1a. It will encounter tennis ball, people and border. Various of tennis ball data will be needed for recognition training. The challenges it will face is that it must find the same ball from its left eye and right eye, otherwise it will not catch the ball correctly because there is more than one ball in the field. In its world, the world consists of two separate photos, it can only reconstruct the stereo world through algorithm.

Robot

Obstacles

Tennis Ball

Border

Need train data for robot training for obstacles such as chair, person etc.

Treat borders as obstacles. Use CannyEdgeDetector to detect

Need train data for robot training for tennis ball recognition

Compute the 3D location of each object using left photos and right photos and do route planning and grab the tennis balls.

1b.

Compute 3D coordinates of the tennis ball

Yes

Has tennis ball?

Take Pictures

Start Position

No

Go back to start position, Dance

FaceDirection change

grabObject(X,Y,Z)

Detect Border

moveToLocation(X,Y,Z) the 3D coordinates of the tennis ball

Route planning

Detect obstacles

1c.

Def PickupBalls(switch\_on, X, Y, Z):

While switch\_on:

no\_ball = True

while no\_ball:

change faceDirection(X, Y, Z)

img = Capture ()

if tennis\_ball in img:

no\_ball = False

(X\_b, Y\_b, Z\_b) = Compute3D (Xr\_ball, Yr\_ball, Xl\_ball, Yl\_ball, K, [R|t])

Border = CannyEdgeDectector(img)

Obstacles = DetectObstaclesWithinBorder(img)

3D\_obstacles = []

For i in range(len(Obstacles)):

(X\_i, Y\_i, Z\_i) = Compute3D (Xr\_i, Yr\_i, Xl\_i, Yl\_i, K, [R|t])

3D\_obstacles.append((X\_i, Y\_i, Z\_i))

(X\_cur, Y\_cur, Z\_cur) = (X, Y, Z)

While not (X\_cur, Y\_cur, Z\_cur) == (X\_b, Y\_b, Z\_b):

T = RoutePlanner((X\_cur, Y\_cur, Z\_cur), 3D\_obstacles)

faceDirection(((X\_cur, Y\_cur, Z\_cur)+T)

(X\_cur, Y\_cur, Z\_cur) = moveToLocation((X\_cur, Y\_cur, Z\_cur)+T)

grabObject(X\_b, Y\_b, Z\_b)

While not (X\_cur, Y\_cur, Z\_cur) == (X, Y, Z):

T = RoutePlanner((X\_cur, Y\_cur, Z\_cur), 3D\_obstacles)

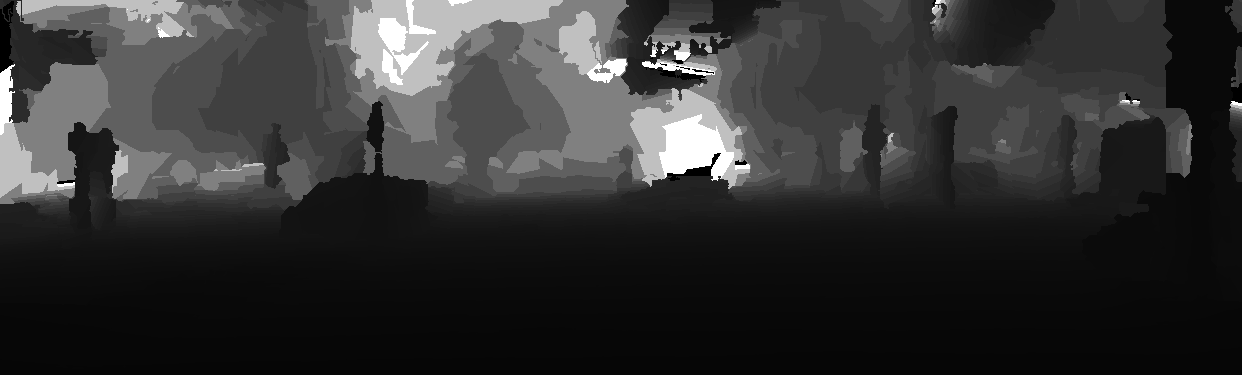
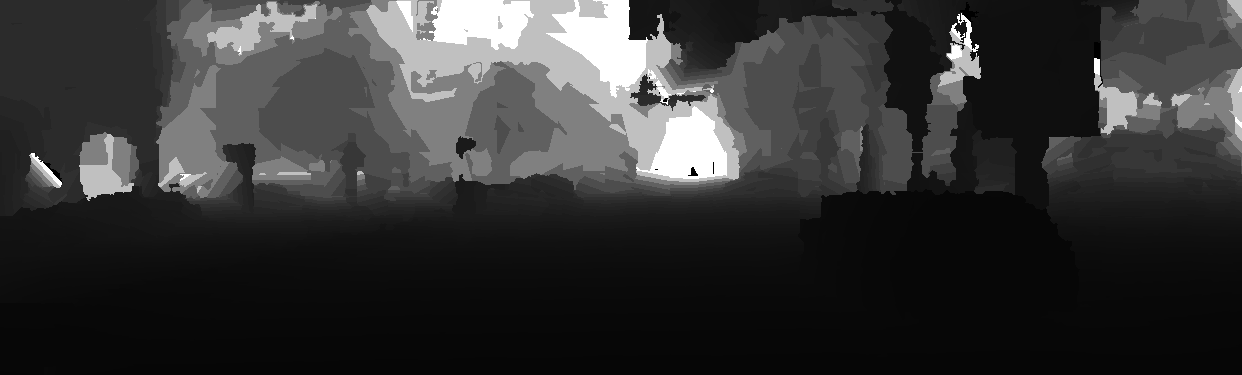
faceDirection(((X\_cur, Y\_cur, Z\_cur)+T)

(X\_cur, Y\_cur, Z\_cur) = moveToLocation((X\_cur, Y\_cur, Z\_cur)+T)

victoryDanceAtLocation(X, Y, Z)

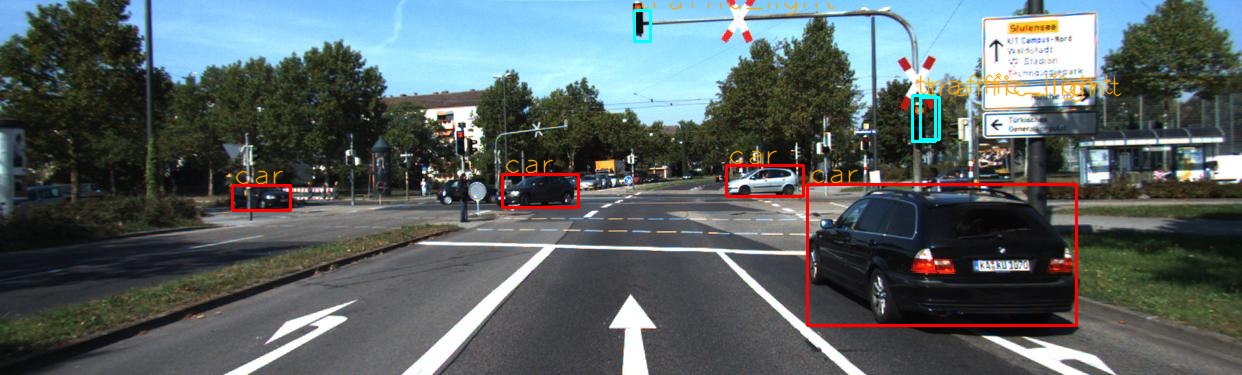
1d. At each position(x,y,z), the robot take a picture first and then it will need to move a certain distance (x,y,z) + T to the right to pretend it has an right eye, then take another picture, and go back to the position(x,y,z). Now, we have baseline T and two photos, so in this way we can pretend the robot has its other eye(lens) back and we can calculate the 3D location of the object. If the robot cannot move a certain distance or the rotation and distance it moves are unknown, the it may not be able to complete its task after losing an eye.

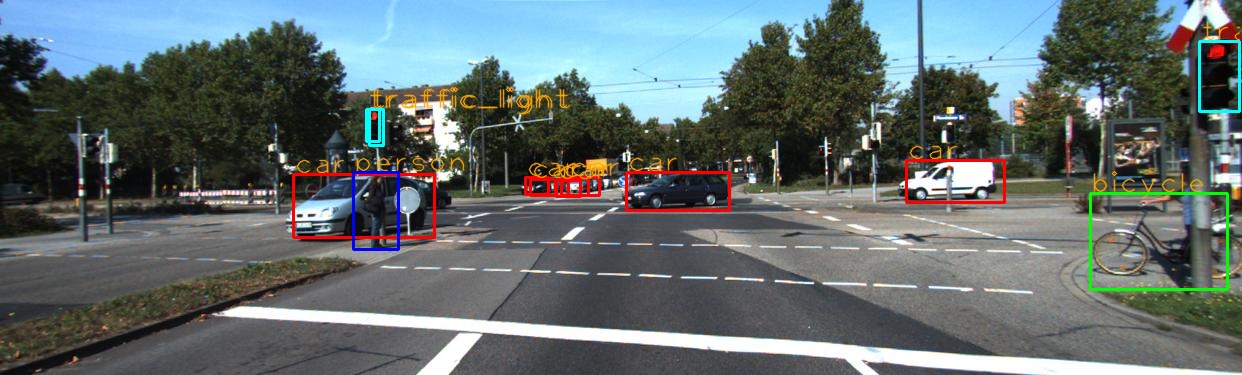
2a. The depth is calculated as follow: depth = baseline \* focal\_length/disparity. Here I just include 3 pictures for presentation purpose.

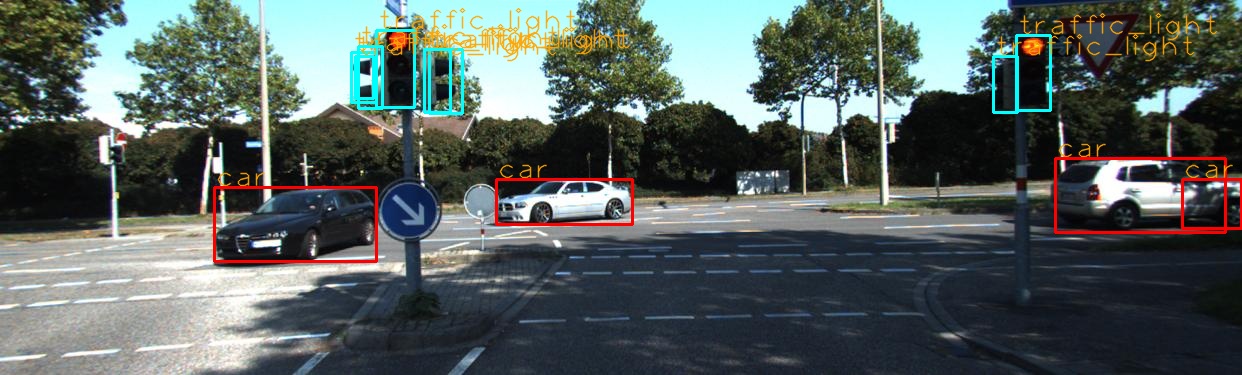
004945.jpg004964.jpg005002.jpg

2b. The detection is stored in detections.csv that I uploaded. I append the attributes to its column accordingly.

2c.

004945.jpg

004964.jpg

005002.jpg

2d. Please see code for compute3D(threshold) in assignment4.py. The formula for calculating 3D location is

In the code, I switch x and y because I treat vertical axis as x-direction and horizontal axis as y-direction. i.e.

2e.

004945.jpg

004964.jpg

005002.jpg

2f.

===========================Result for 004945.jpg============================

There is a car 3.2 meters to your right

It is 7.7 meters away from you

There is a car 3.4 meters to your left

It is 35.1 meters away from you

There is a car 10.3 meters to your right

It is 49.1 meters away from you

There is a car 23.2 meters to your left

It is 53.4 meters away from you

There is a traffic\_light 0.9 meters to your right

It is 20.7 meters away from you

There is a traffic\_light 7.6 meters to your right

It is 19.1 meters away from you

There is a traffic\_light 34.0 meters to your right

It is 84.3 meters away from you

===========================Result for 004964.jpg============================

There is a person 5.4 meters to your left

It is 17.6 meters away from you

There is a bicycle 8.1 meters to your right

It is 13.4 meters away from you

There is a car 3.3 meters to your right

It is 35.1 meters away from you

There is a car 5.7 meters to your left

It is 17.7 meters away from you

There is a car 16.7 meters to your right

It is 38.7 meters away from you

There is a car 2.8 meters to your left

It is 76.9 meters away from you

There is a car 7.8 meters to your left

It is 77.3 meters away from you

There is a car 7.2 meters to your left

It is 77.2 meters away from you

There is a car 3.8 meters to your left

It is 77.0 meters away from you

There is a car 4.4 meters to your left

It is 77.0 meters away from you

There is a traffic\_light 7.2 meters to your right

It is 11.2 meters away from you

There is a traffic\_light 5.5 meters to your left

It is 17.6 meters away from you

There is a traffic\_light 5.4 meters to your left

It is 17.6 meters away from you

===========================Result for 005002.jpg============================

There is a car 6.7 meters to your left

It is 16.8 meters away from you

There is a car 13.5 meters to your right

It is 22.7 meters away from you

There is a car 1.6 meters to your left

It is 25.7 meters away from you

There is a car 16.0 meters to your right

It is 25.0 meters away from you

There is a traffic\_light 2.2 meters to your left

It is 7.9 meters away from you

There is a traffic\_light 4.9 meters to your right

It is 9.8 meters away from you

There is a traffic\_light 9.7 meters to your left

It is 44.1 meters away from you

There is a traffic\_light 2.6 meters to your left

It is 8.2 meters away from you

There is a traffic\_light 2.6 meters to your left

It is 8.2 meters away from you

There is a traffic\_light 10.2 meters to your left

It is 44.3 meters away from you

There is a traffic\_light 2.6 meters to your left

It is 8.3 meters away from you

There is a traffic\_light 4.8 meters to your right

It is 10.0 meters away from you