ENPM 673 Project 1

Param Dave
Master of Engineering in Robotics
University of Maryland, College Park
Email: pdave1@umd.edu

I. Problem1

A. Part A

This section of the project involved using Fast Fourier Transform to detect the edges. Convolution in spatial domain is equivalent to multiplication in frequency domain and vice versa. Leveraging this we develop high frequency binary mask i.e black circle in a white background and multiply it with a Fast Fourier Transform output of image taken from the first frame of video. After multiplying and applying inverse fast Fourier transform we get the edge map of the image. Finally cropping the image we get the April Tag outline. The output of the solution is given in figure 1. The size of the the circle for the mask does up to a certain doesn't affect the edge output as we shift all the lines in the FFT at origin.

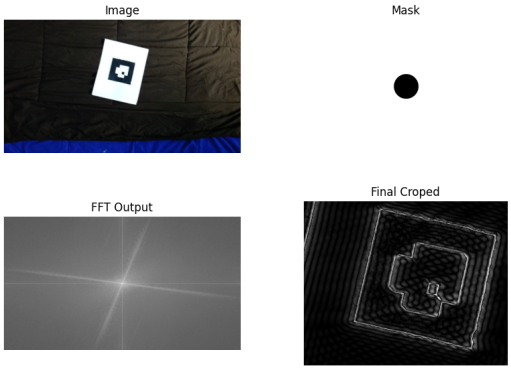

Fig. 1. Output Images of FFT

B. Part B

Here we detect and decode the April Tag. The simple solution is to reshape the image in a size multiple of 8, in our case (640,640). After reshaping we calculate median for ROI squares like corners of April Tag to check for orientation. We rotate the image until the image is as per our liking. We then check for median pixel value of the innermost 2x2 grids and decode accordingly. The output of the image after griding is shown in figure 2.

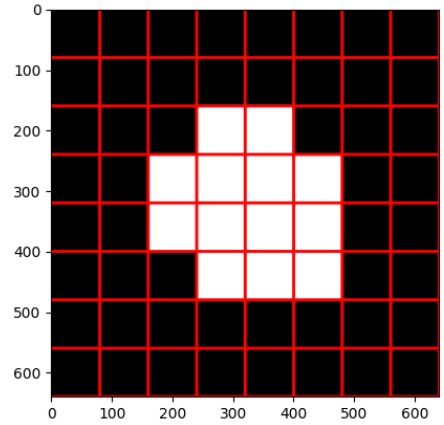

Fig. 2. Output Images of AR decode pipeline

II. Problem 2

A. Superimposing image onto Tag

This part of the project deals with superimposing an image over the tag. The image is shown in figure 3. The pipeline of this problem is shown in figure 4. As shown in the image first from video a frame is taken and a sequence of blur , grayscale and binary filter is used. This is the prepossessing part for edge detection. After this step using good features to track we select corners. This corners are then filtered to only give us a set of 4 corners corresponding to the 4 corners of April Tag. We compute homography between these corners and other 4 points on rectangle of a arbitary shape.We also try to take care of orientation of the tag. We apply wraping and resize the image.We finally conmpute Homography between April Tag points and Testudo and compute the inverse of it and wrap the testudo using the same ti get out output frame.

The problem we face is sorting the right corners for our pipeline. One approach may work on some frame and fail on other while one may work on others and fail on some. So it was not possible to detect the corners that would be constant for every frame. Other than that the pipeline was just too long for each frame an video took time to compute.

B. Placing a virtual cube onto Tag

In this subsection of problem we try to superimpose a cube on the AR Marker. The pipiline is shown in figure 5. We

Fig. 3. Testudo Image to be superimposed on AR Tag

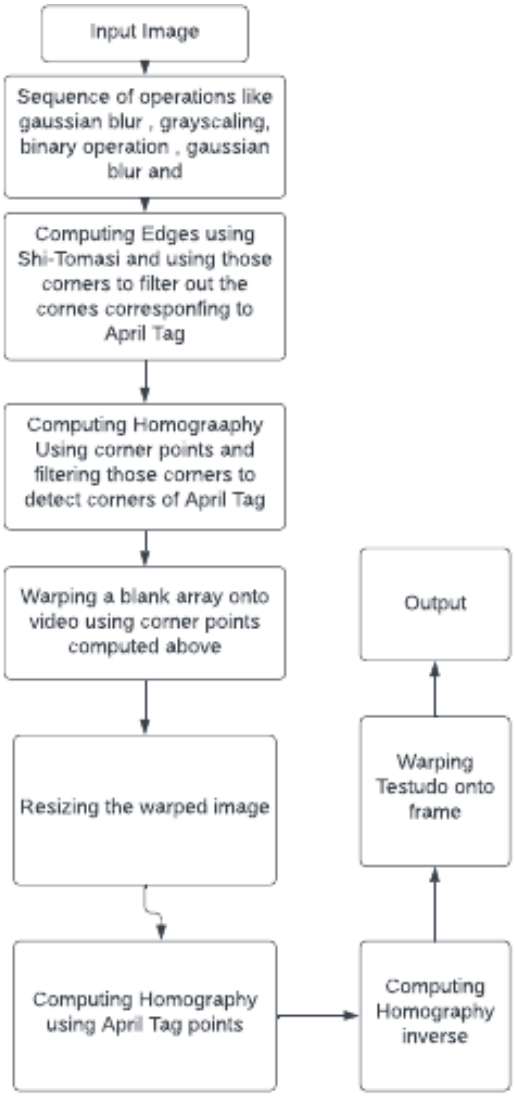

Fig. 4. Pipeline for Problem2 part a

apply same preprocessing steps to the frame and detect the edges of the April Tag. We compute the homography between the 4 point and point on the square. We use that Homography Matrix to compute the Projective Matrix which is a 3x4 matrix. Using projective matrix we compute 8 coordinated of a 3-d cube onto a 2-d plane. We draw lines between the points and get a cube.

A similar problem was faced for corner detection as same pipeline was used.

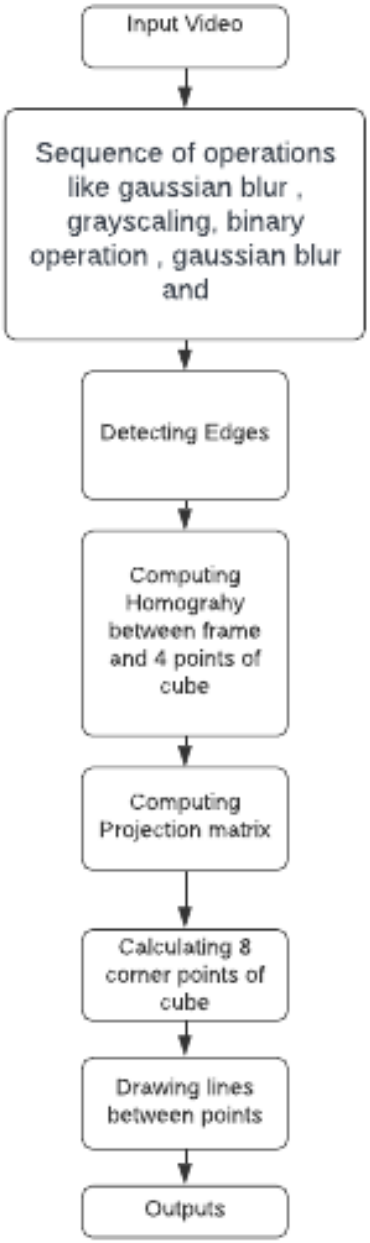

Fig. 5. Pipeline for Problem2 part b

III. DexiNed (Bonus Question)

The paper talks about edge detection using their neural network DexiNed based on VGG16 architecture. Paper addresses

the issue with state of the art architectures RCF, BDCN, and CATS which are trained on BSDS dataset which was created for the image segmentation purpose. The authors in the paper present their architecture along with a dataset BIPEDv2 that would better suit the edge detection problem.

The architecture is based on VGG16 CNN architecture. The Dexi architecture contains six blocks acting similar to an encoder. Each block is a collection of smaller sub-blocks with a group of convolution layers. The skip connections which are generally used in deep neural networks are used in this architecture. Skip-connections couple the blocks as well as their sub-block with each other. The feature-maps generated at each of the blocks are fed to a separate USNet to create intermediate edge-maps. These intermediate edge-maps are concatenated to form a stack of learned filters. At the very end of the network, these features are fused to generate a single edge-map.

USNet aka upsampling network (USNet) is a conditional stack of two blocks. Each one of then consists of a sequence of one convolutions and deconvolutional layer that up-sample features on each pass. A deconvolution increases the dimensions of the image and reduce the number of filters as opposed to the convolution operation.

Each sub-block in the constitutes two convolutions layers (the number of kernels is specified at the right side of blue rectangles). All kernels are of size 3 × 3. In the very first block, convolutions are with a stride of 2, hence in its name. Each convolutional layer is followed by batch normalization and a rectified linear unit (ReLU). From the Block 3 (light grey rectangles) the last convolutional, of the last sub-block, does not contain ReLU function. Rectangles in red are max-polling operators with a 3 × 3 kernel size and stride of 2. From the third block (Block-3) forward, the output of each sub-block is averaged with another skip-connection termed second skip-connections—SSC (green rectangles on the right side of Fig. 3). After the max-pooling operation, these SSC average the output of connected sub- blocks prior to summation with the first skip-connection—FSC,green rectangles on the left side. In parallel to this, the output of max-pooling layers is directly fed to subsequent sub- blocks. For instance, the sub-blocks of block-3 receive input from the first max-pooling; and the sub-blocks of block-4 receive input with a summation of the first and second max-pooling. The figure shows architecture of Dexined.

The outputs of canny and DexiNed are shown in figures respectively. It is clear the DexiNed outperform canny due to it being SOTA and due to custom BIPEDv2 it trained upon. There is more attention to details as a whole rather than just edges being places of intensity change.

IV. Conclusion

The superimposition algorithm can be improved upon to take less time to compute. While corner detection pipeline can be improved to be inline with the contour detection algorithm used in cv2.findcontour finction of opencv.

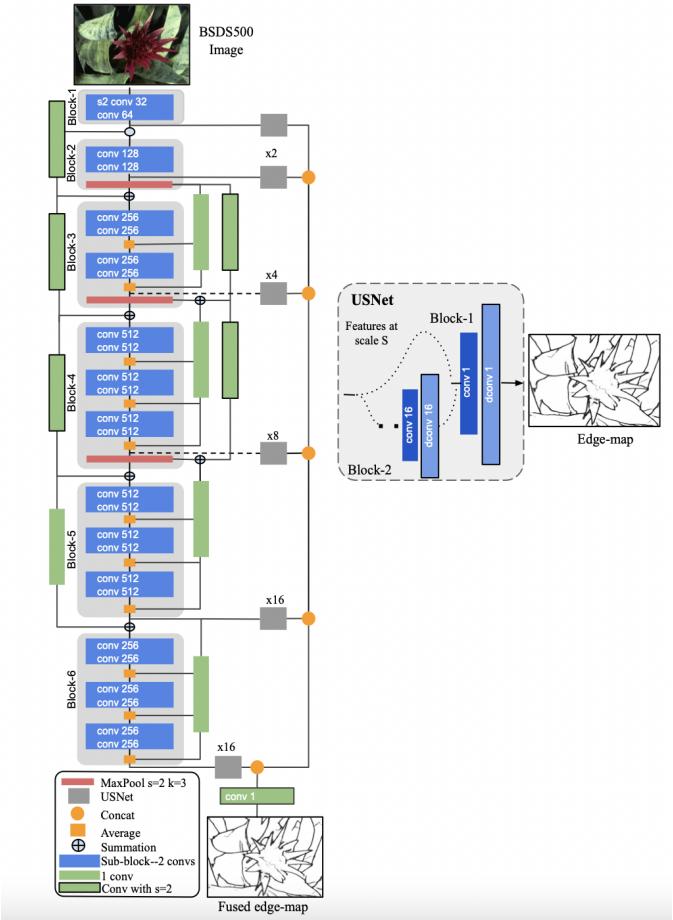

Fig. 6. DexiNed Architecture

Fig. 7. Input Image to DexiNed Cityscape

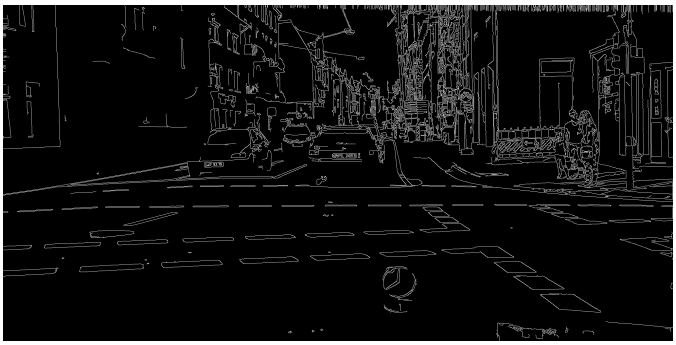

Fig. 8. Canny Output to Cityscape image

Fig. 9. DexiNed output to Cityscape image

REFERENCES

[1] X. Soria, Angel Sappa, and A. Arbarinia, "Dense Extreme Inception Network for Edge Detection," IEEE Computer Society, 2021.