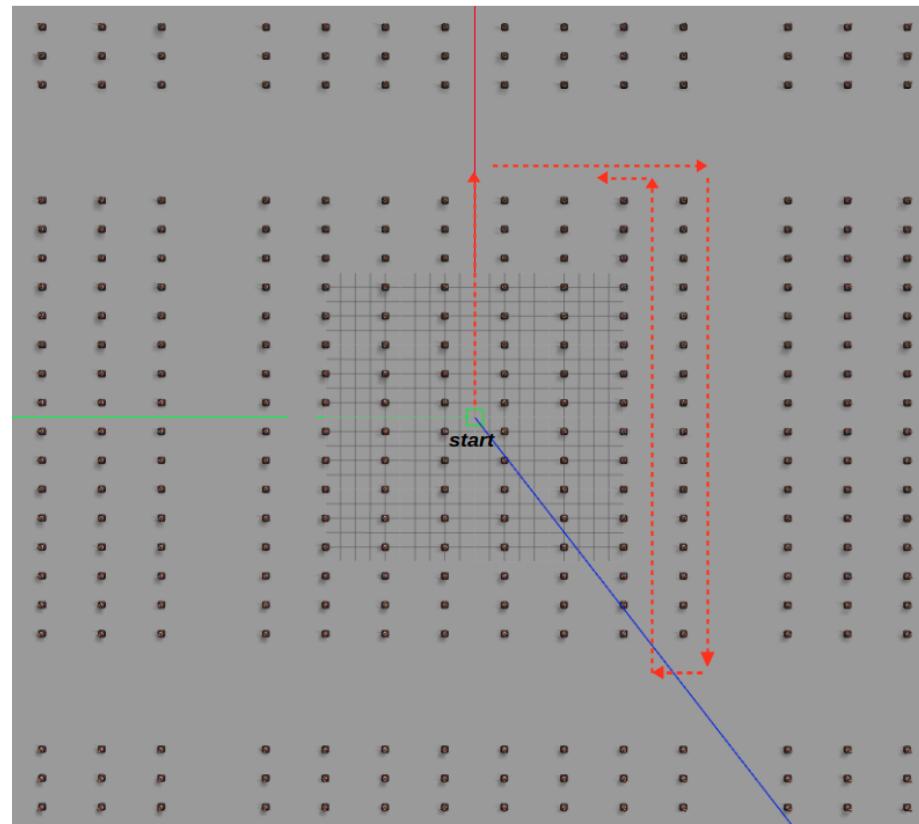
# Project Structure and general info

- 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.

The image shows two terminal windows side-by-side. The left window is titled '/bin/bash' and has a red header bar. It displays ROS log messages from the 'jackal' package. The right window is titled 'Terminal' and has a grey header bar. It displays sensor data, specifically a list of 16 values representing a covariance matrix.

**/bin/bash**

```
[INFO] [1506878364.009218, 0.000000]: To stop Jakcal CTRL + C  
[INFO] [1506878371.226771, 886.675000]: Wait for the action server to come up  
[INFO] [1506878371.227423, 886.675000]: Waiting for odometry to become available  
..  
[INFO] [1506878371.227952, 886.675000]: Odometry on is False !  
[INFO] [1506878371.229127, 886.676000]: odom  
[INFO] [1506878371.229744, 886.676000]: [0.001632722646769761, 1.309134624819877  
8e-06, 0.0]  
[INFO] [1506878377.928148, 887.676000]: Odometry on is True !
```

**Terminal**

```
0.0, 0.0, 0.0, 0.0, 0.0, 0.25, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  
0.0, 0.06853891945200942]  
, 883.655000]: header:
```

x: 0.0  
y: 0.0  
z: 0.0267568523876  
w: 0.999641971333  
covariance: [0.5, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.25, 0.0, 0.0, 0.0, 0.0, 0.0,  
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  
0.0, 0.0, 0.0, 0.0, 0.0, 0.06853891945200942]

# Project Structure and info

- run\_jackal.sh file structure:

Killing current Process

Choosing world name

Create map flag

Injecting fake Initial pose

```
#!/bin/bash
killall gzserver
killall -9 gazebo
sleep 8s

#mapname=cone_orcahrd4_single_cone_dz2
#mapname=cone_orcahrd4_single_cone_dy
#mapname=cone_orcahrd4a
#mapname=cone_orcahrd4_interleaving_dy3
#mapname=cone_orcahrd4_unique_pattern_cones_dy
#mapname=cone_orchard4_one_rect
mapname=cone_orcahrd5

export JACKAL LASER=1
export JACKAL LASER_SCAN_TOPIC=front/scan

##create bag
#####
create_bag=false
#create_bag=false

##create map
#####
#create_map=true
create_map=false

##headless
#####
###see http://gazebosim.org/tutorials?tut=ros_roslaunch
###see http://gazebosim.org/tutorials?tut=quick_start
#headless=true
headless=false

##inject pose
#####
#inject_initial_pose=True
inject_initial_pose=False
xinitial=8.0 #initial x for AMCL
yinitial=8.0 #initial y for AMCL
```

# Project Structure and info

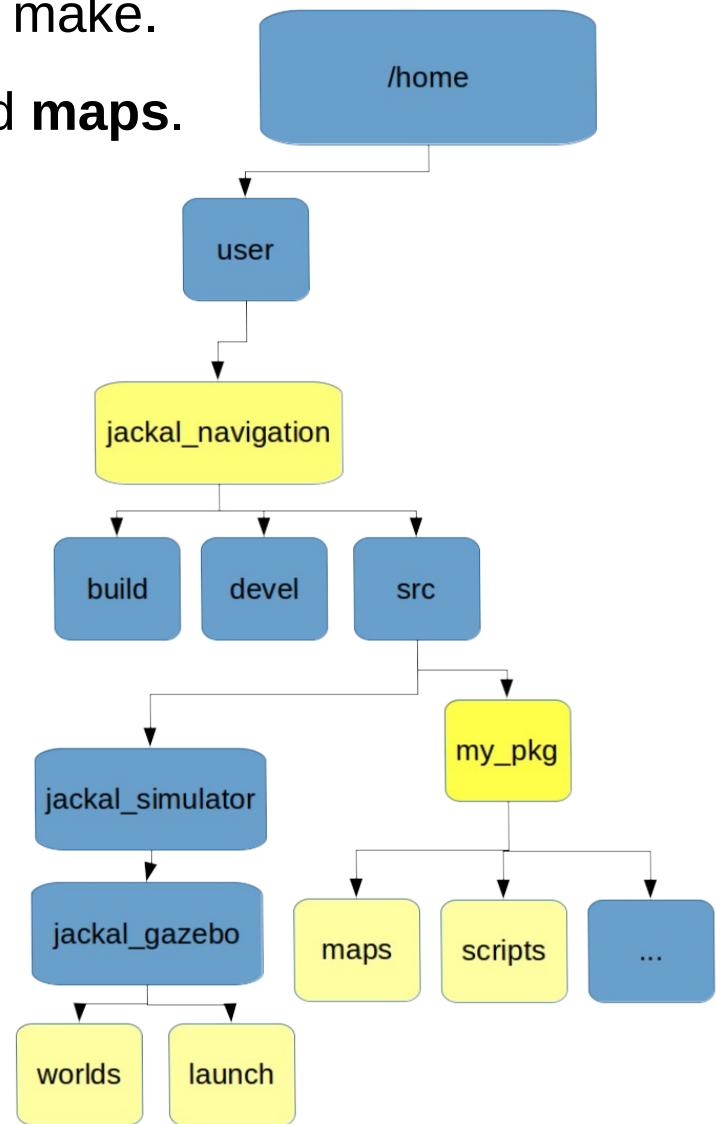
- run\_jackal.sh file structure (Cont.):

Launching the world  
Creating a map  
Launching amcl/Rviz  
Running Scripts

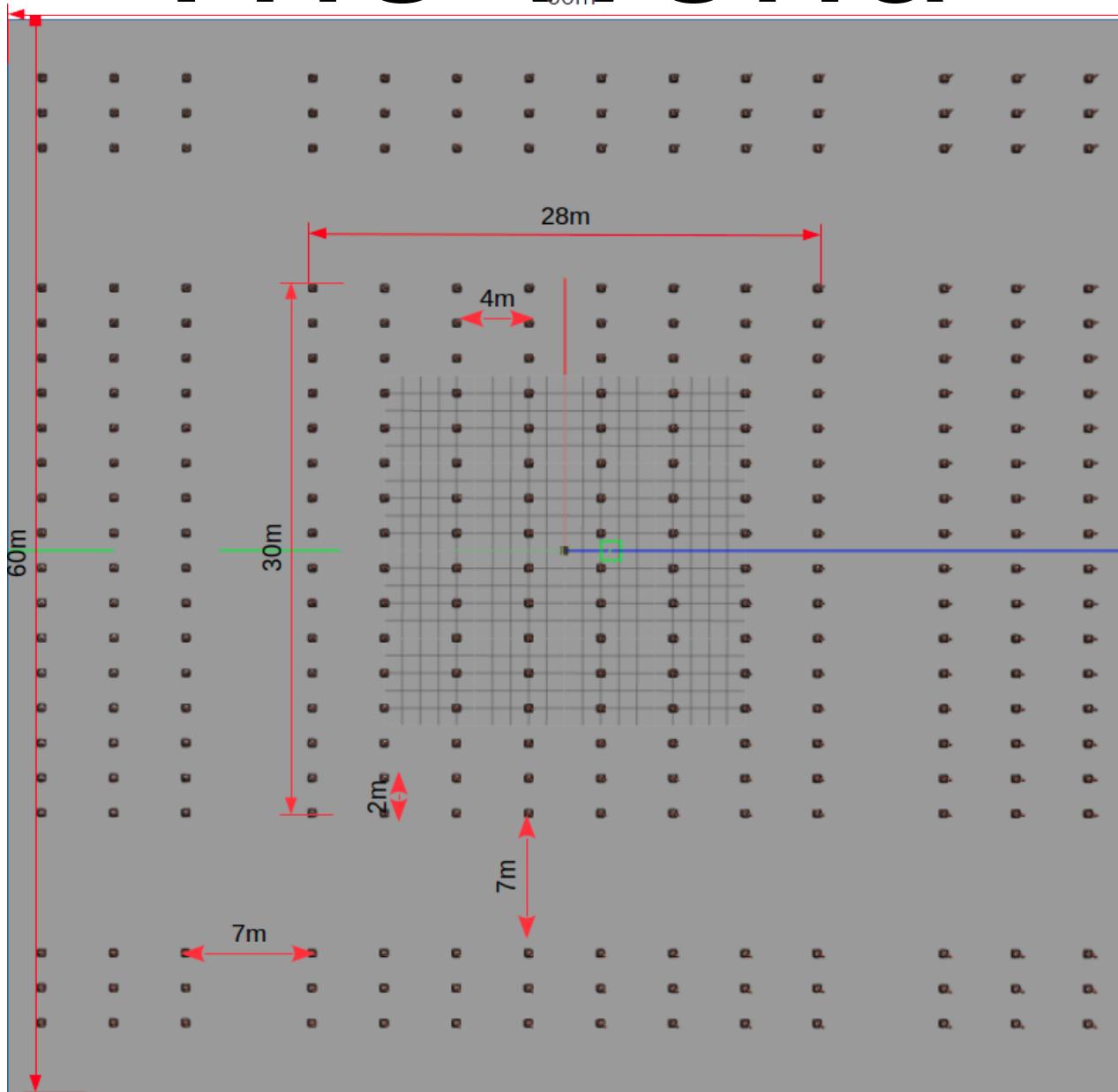
```
amcl.launch ✘ run_jacakl.sh ✘ cone_orcahrd5.world ✘ cone_orcahrd5.launch ✘ cone_orcahrd5.yaml ✘ cone_orcahrd4.world ✘ cone_...  
70 else  
71 #gnome-terminal -e "roslaunch gazebo_ros empty_world.launch"&  
72  
73 gnome-terminal -e "roslaunch jackal_gazebo $mapname.launch gui:=true headless:=false"&  
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 sleep 5s  
102  
103 gnome-terminal -e "rosrun my_pkg doitall_node2.py $xinitial $yinitial $inject_initial_pose"&  
104 sleep 6s  
105  
106  
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"&  
113
```

# 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.

Name
cone_orcahrd1
cone_orcahrd2
cone_orcahrd3
cone_orcahrd4
cone_orcahrd4a
cone_orcahrd4b
cone_orcahrd4_interleaving_dy
cone_orcahrd4_interleaving_dy3

Name
Jackal X Coordinate AMCL Vs Ground Truth .png
Jackal X Coordinate Vs Time.png
Jackal X Coordinate Vs Time AMCL.png
Jackal X Y Coordinate AMCL Vs Ground Truth.png
Jackal Y Coordinate AMCL Vs Ground Truth .png
Jackal Y Coordinate Vs Time.png
Jackal Y Coordinate Vs Time AMCL.png

# 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)

Name
cone_orcahrd1
cone_orcahrd2
cone_orcahrd3
cone_orcahrd4
cone_orcahrd4a
cone_orcahrd4b
cone_orcahrd4_interleaving_dy
cone_orcahrd4_interleaving_dy3

Name
Jackal X Coordinate AMCL Vs Ground Truth .png
Jackal X Coordinate Vs Time.png
Jackal X Coordinate Vs Time AMCL.png
Jackal X Y Coordinate AMCL Vs Ground Truth.png
Jackal Y Coordinate AMCL Vs Ground Truth .png
Jackal Y Coordinate Vs Time.png
Jackal Y Coordinate Vs Time AMCL.png

# 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.

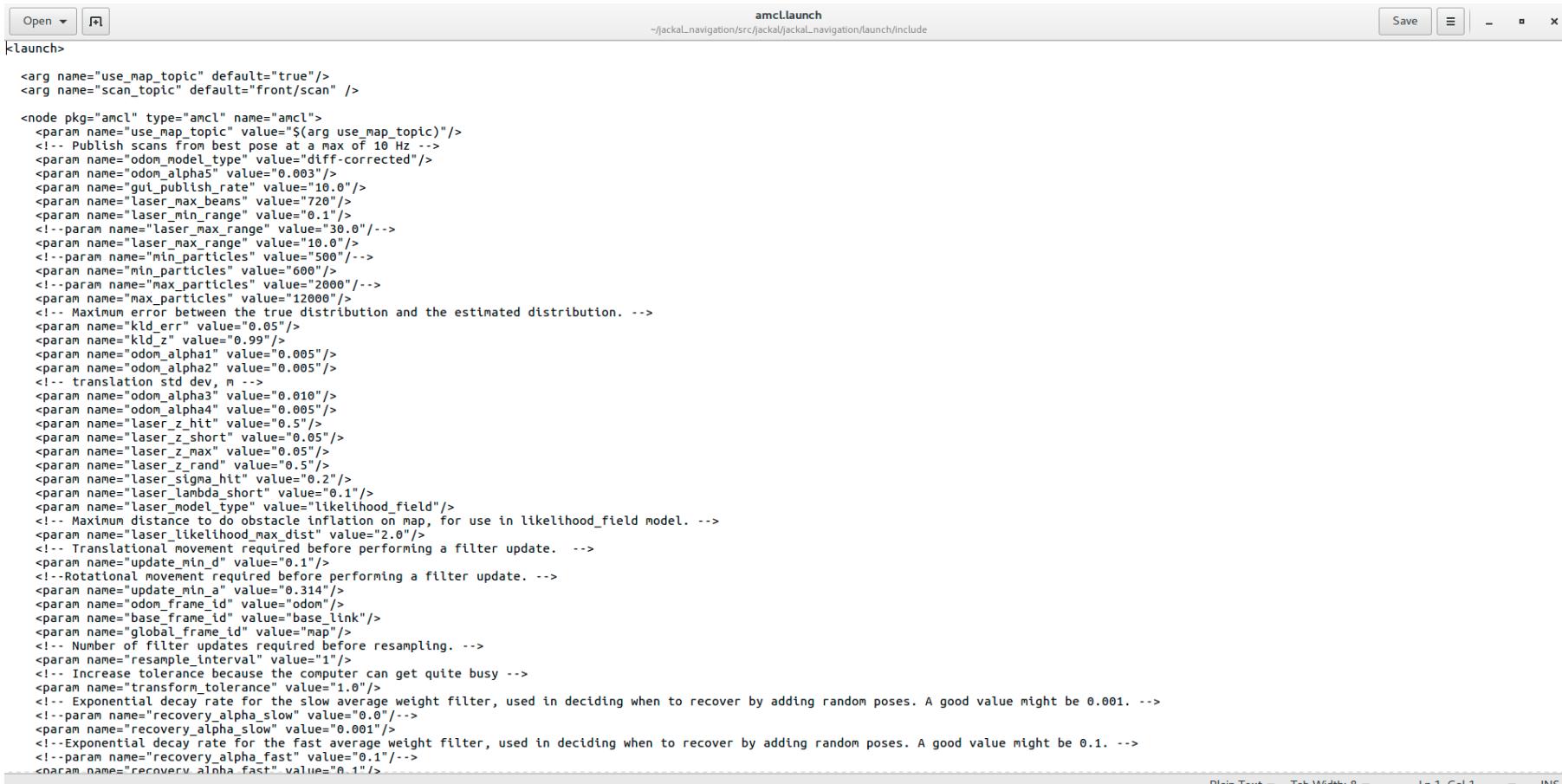
Name
cone_orcahrd5.launch
cone_orcahrd5.pgm
cone_orcahrd5.png
cone_orcahrd5.txt
cone_orcahrd5.world
cone_orcahrd5.yaml
jackal.launch
jackal_amcl_coordinates.txt
jackal_ground_truth_coordinates.txt

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/>



The screenshot shows a code editor window with the title "amcl.launch" and the path " ~/jackal\_navigation/src/jackal\_navigation/launch/include". The code is XML configuration for the AMCL node. It includes parameters for map topics, laser topics, and various AMCL parameters like particle filters, sensor models, and frame IDs. The code is heavily annotated with comments explaining the purpose of each parameter.

```
<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"/>
```

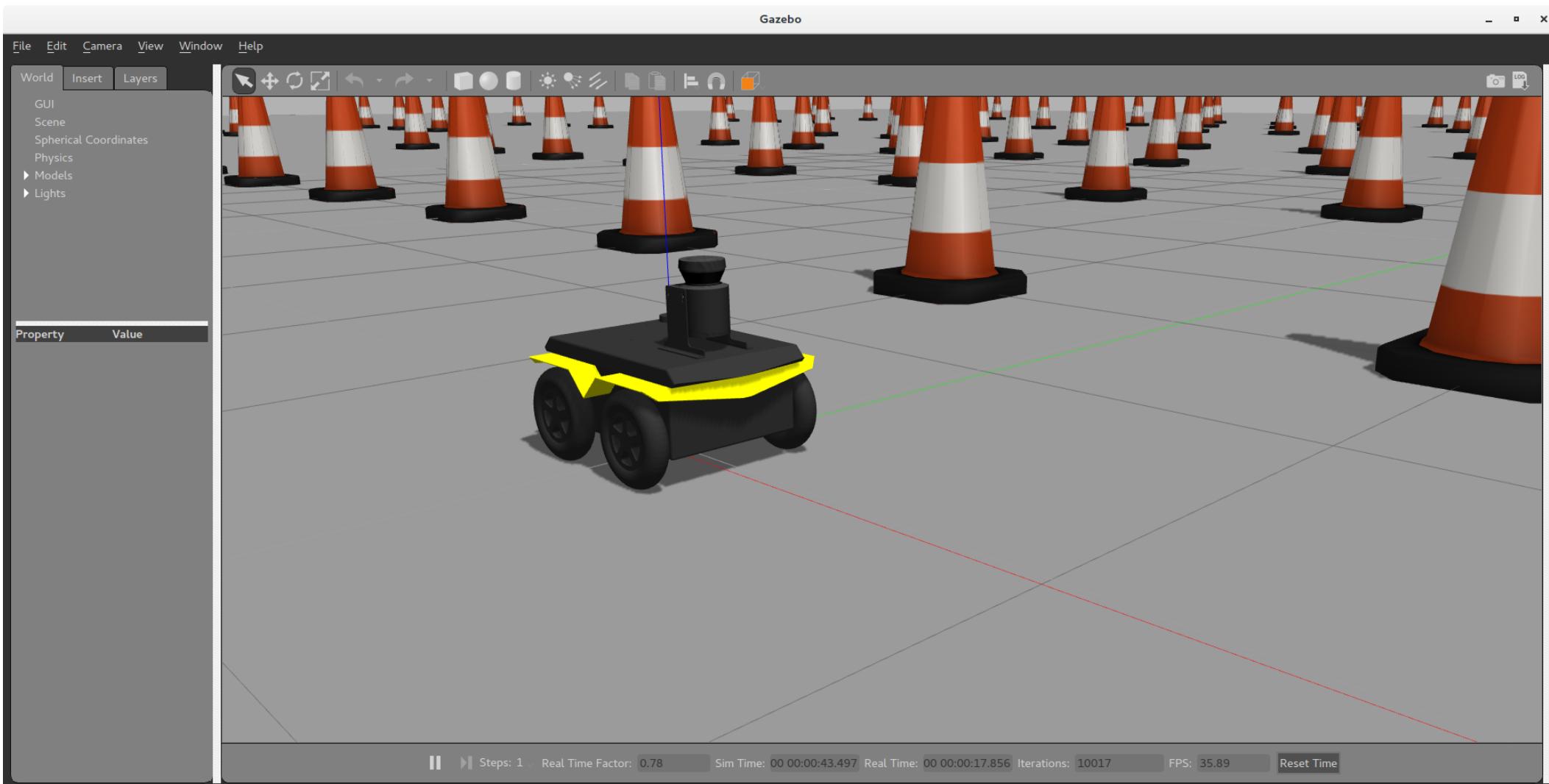
# Project Structure and info

## ***Project location***

- *Github : [https://github.com/sliri/my\\_pkg.git](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.

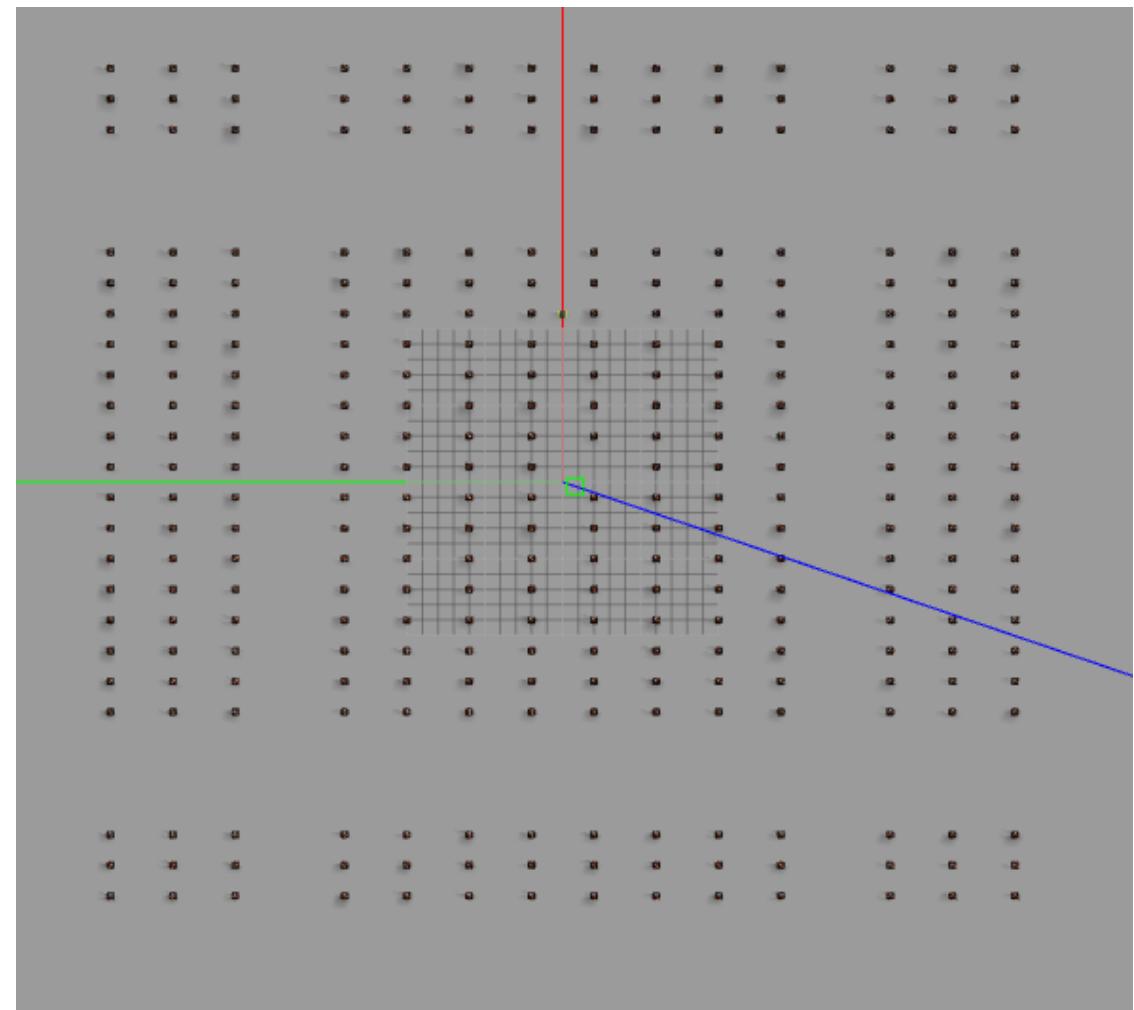
# Experiments

# Experiments



# Experiment #1 : single cone missing

- **World 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:**

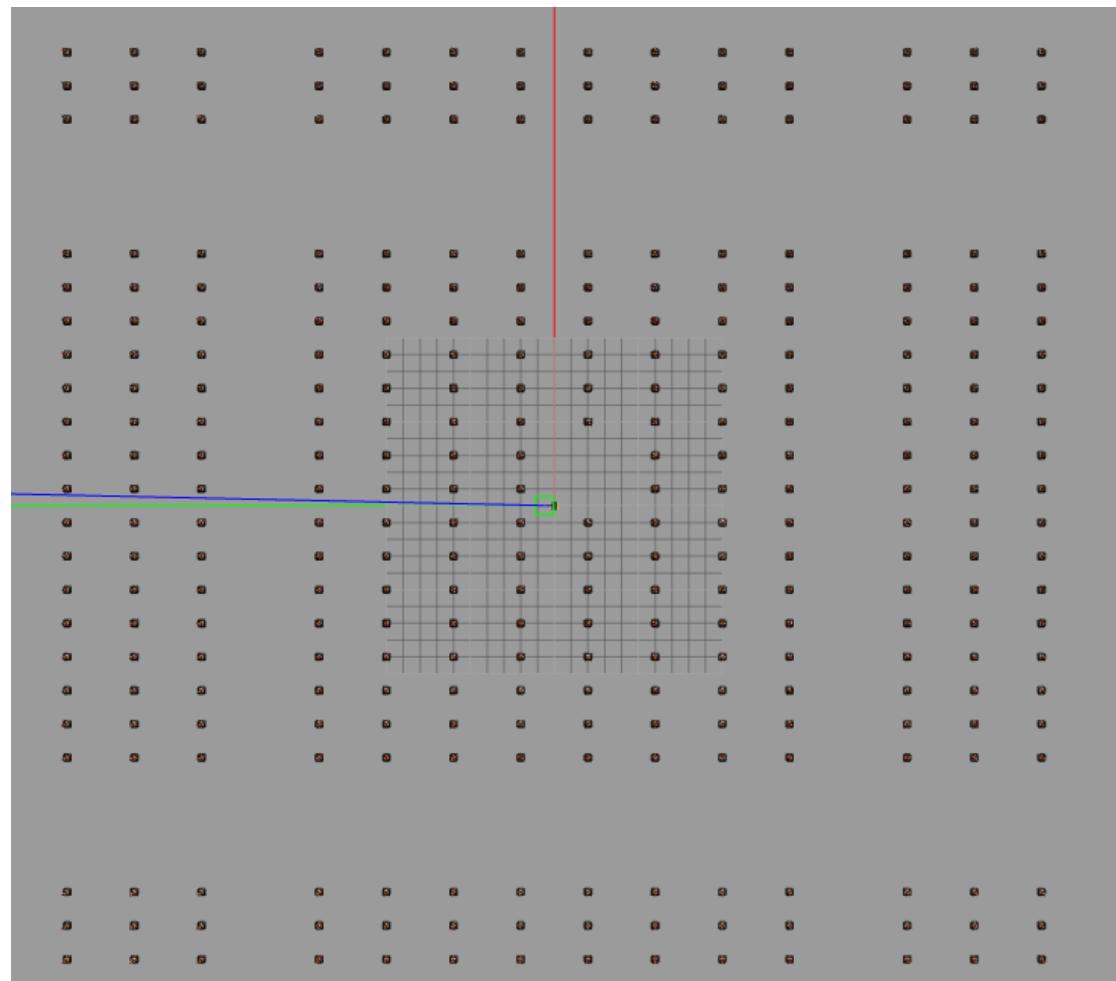


## Experiment #1 : single cone missing

- **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

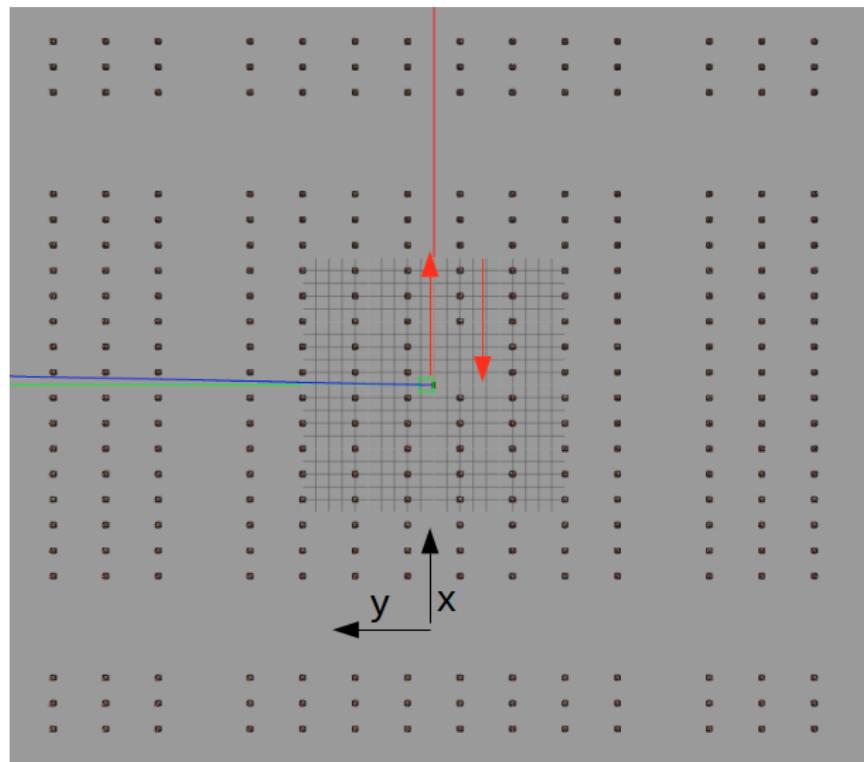
## Experiment #2 : two cones missing

- **World 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”.

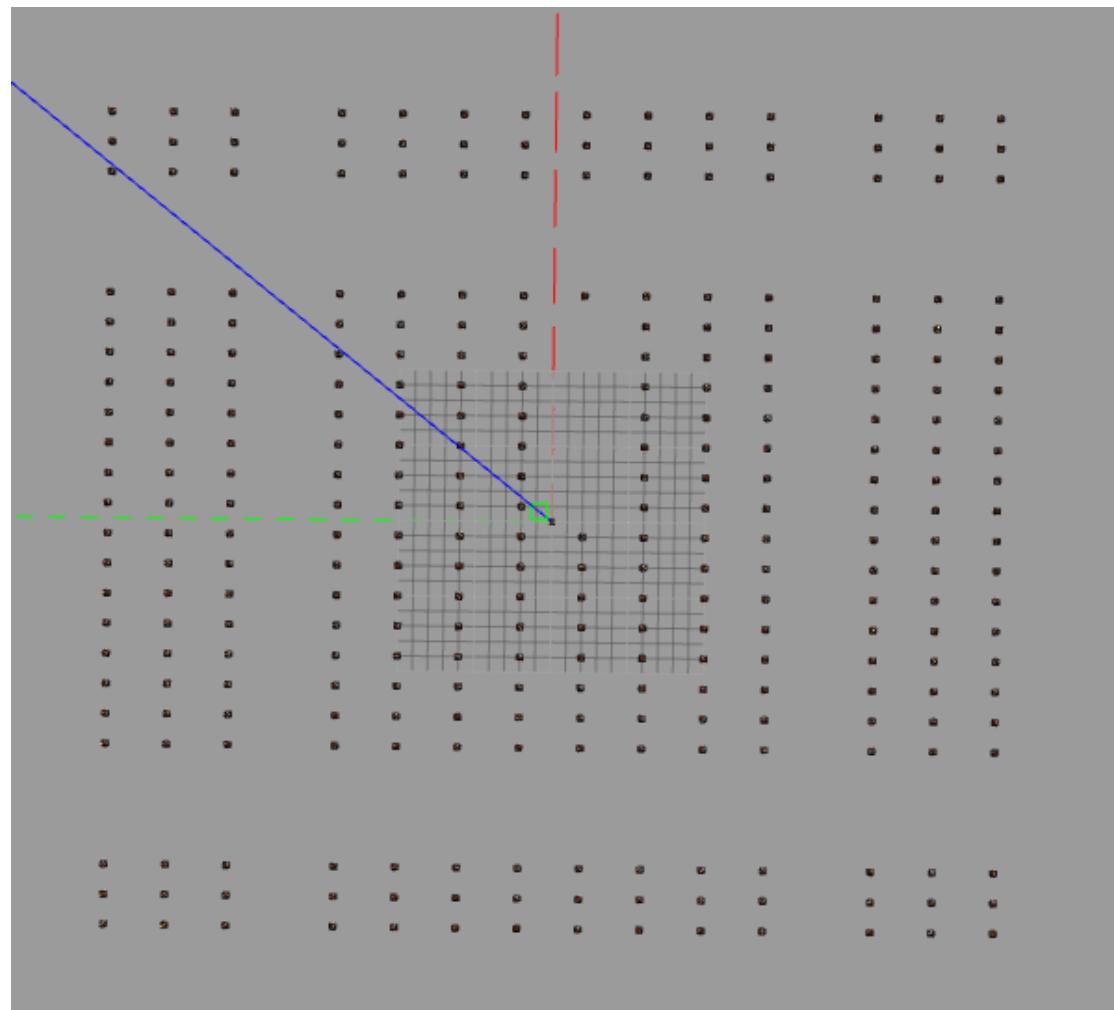


## Experiment #2 : two cones missing

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

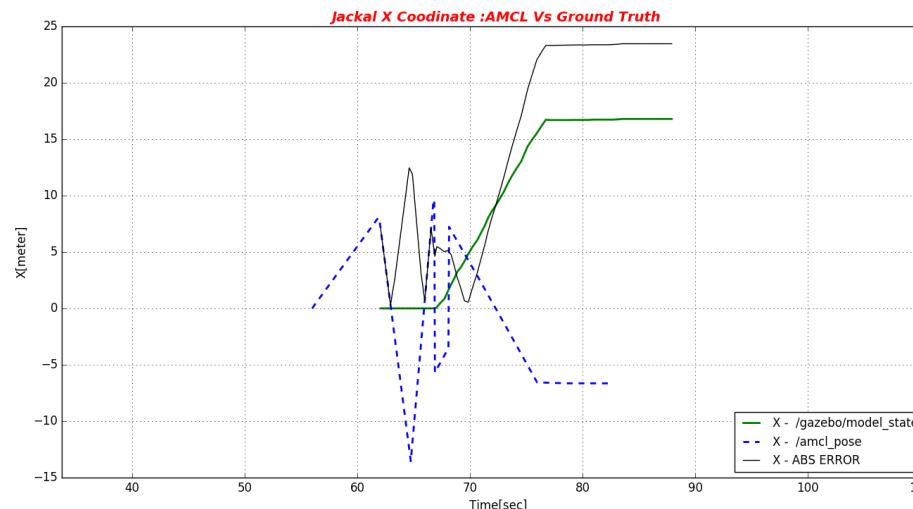
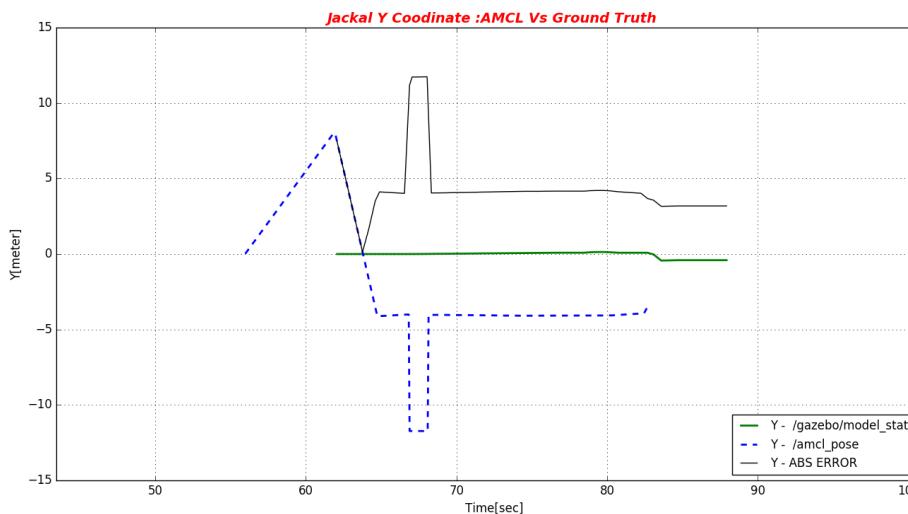
## Experiment #3 : half row missing

- **World 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:**



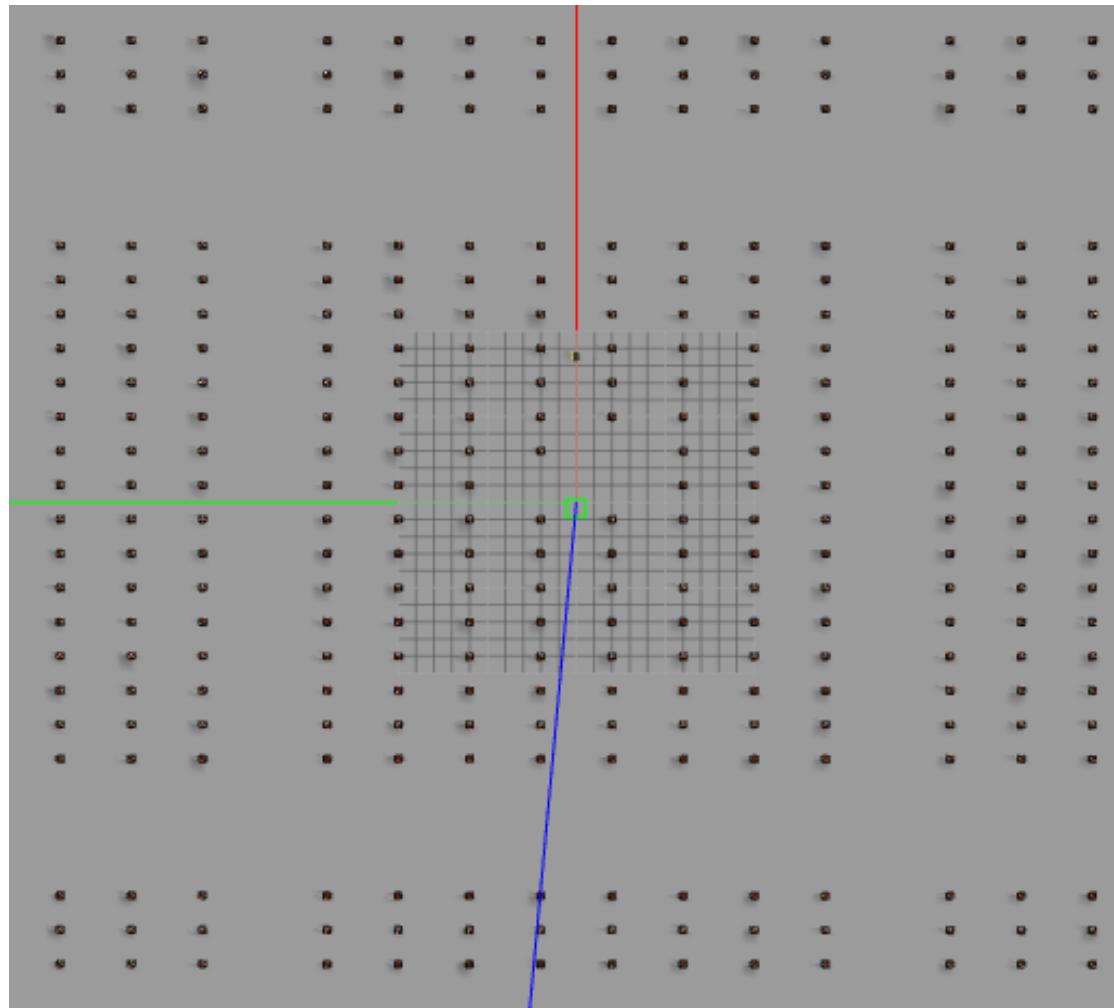
## Experiment #3 : half row missing

- **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.



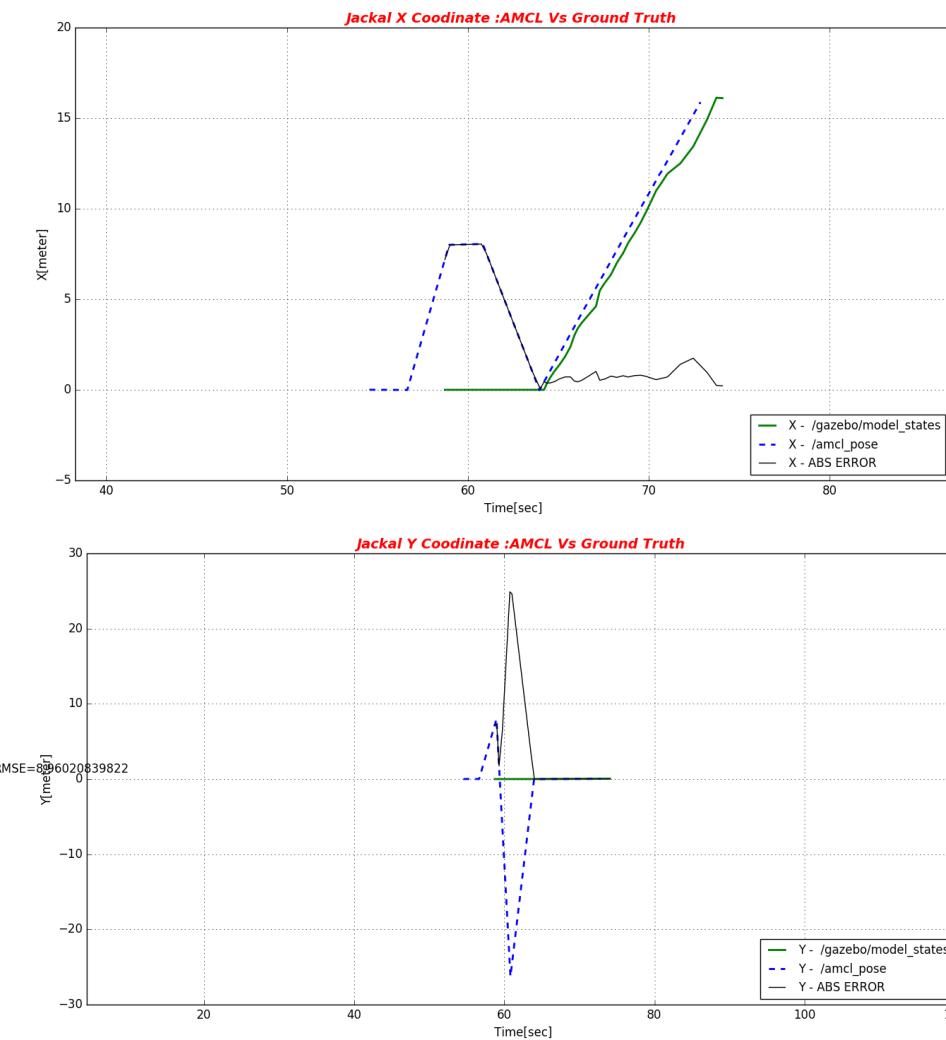
## Experiment #4 : 3 cones missing both sides

- **World 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:**



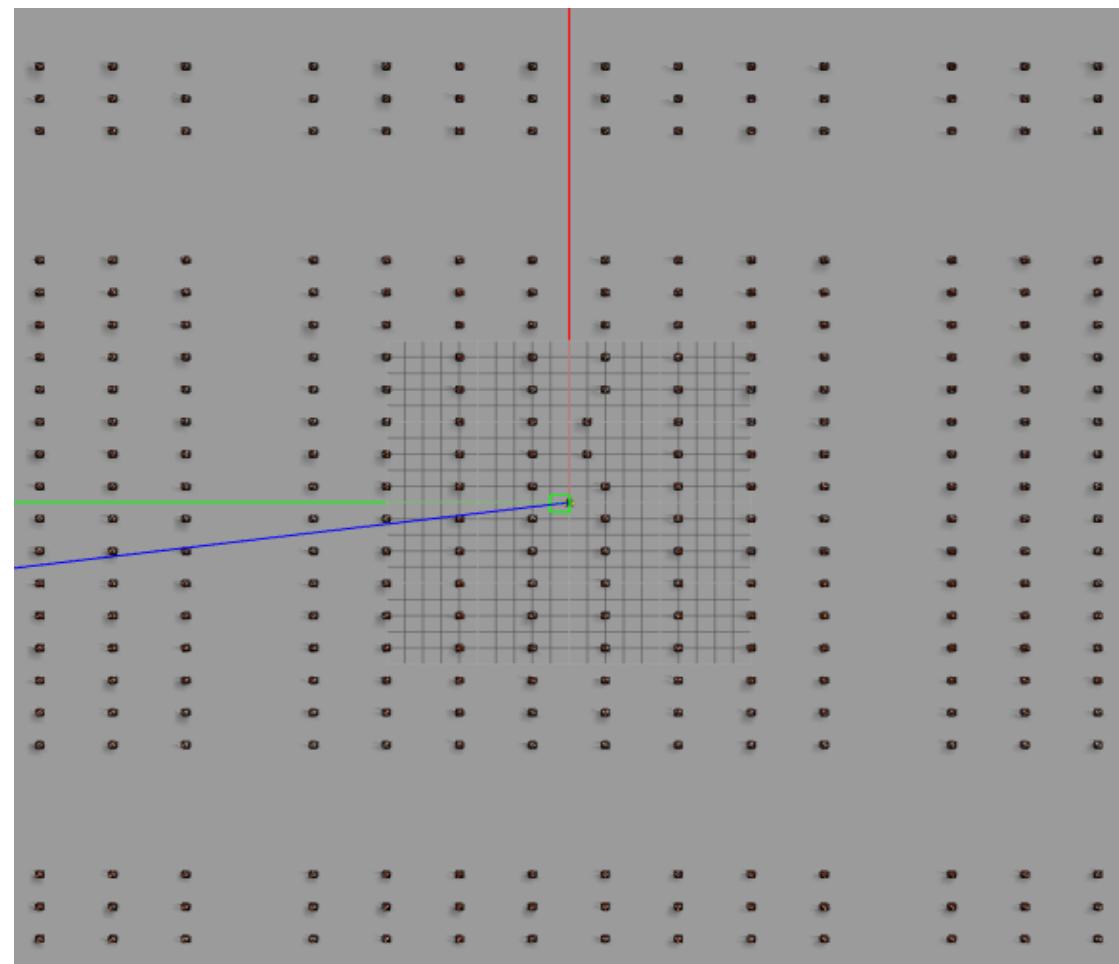
## Experiment #4 : 3 cones missing both sides

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



## Experiment #5 : two cones out of line

- **World 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:**

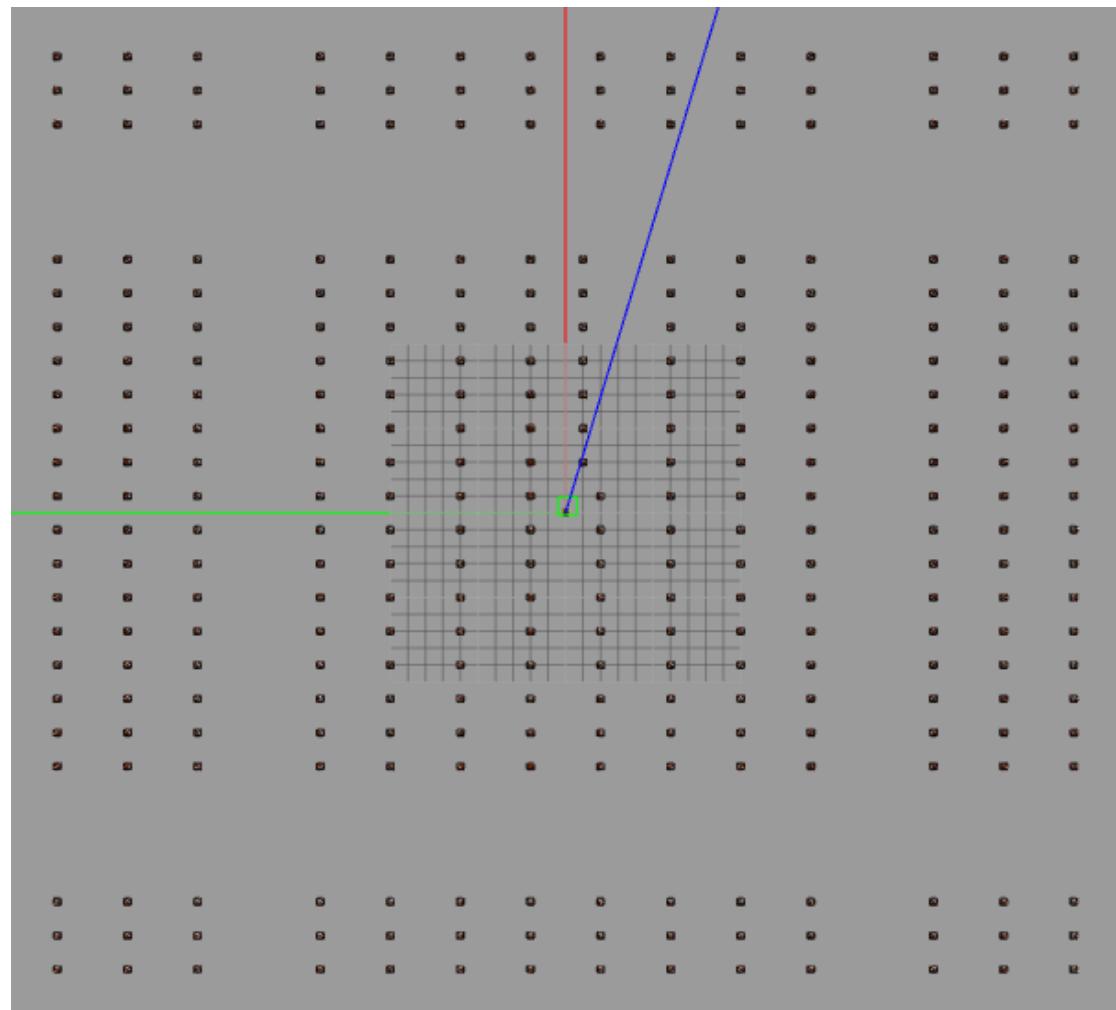


## **Experiment #5 : two cones out of line**

- 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.

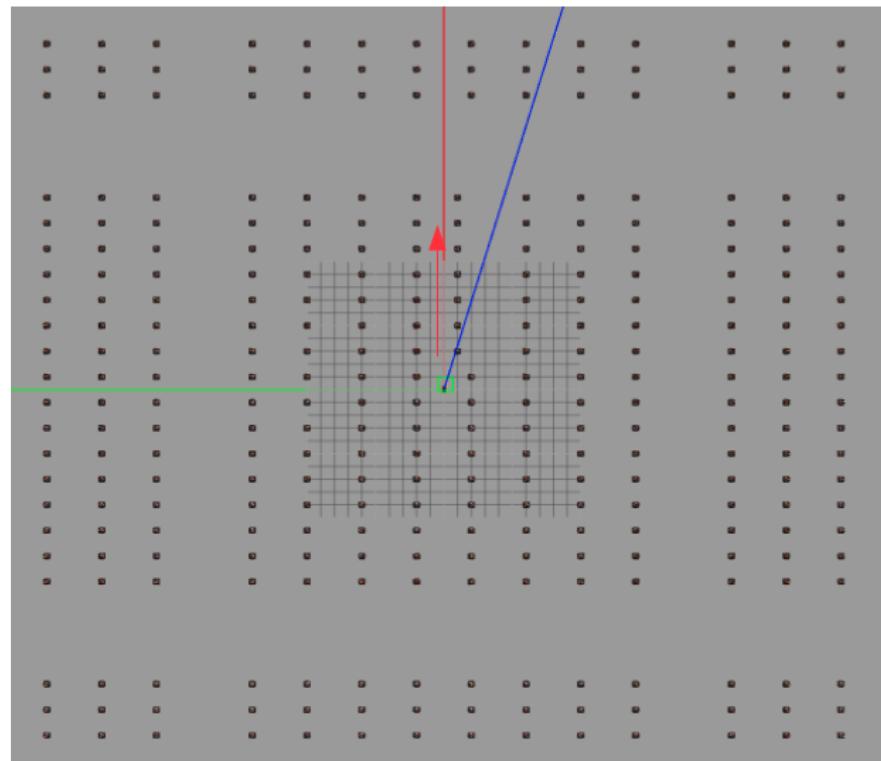
## Experiment #6 : half row of cones out of line

- **World 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:**



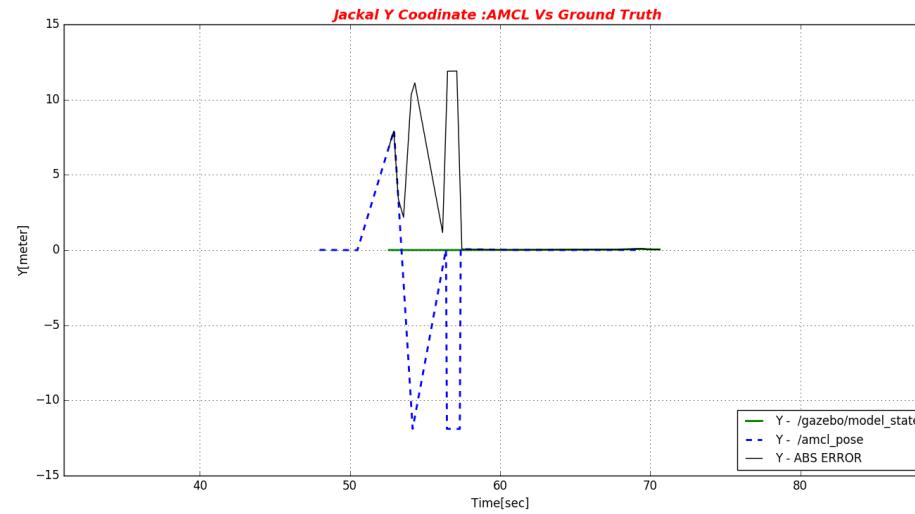
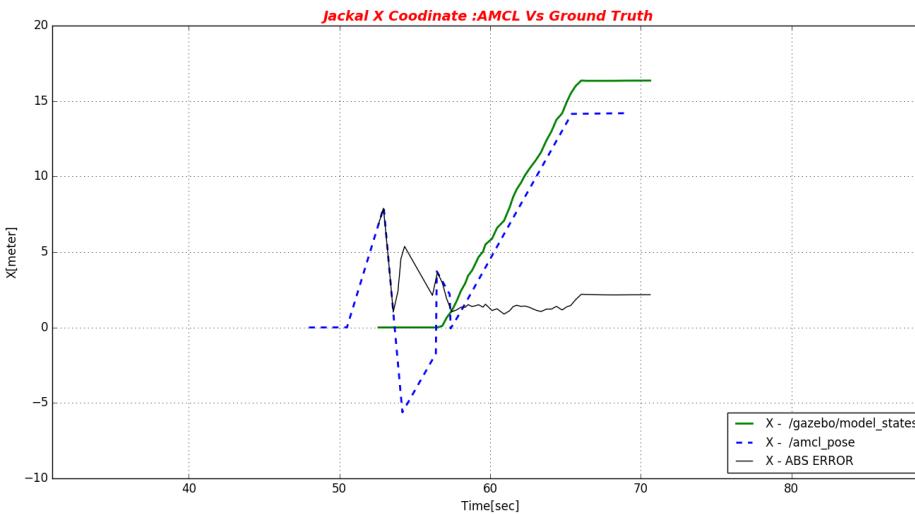
## Experiment #5 : half row of cones out of line

- **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).



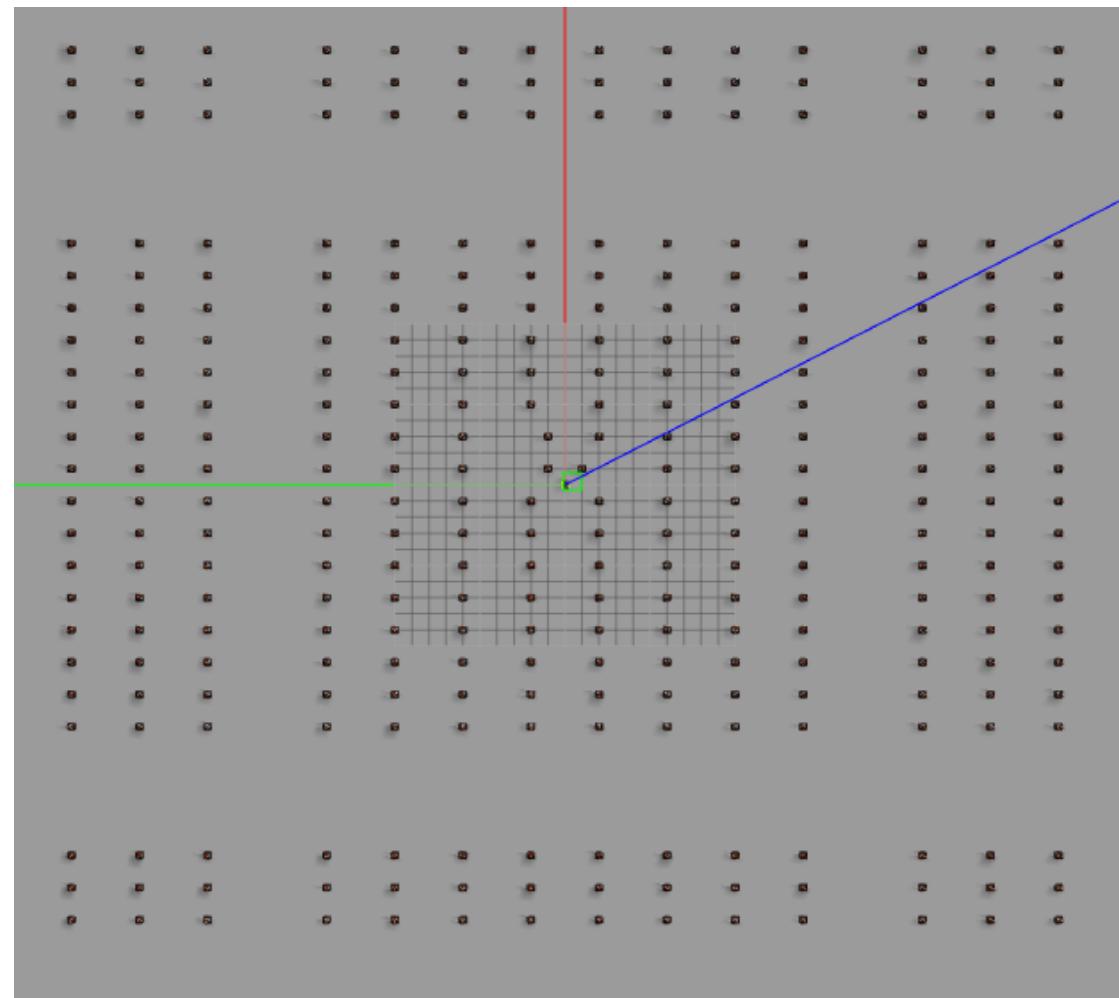
# Experiment #6 : half row of cones out of line

- Example success graph:



## Experiment #7 : 3 cones out of line both sides

- **World 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:**

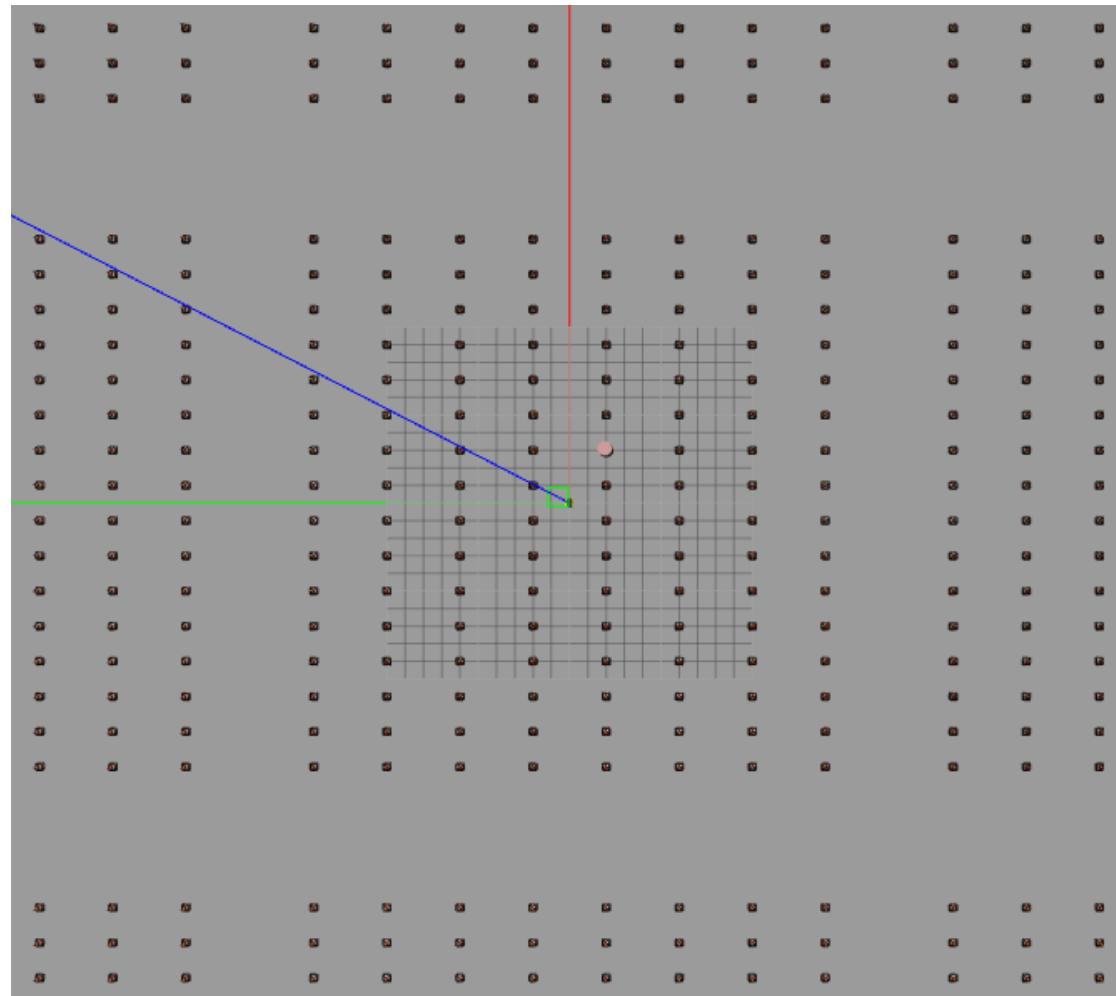


## **Experiment #7 : 3 cones out of line both sides**

- 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

- **World 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:**

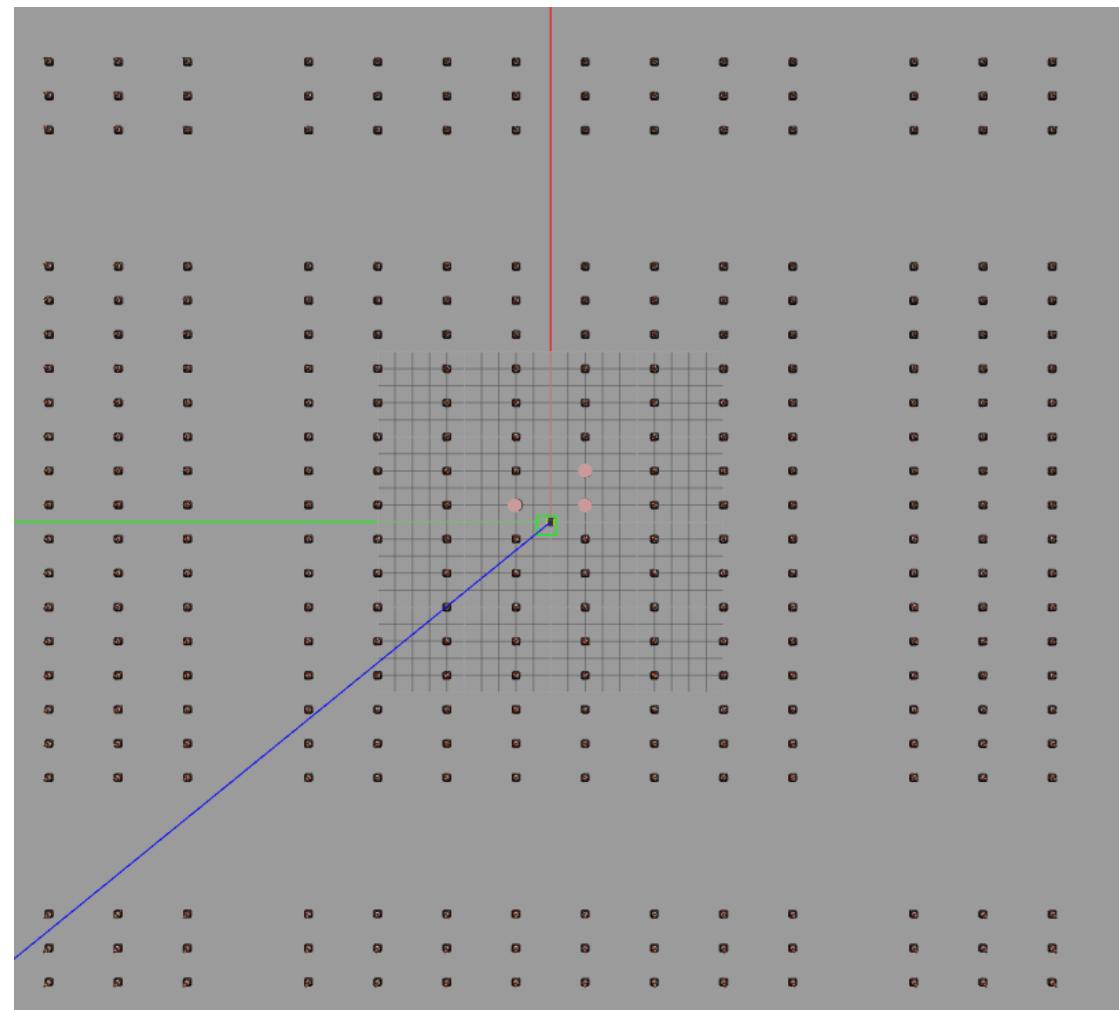


## Experiment #8 : Single cylinder

- 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.

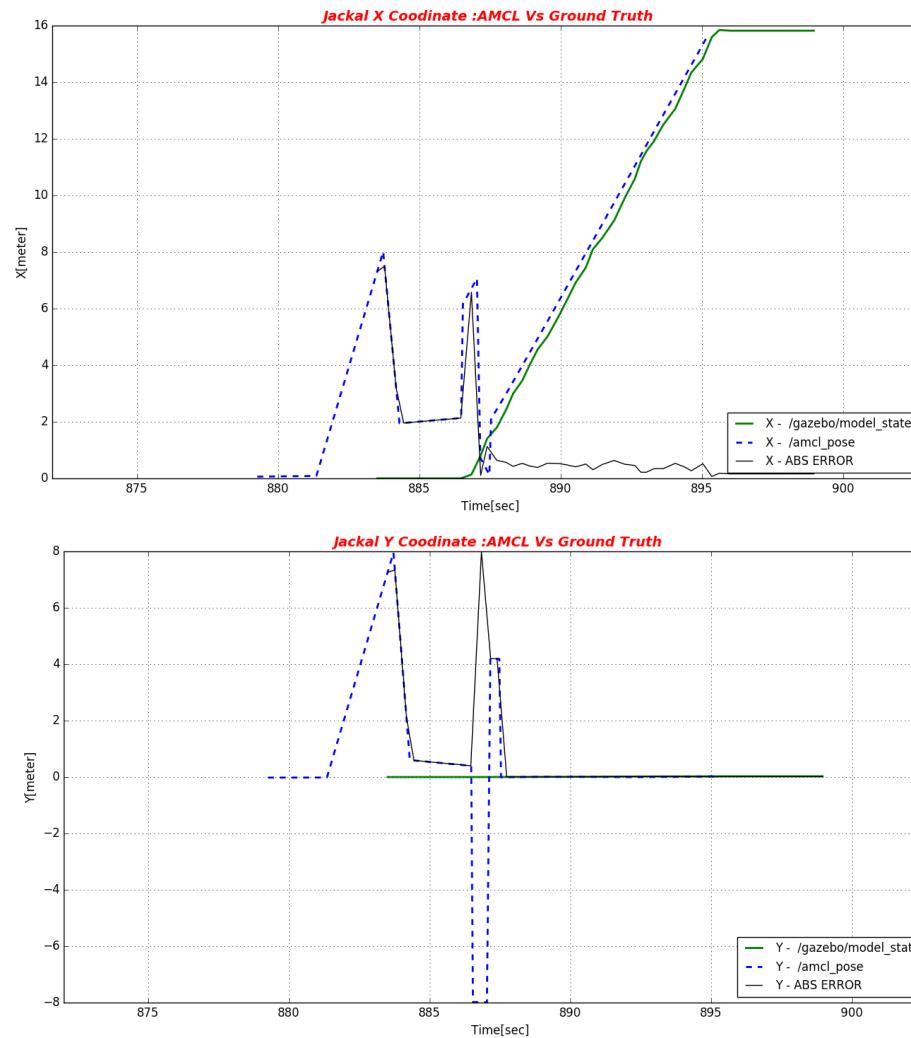
## Experiment #9 : 3 cylinders both sides

- **World 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:**



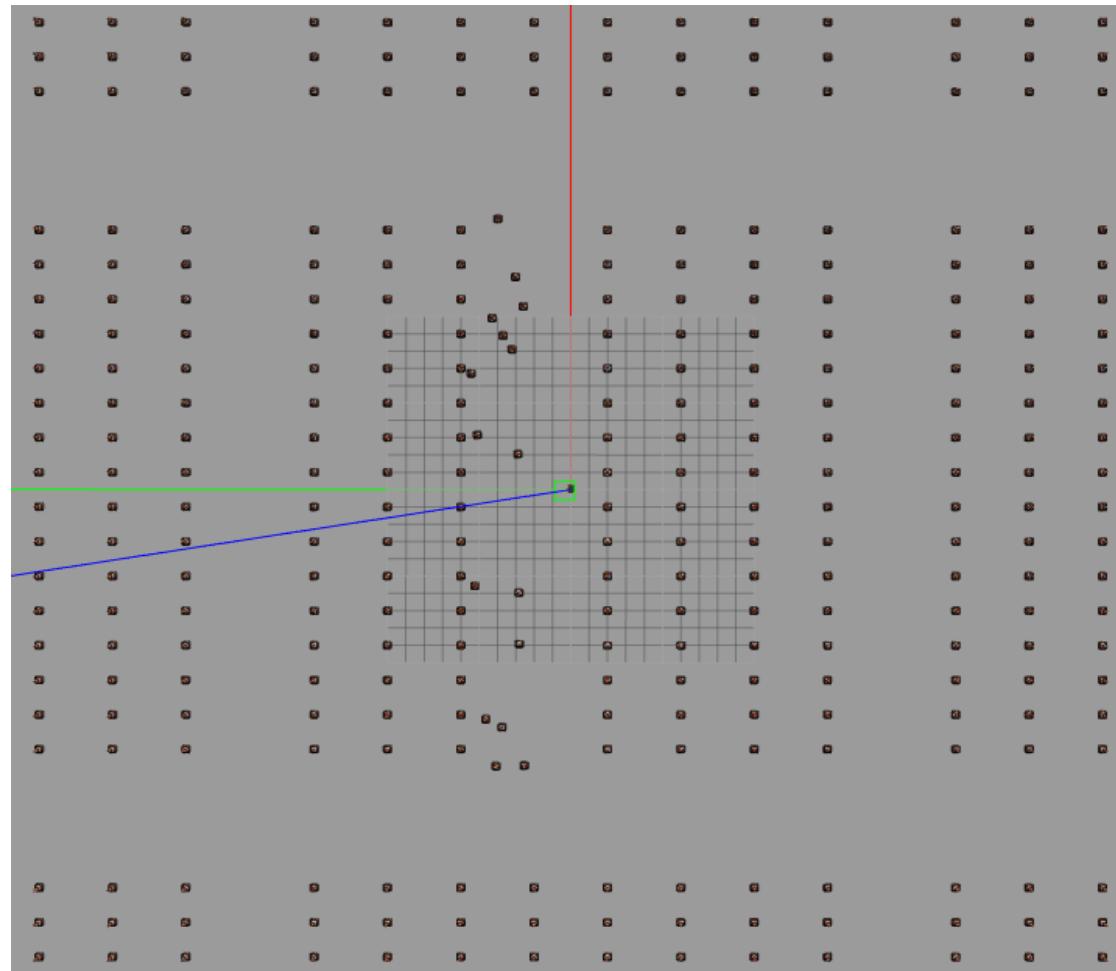
## Experiment #9 : 3 cylinders both sides

- 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:



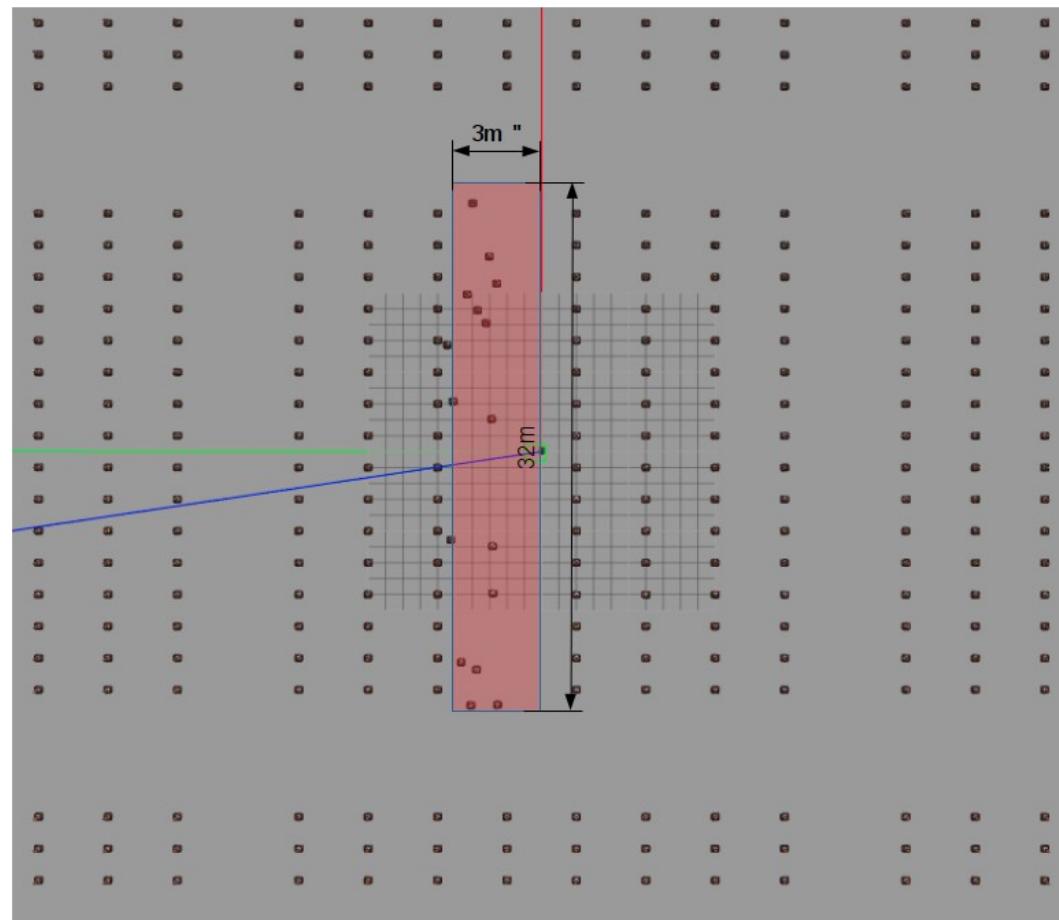
## Experiment #10 : random row of cones

- **World 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:**



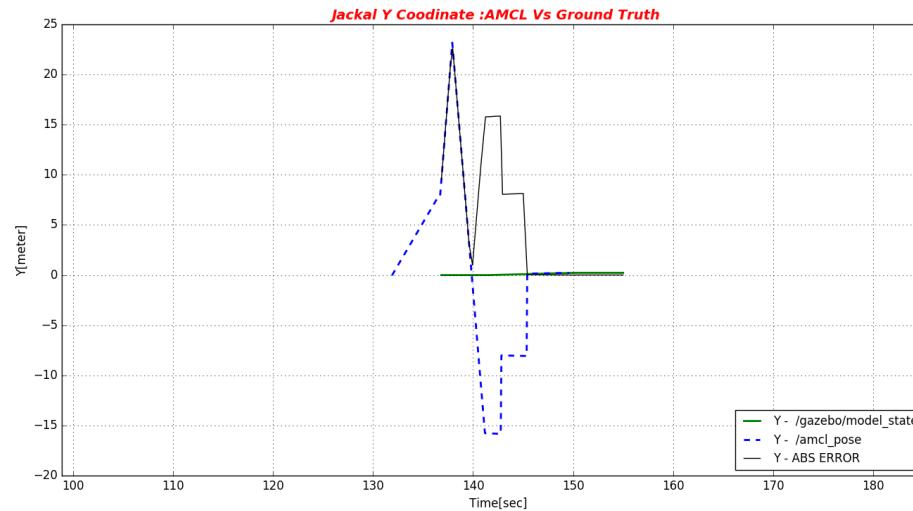
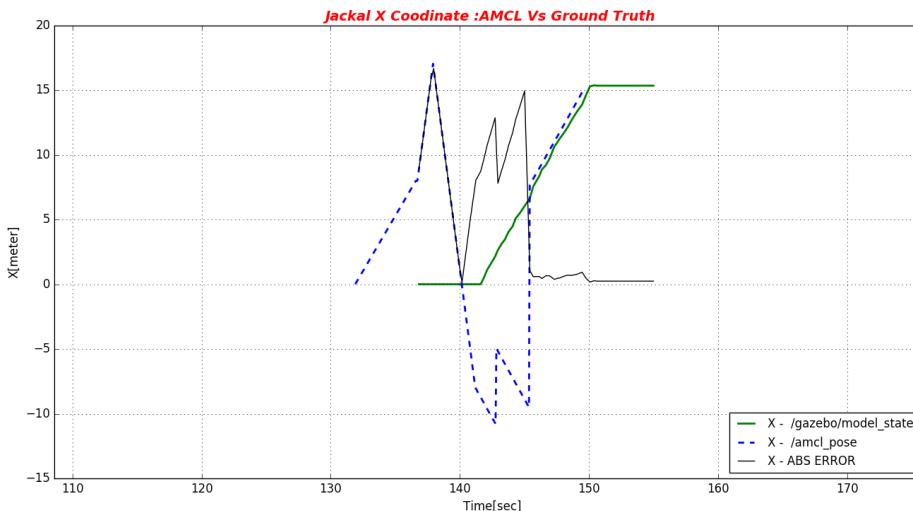
## Experiment #10 : random row of cones

- **World 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=](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.



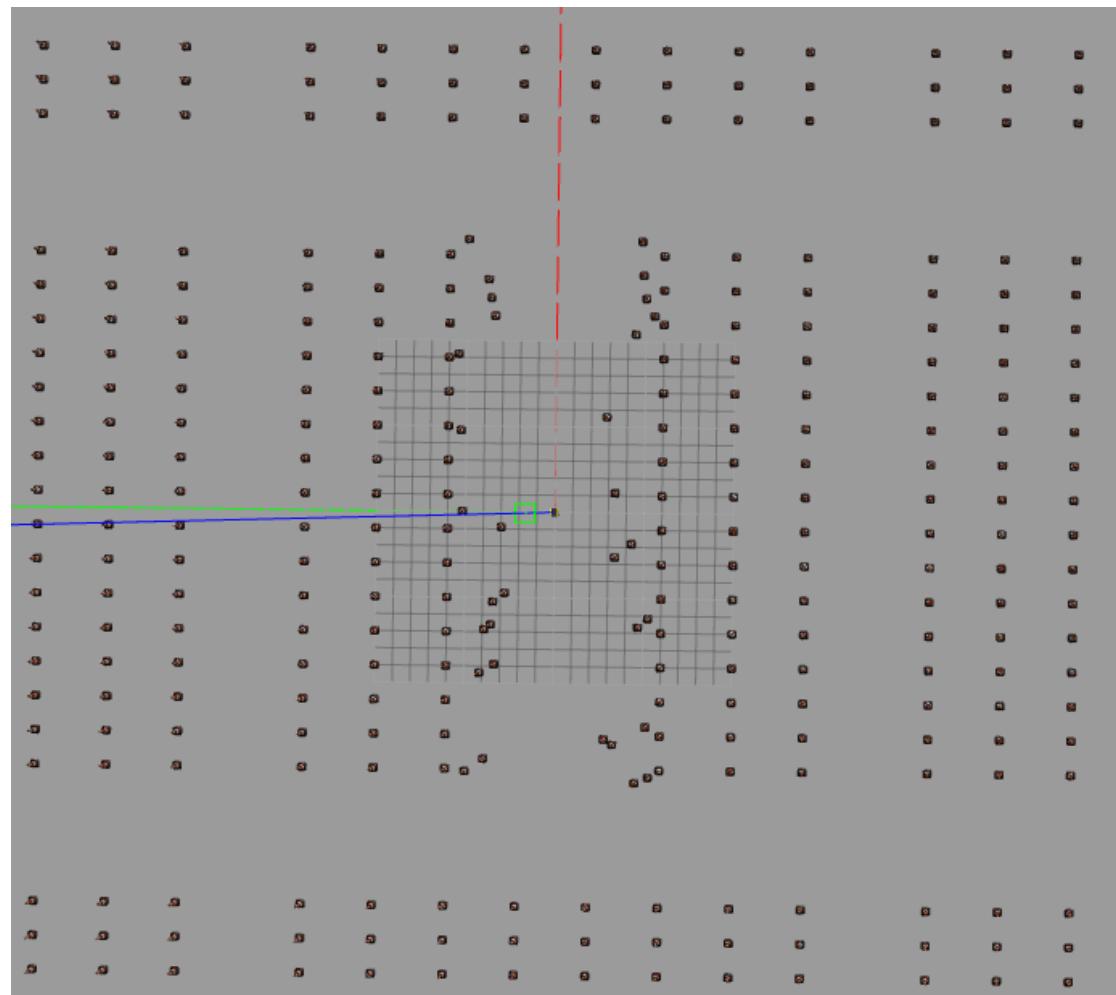
# Experiment #10 : random row of cones

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



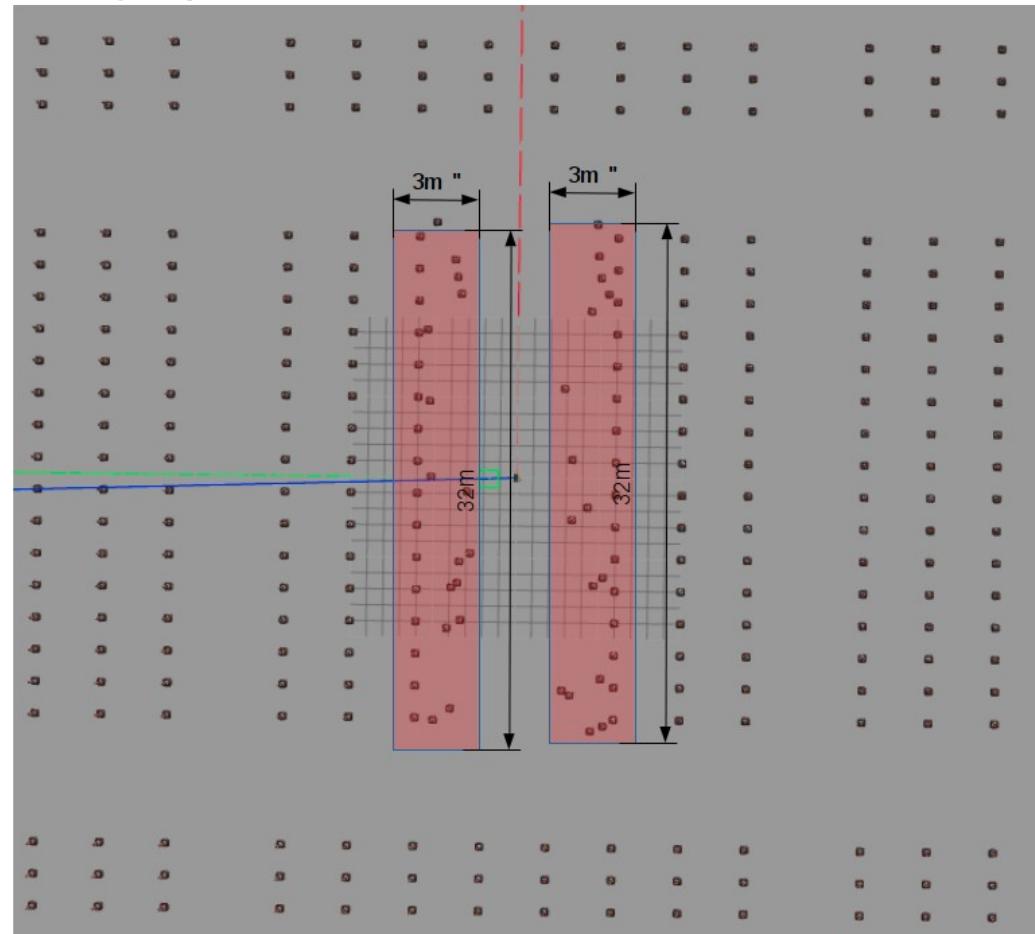
## Experiment #11 : two random rows of cones

- **World 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:**



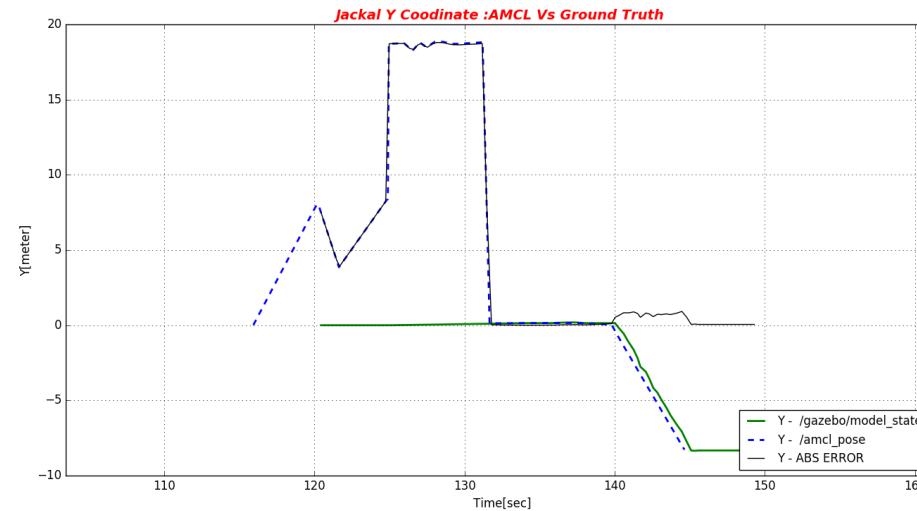
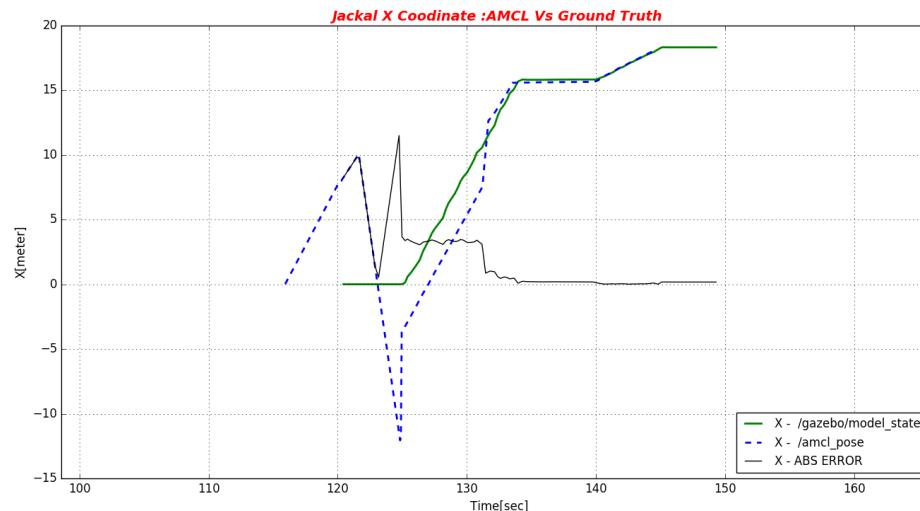
## Experiment #11 :two random rows of cones

- **World 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=](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.
- The algorithm located the robot correctly in both axis
- Example success graph:



## 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 from 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 yielded ~80% success.