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)

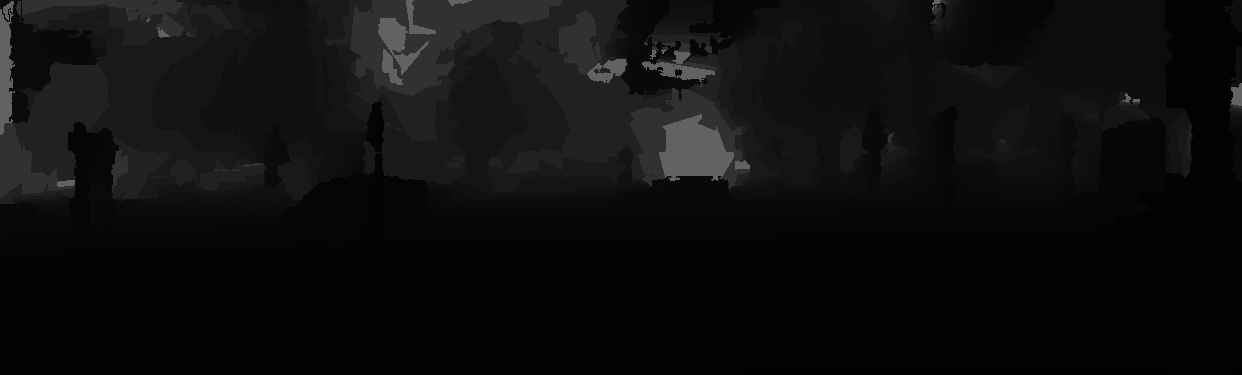
faceDirection(((X\_cur, Y\_cur, Z\_cur)+T)

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

victoryDanceAtLocation(X, Y, Z)

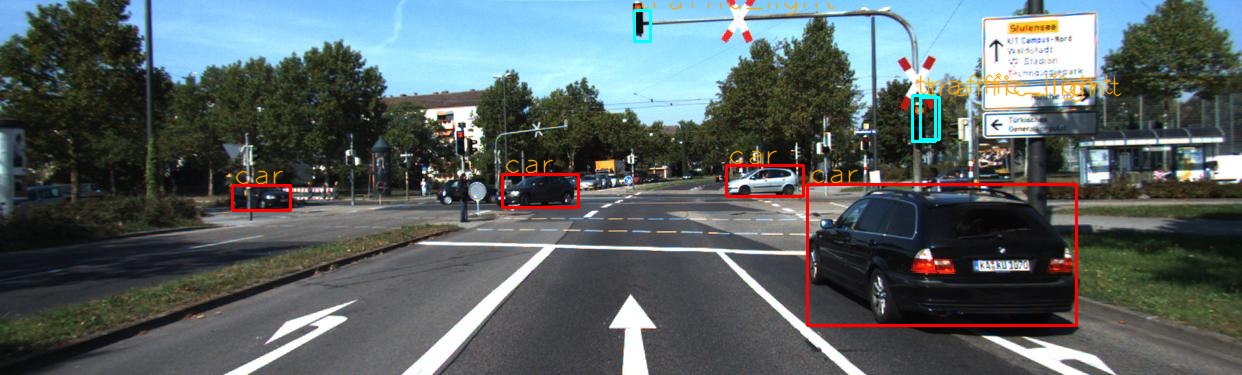
1d. At each position(x,y,z), the robot take a picture first and then it will need to move a certain distance (x,y,z) + T to the right to pretend it has an right eye, then take another picture, and go back to the position(x,y,z). Now, we have baseline t and two photos, so in this way we can pretend the robot has its other eye(lens) back. 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.

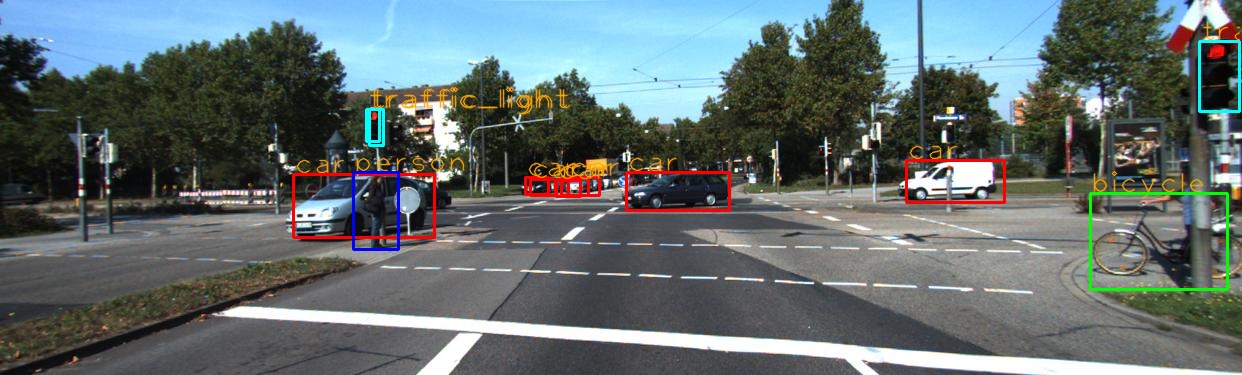
2a. The depth is calculated as follow: depth = baseline \* focal\_length/disparity. Here I just include 3 pictures for presentation purpose.

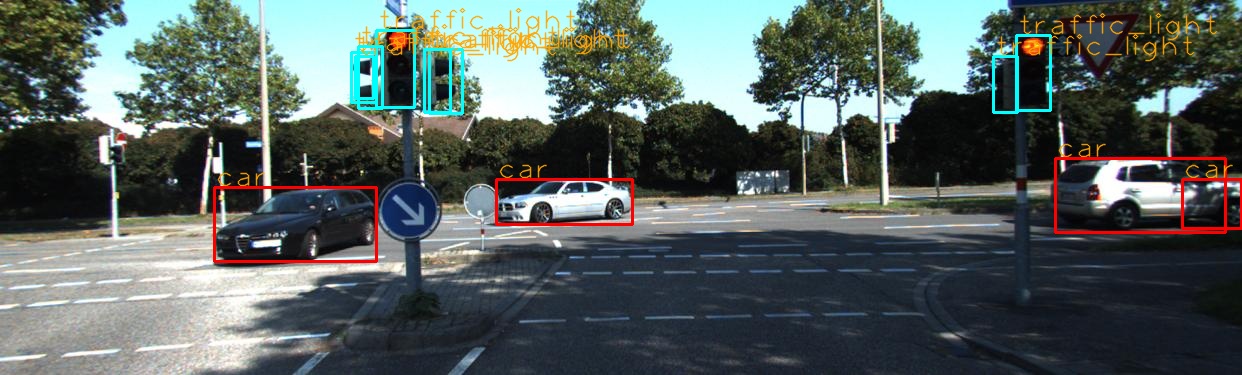


2b. The detection is stored in detections.csv that I uploaded.

2c.







2d.