Part 1. Simulate a Lidar

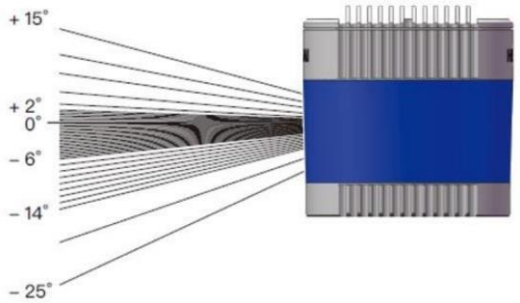
(Run learn_lidar.py)

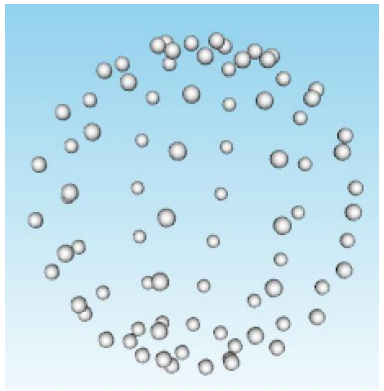
* If you can't imagine how a lidar illuminates an object, think of it as a light bulb.

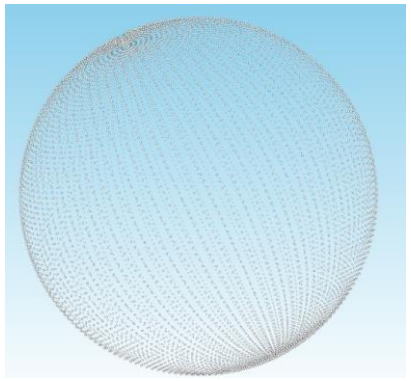
* So, generate a ball, extract points from it and check if it is the result we want.

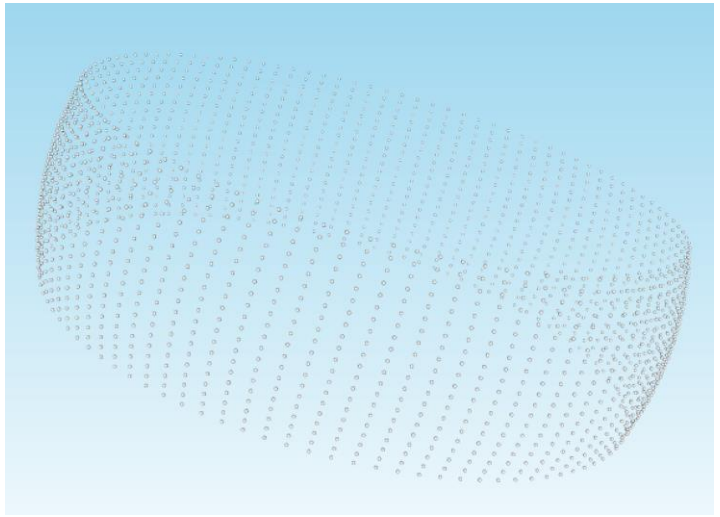
* If you think it is too sparse and you want to increase the sampling frequency, modify res_theta and res_phi to see the differences.

* If we cut off the top and bottom of the ball, we can get the view shown in the right figure.





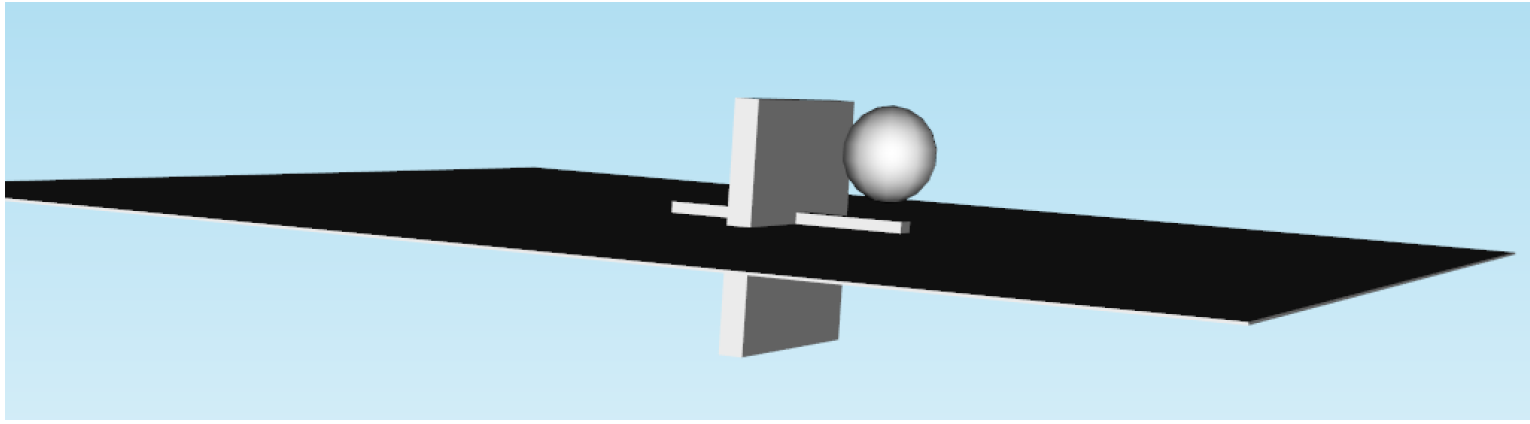




Part 2. Make a Stage

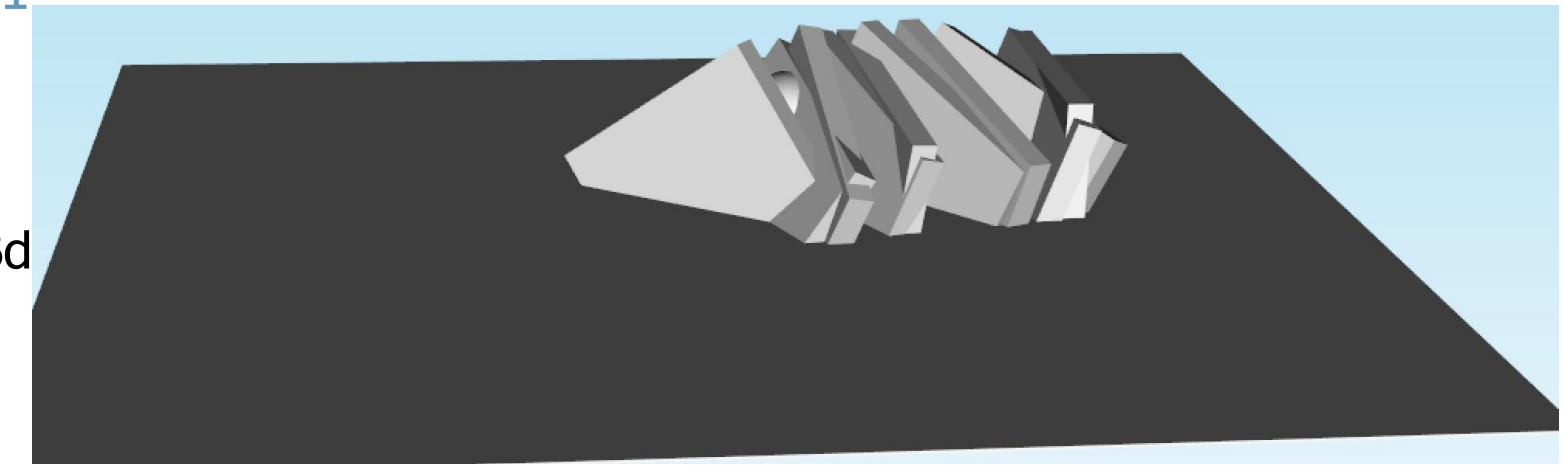
(Run Stage.py)

```
@staticmethod
def make_default_stage3d():
    stage = Stage3d()
    stage.addObject(utils.makeCube(center=[0, 0, 0], shape=[10, 100, 100]))
    stage.addObject(utils.makeCube(center=[0, 0, 0], shape=[100, 10, 10]))
    stage.addObject(utils.makeCube(center=[0, 0, 0], shape=[500, 500, 1]))
    stage.addObject(utils.makeSphere(center=[30, 30, 30], radius=20))
    return stage
```



Add some thing you like, make use of the function,
`utils.transModel`
There are four arguments.

```
mesh,  # the origin shape
rot=[0,0,0], # rotation in xyz
pos=[0,0,0], # translate in xyz
scale=1 # let model bigger for >1
```



You can make any Stage you like
Do not forget to let make_default_stage3d
Return the `make_your_stage3d function`

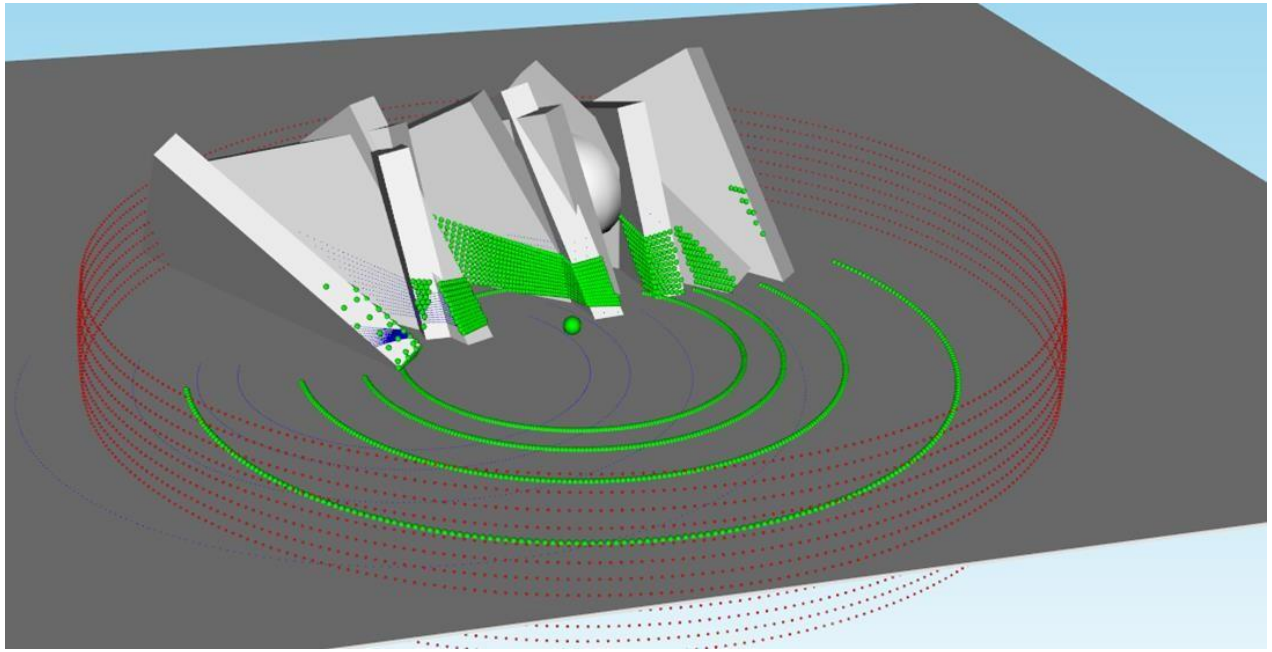
Part 2. Make a Stage

(Run LidarCore.py)

`lidar.setLidarPosition` and `li.scan(renderStep=True)` are two key functions.
You should input the position into lidar And let it scan.

When repeat it, the lidar system works. You

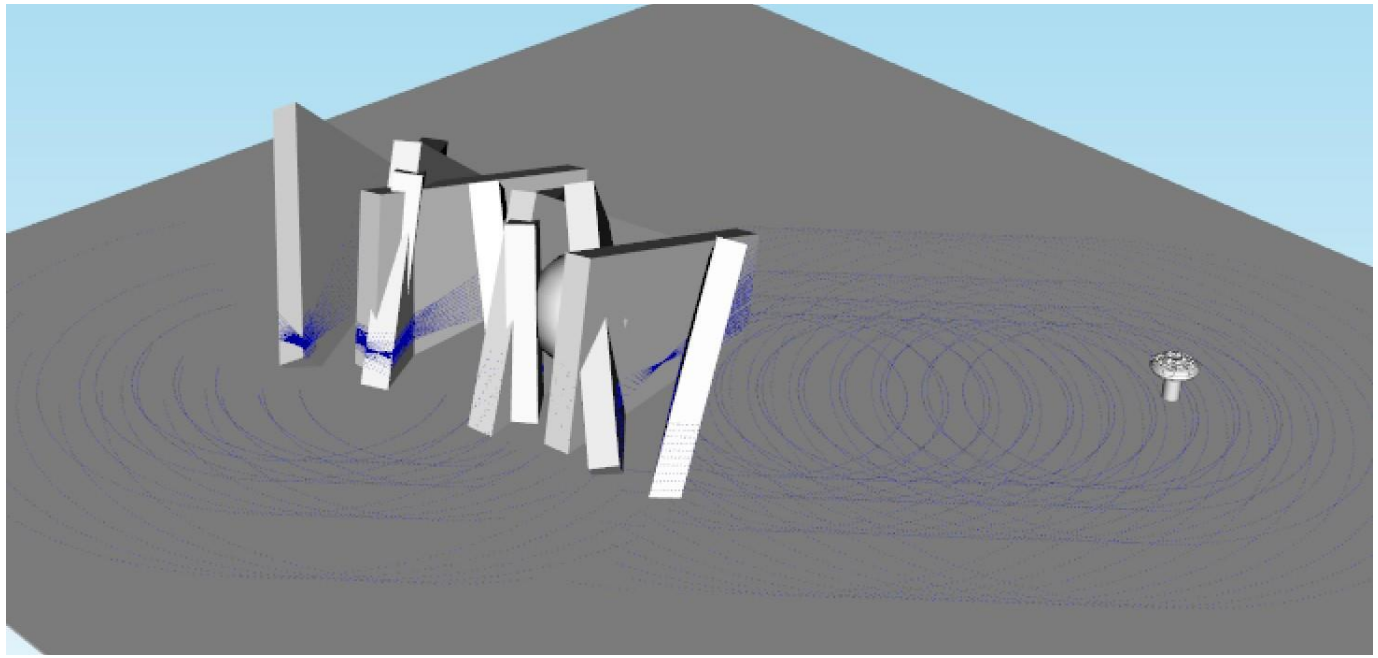
will get an output like that Image



Part 2. Scan Your Stage

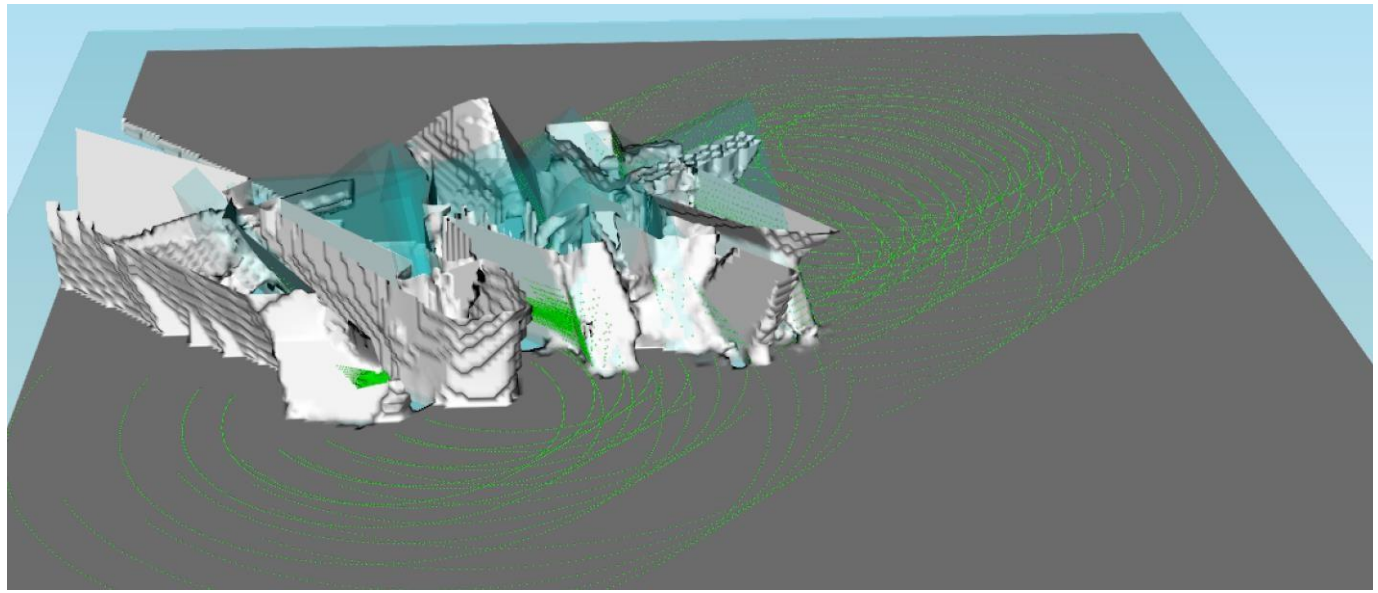
(Run AnimePlayer.py)

Moidfy the function named test_anime_your_model()
You can load your own .obj model for fun. In this case, UFO is used.



Run showWhatYouSampled()

You can use this function to reconstruct your point cloud into a 3d model.



Part 3. Semantic Segmentation

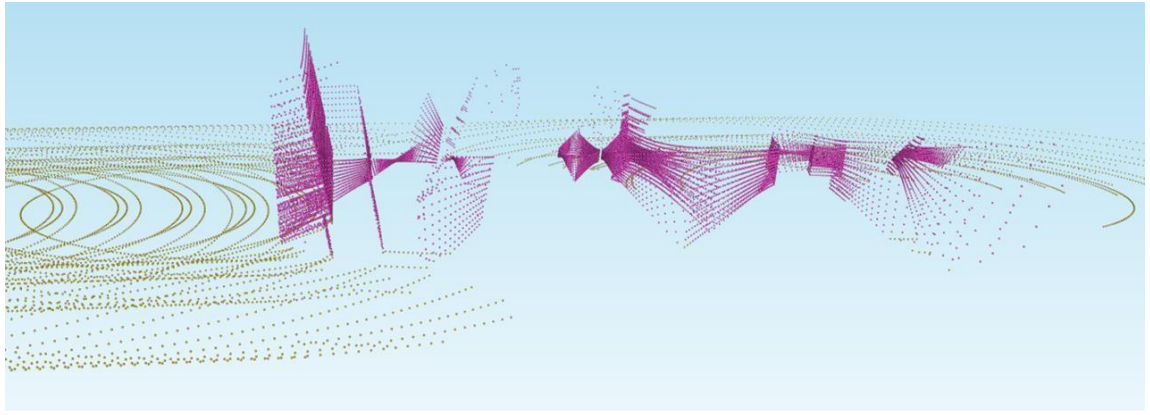
(Run PointsSegmentation.py)

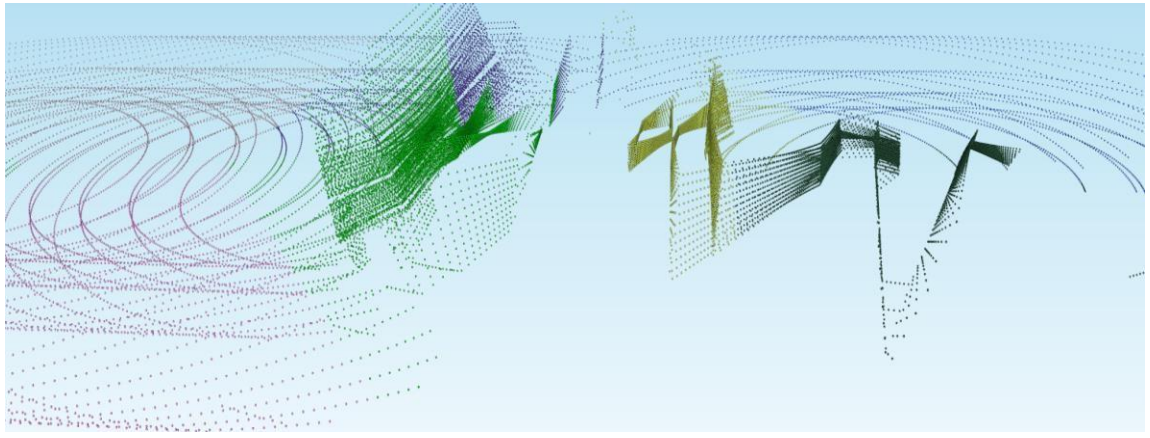
segmentation(x, 'rule')

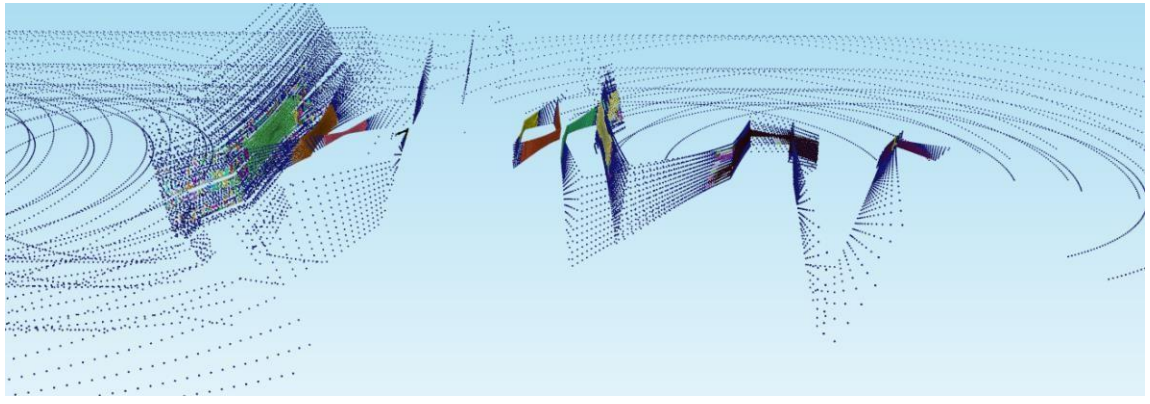
segmentation(x, 'kmeans')

segmentation(x, 'dbscan')

Compare your rule with machine learning algorithm.





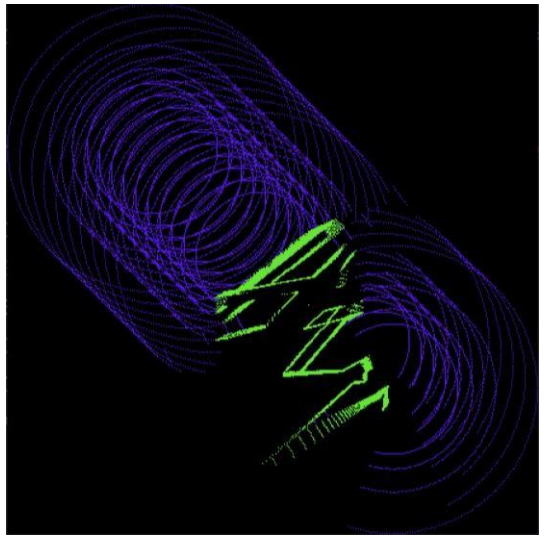


Part 3. Semantic Segmentation

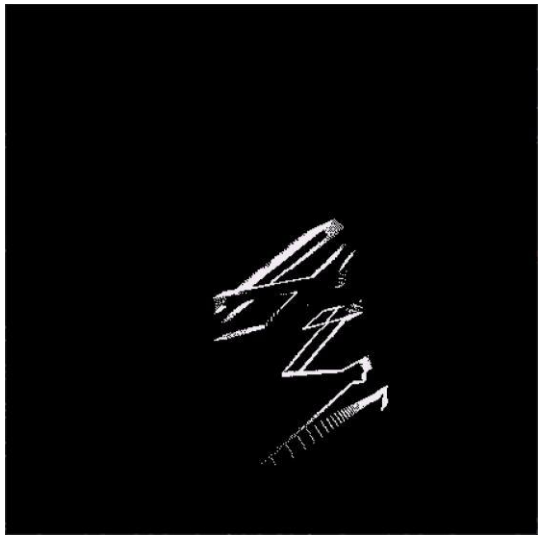
(Run PointsSegmentation.py)

Input your point segmentation into the function `points2Image`

And you will get a result like that.



view_map

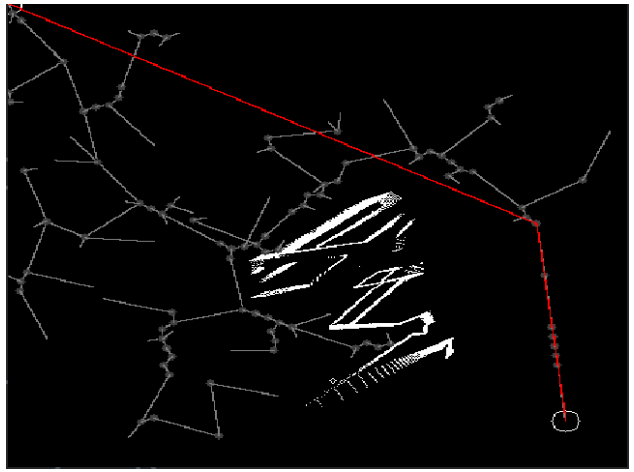


Obj_map

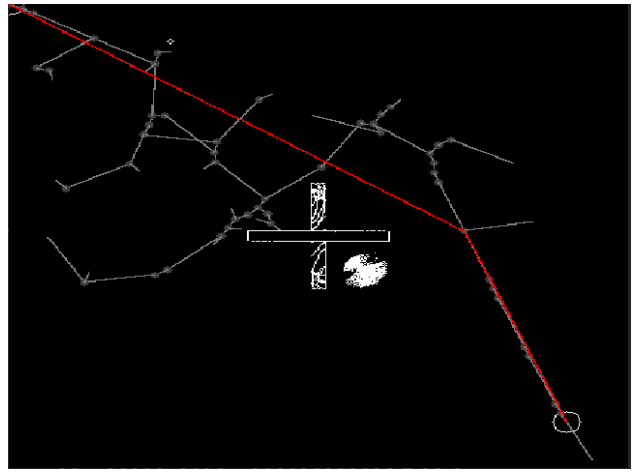
Hence, remove the points of ground, using points2Image, you will get a obj_map like img2

```
(Run rrt_mystage.py)
# you can change the start/end for fun
```

You will get a result like that.



rrt_result



rrt_result

And congratulations you finished all the base tutorials of navigation!

Part 4. Learn How RRT Works.

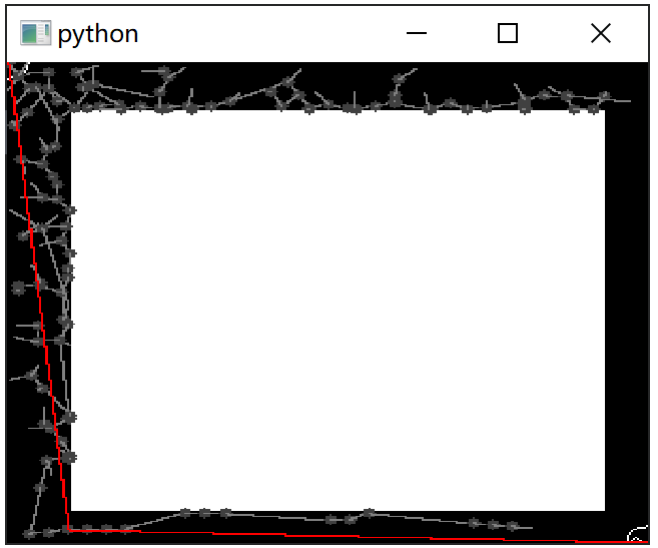
There are four environments encountered by the robot.
The first picture is for you to test the effect of RRT. (case_0() in RRT_Test.py)

Second: what the algorithm should do when the path is completely blocked.
Third: RRT's big problem.

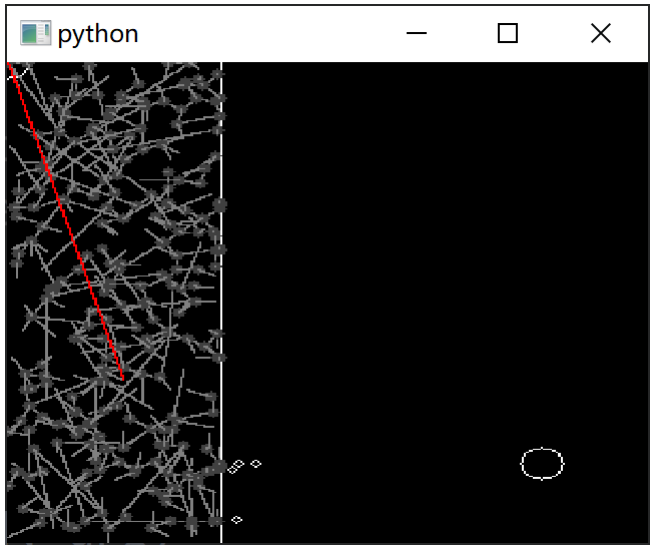
Fourth: The dilemma of RRT.

You need to modify the RRT parameters to solve these questions.

After doing these demo, you will understand the meaning of each parameter of RRT.

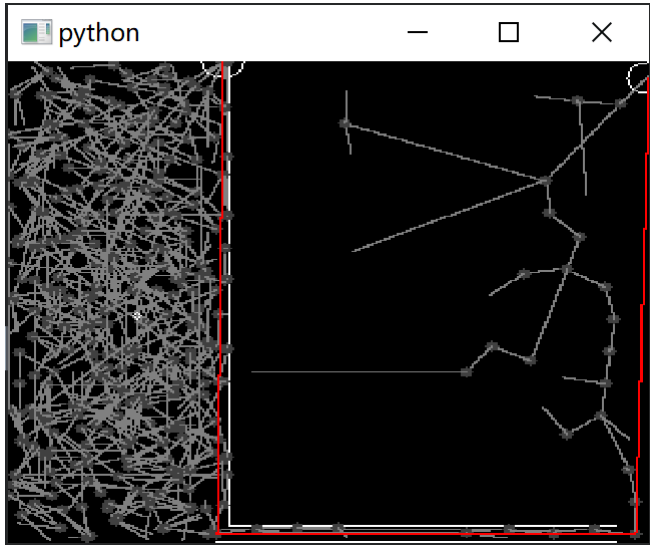


example



blocked

Bottleneck



Neck with open air