ENPM 673

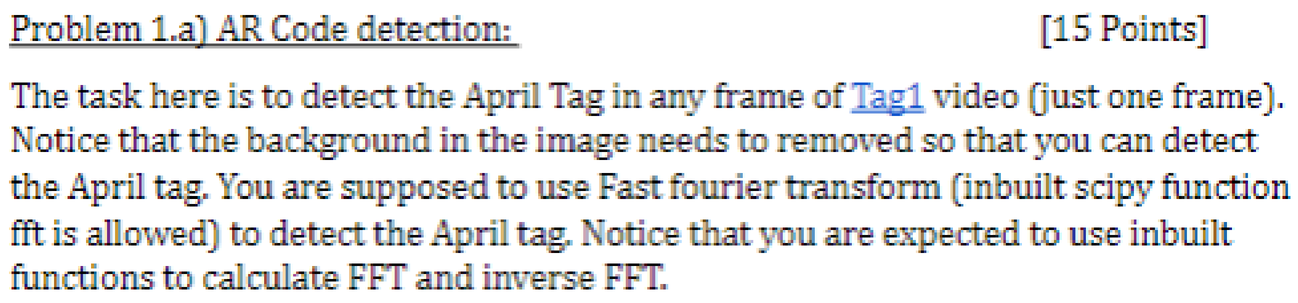
Perception of Robotics

Project 1

By

Mothish Raj Venkatesan Kumararaj

Question 1:



Problem 1.a) AR Code detection: [15 Points]

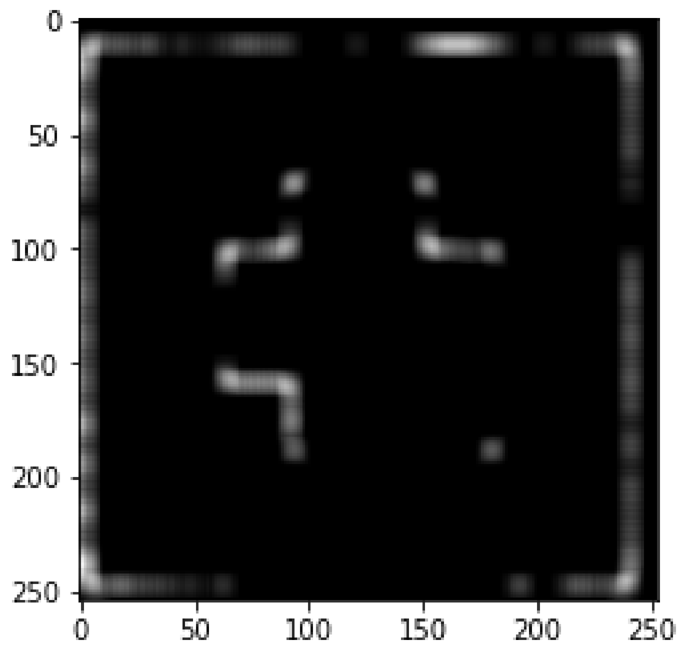
The task here is to detect the April Tag in any frame of Tag1 video (just one frame). Notice that the background in the image needs to removed so that you can detect the April tag. You are supposed to use Fast fourier transform (inbuilt scipy function fft is allowed) to detect the April tag. Notice that you are expected to use inbuilt functions to calculate FFT and inverse FFT.

Process:

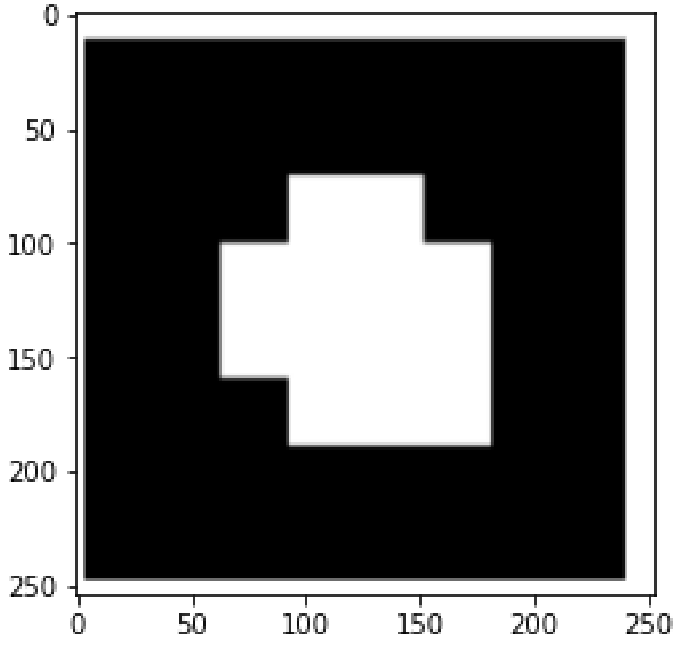
- The reference tag is imported in the python code using OpenCV.
- The tag is then done a FFT and FFT shift.
- Draw a circle in the center of the FFT shifted image and then convert the rest of the image to 0 binary format.
- The converted image is then inverse FFT shifted and FFT inverse is computed.
- The received image will consist of the lines of the paper.
- The FFT image is then warped to the size 160x160 by computing the homohraphy matrix, and binary threshold is applied to the image to get the tag

Results:

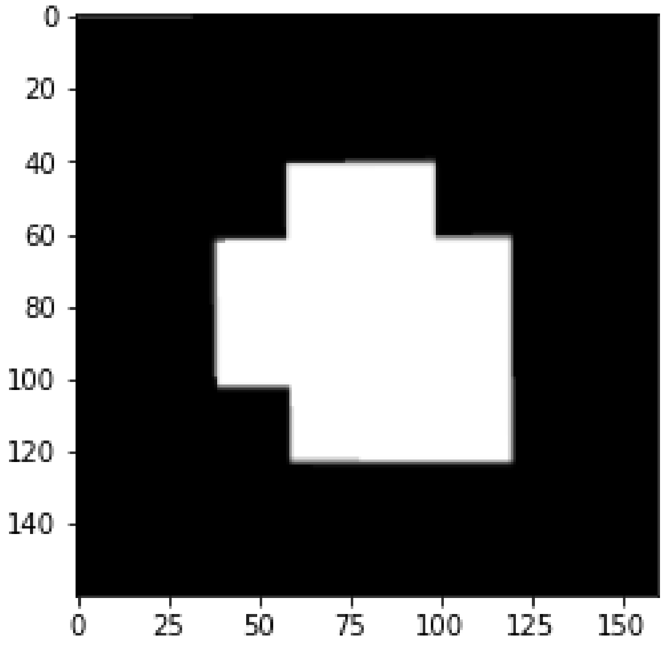
FFT Image:

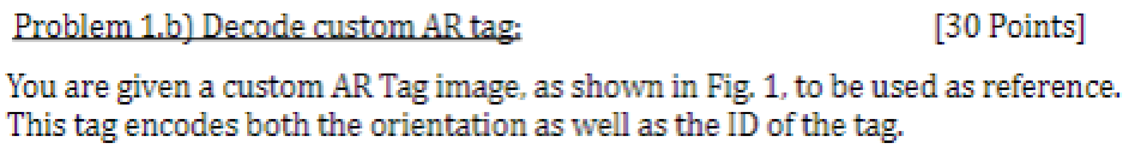


Original Image:



Warped Image





Problem 1.b) Decode custom AR tag: [30 Points]

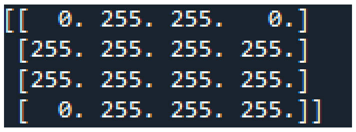
You are given a custom AR Tag image, as shown in Fig. 1, to be used as reference. This tag encodes both the orientation as well as the ID of the tag.

Process:

- After receiving only, the tag from the FFT method the tag is in the size 160x160
- Now I traverse through the tag in a 40X40 i.e., 4x4 grid format to get the grid which are white and grid which are black
- I are rotating the tag till I have the white region on the [3,3] in orientation.
- Compute the id of the AR tag.

Results:

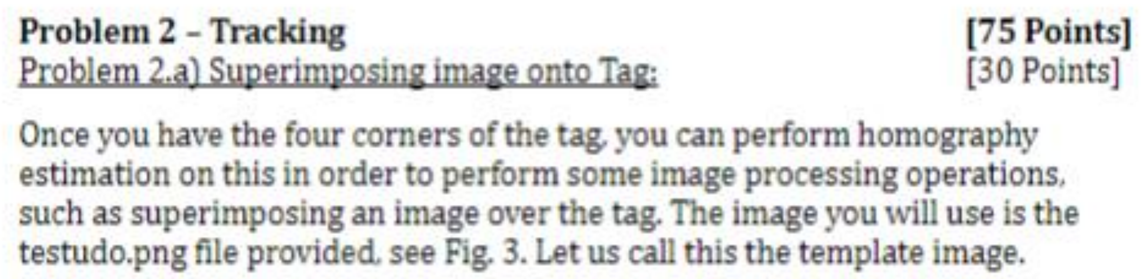
Orientation:



```
[[  0. 255. 255.   0.]
 [255. 255. 255. 255.]
 [255. 255. 255. 255.]
 [  0. 255. 255. 255.]]
```

AnsIr: 15

Question 2:



Problem 2 – Tracking [75 Points]

Problem 2.a) Superimposing image onto Tag: [30 Points]

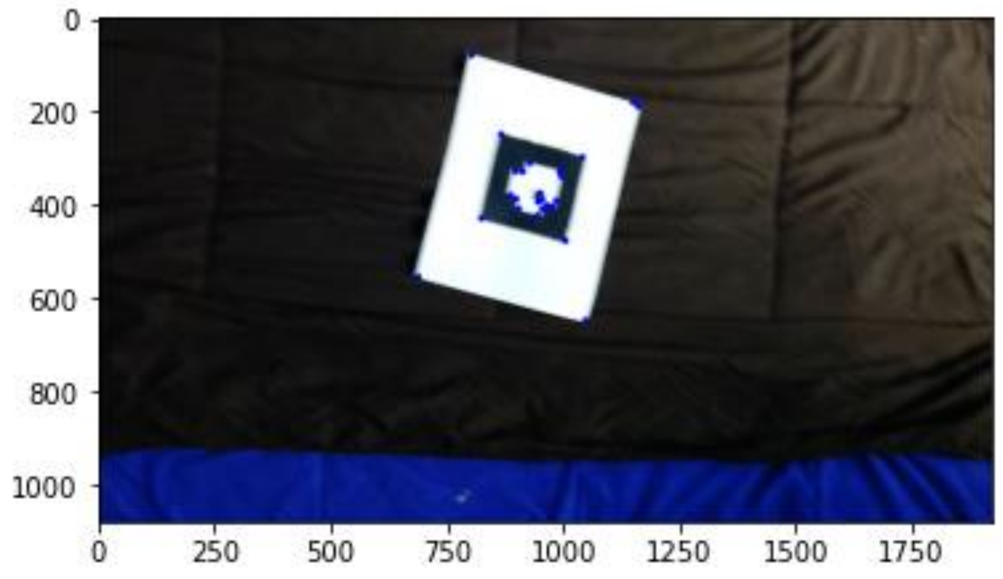
Once you have the four corners of the tag, you can perform homography estimation on this in order to perform some image processing operations, such as superimposing an image over the tag. The image you will use is the testudo.png file provided. see Fig. 3. Let us call this the template image.

Process:

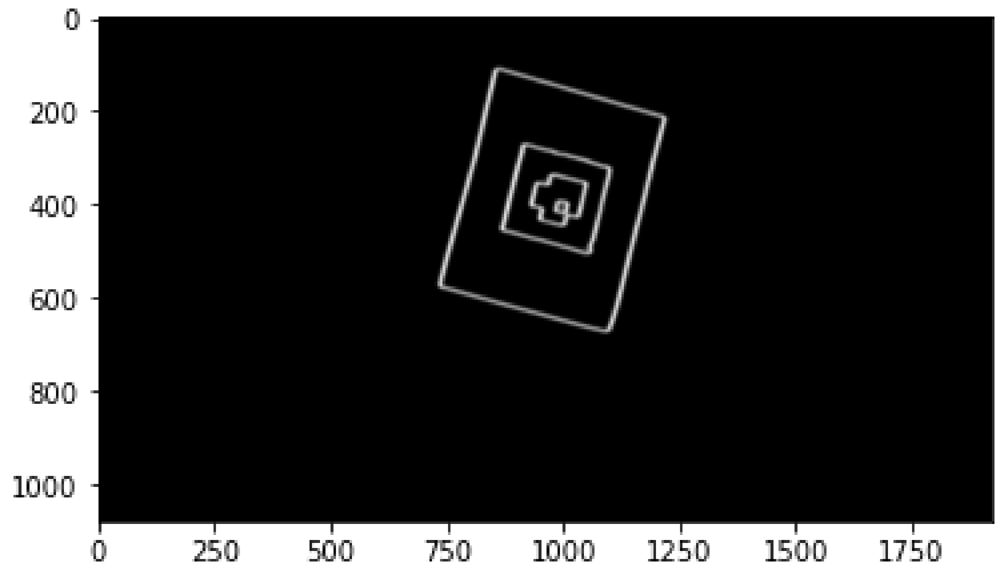
- After getting the FFT image I can compute the corners of the image using Shi-Tomasi.
- Then with the corner point of the tag and the paper using that I draw a polygon of paper.
- Segregate the points inside the paper and I get the corner points of the Aruco Marker.
- I rotate the marker corner according to the orientation of the marker to change the orientation of the Testudo when the video rotates.
- Then I compute the Homography warp the testudo image on the video frame.

Results:

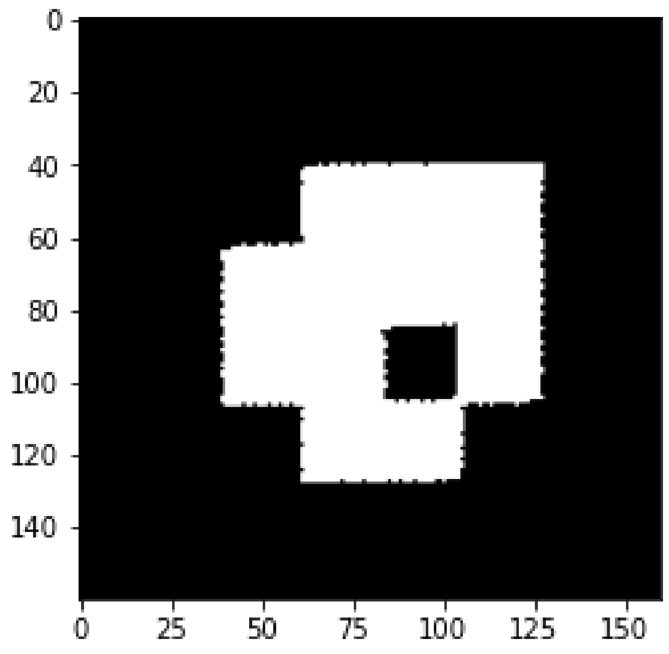
Corners Detected in paper and marker



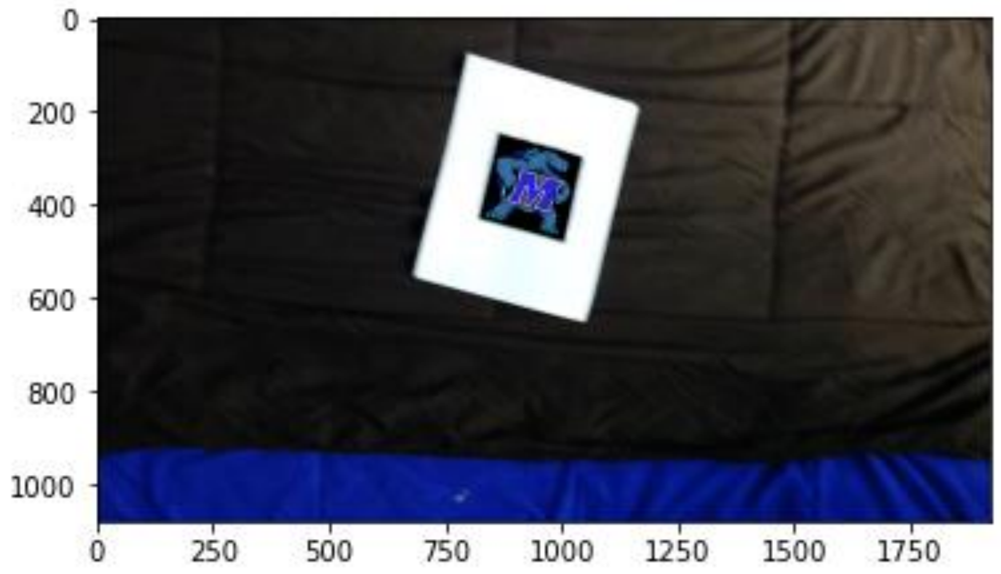
FFT Image

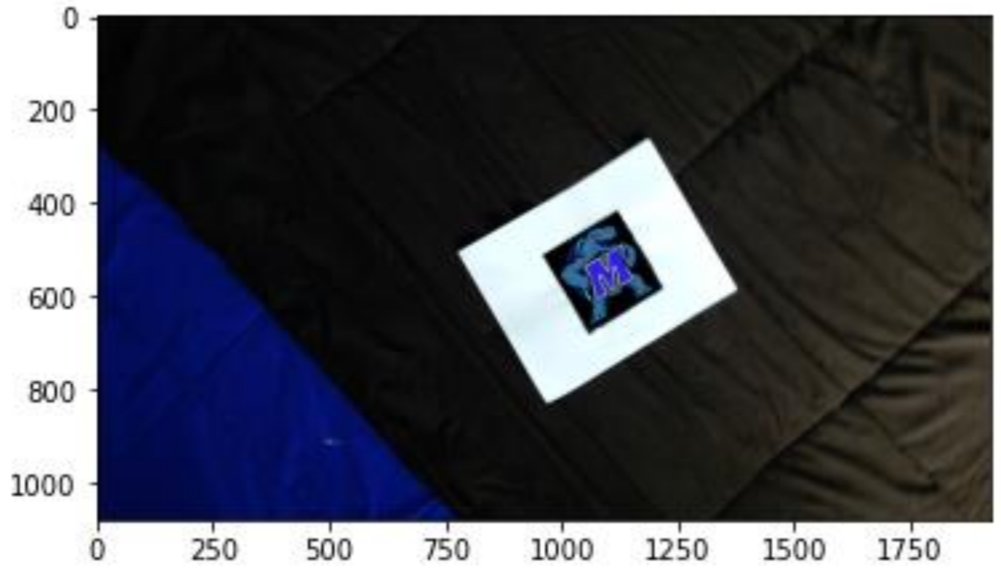


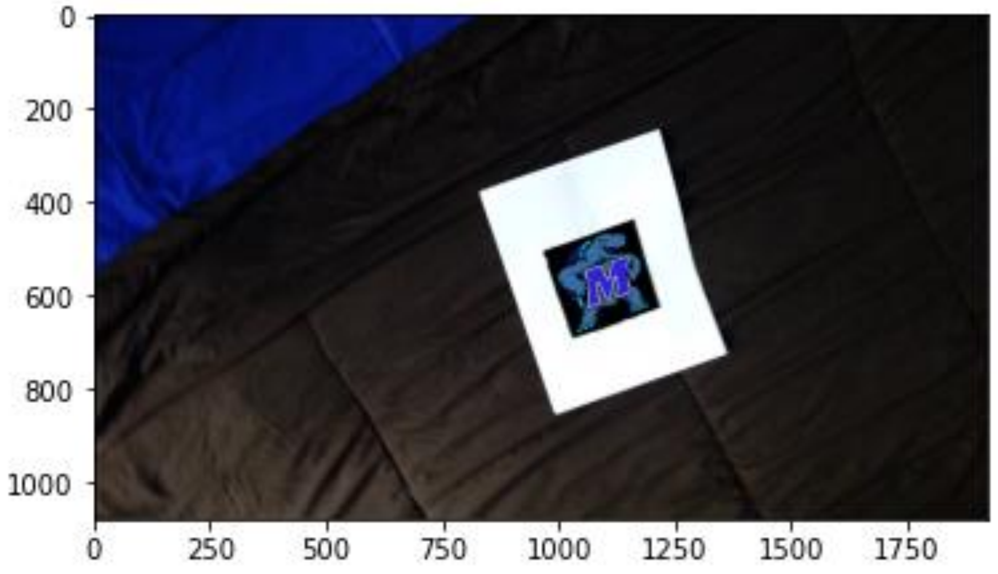
Marker



Testudo Warped Image:






0
200
400
600
800
1000
0
250
500
750
1000
1250
1500
1750

Problem 2.b) Placing a virtual cube onto Tag: [45 Points]

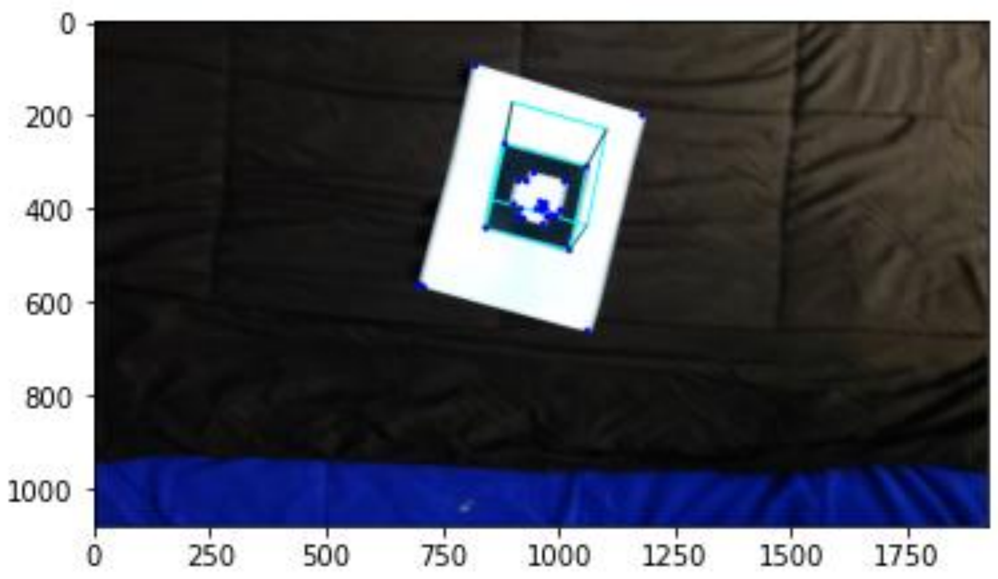
Augmented reality applications generally place 3D objects onto the real world, which maintain the three-dimensional properties such as rotation and scaling as you move around the "object" with your camera. In this part of the project you will implement a simple version of the same, by placing a 3D cube on the tag. This is the process of "projecting" a 3D shape onto a 2D image.

Process:

- After getting the FFT image I can compute the corners of the image using Shi-Tomasi.
- Then with the corner point of the tag and the paper using that I draw a polygon of paper.
- Segregate the points inside the paper and I get the corner points of the Aruco Marker.
- The cube has 3D coordinates, so the projection matrix has the shape 3x4. Since we have z coordinates in our case, we are finding a new homography matrix from the 2D homography and the camera calibration.
- Once the projection matrix is obtained, we then superimpose it to the video frame.

Results:

Cube superimposed to the video frame



Problem Faced:

The corner detection method used was Shi-Tomasi method the edges detected were not correct at times. So, for frames the image and cube is distorted.

Initially for get the marker corners, I thought we can remove the detected corners of the paper to detect the corners of marker, but it was inefficient, so polygon method is used as mentioned above.

When trying to write the video, it gets corrupted sometime because of the PC setup so cropped video is given.

Video Links:

https://drive.google.com/drive/folders/1Xf0D66QCERUT6iF5IlGOYSGDkZ-4na0U?usp=sharing

Question 3:

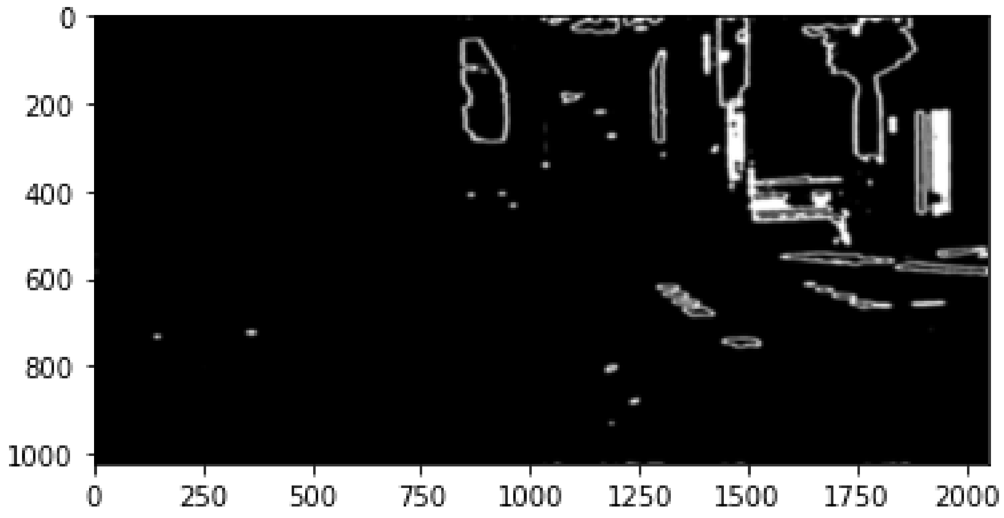
Extra Credit **[30 Points]**

You would have explored different types of masks to obtain gradients and their implementation with Canny edge detection. Dense Extreme Inception Network for Edge Detection (DexiNed) is a supervised deep learning model that performs edge detection. You need to install the Pytorch version of DexiNed and obtain predictions from it. Follow the README.md of the github repository for more details on installation . Describe your understanding of the network architecture and compare the predicted results with Canny Edge Detector's results. This link contains the images from *Cityscape* and *Kitti Dataset* that you need to run DexiNed with. You should be able to complete this without needing GPU resources since **you do not have to train the model again.**

DEXINED incorporates Convolutional Neural Network (CNN) and Deep Learning (DL) to predict the edges of the image. Since multiple images are trained using DEXINED a predict results that are substantially better than the conventional edge detection method. DEXINED trains with multiple input images and their edge detected outputs. Some of the images that are provided can have issue with annotation which results in a missing edge and the training DEXINED with that data will become difficult. Since it is using machine learning techniques DEXINED cannot detect all the edges clearly because it cannot be over trained which can lead to overfitting of data. DEXINED is also capable of detecting edges of image type which it has not been trained before but the efficiency of it will be reduced. As shown in the results below DEXINED provides a better performing output that conventional edge detection methods but still not 100% efficient.

Results:

Canny Edge Image:



DEXINED Image: