-Jeffin Johny Kachappilly
UID : 118293929

Solution 1.a:

- The initial frame of the video was read and resized.
- As fourier transforms works only on grayscale image, the image was converted into grayscale.
- Inbuilt scipy function was used to do fourier transform and origin of image was shifted using ffshift.
- After converting into frequency domain, a square of size 100 was assigned value = 0 (i.e. High Pass Filter.)
- The high pass filtering was used to determine edges of paper and AR tag.
- After application of HPF, the origin was shifted back to initial position and inverse fourier was applied to change the image back into spatial domain.
- The figures displayed below represents the output:

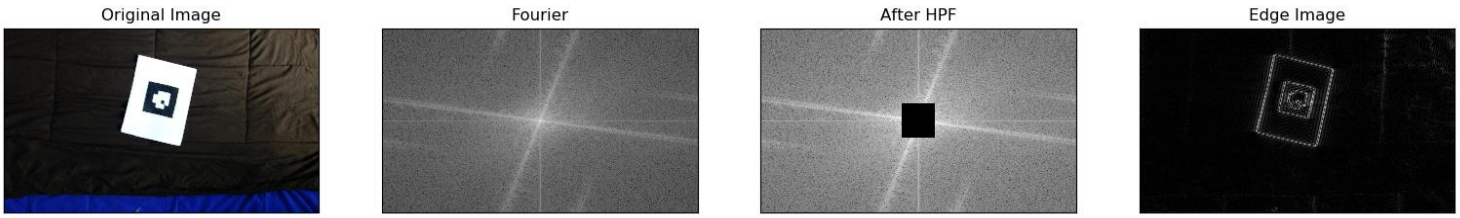

Fig. 1: Output from python script

Solution 1.b:

- The video is read, one frame after another.
- Corner Detection :
  - The image is converted into grayscale.
  - To get rid of noises outside the paper, a low pass filter (Gaussian Blurr) was applied.
  - Thresholding was done to make image into binary form.
  - As the inside of the AR tag wasn't important for corner detection of tag, the image was again smoothened out using Gaussian Blur.
  - After all the parameters of the functions were tuned out, Shi-Tomasi corner is implemented.
  - The maximum number of corners used in 'goodfeaturestotrack' takes into account the 4 corners of the paper, followed by corners of tag and some unavoidable corners inside tag.

- Function for choosing the required corners:
  - The mean of x coordinates and y coordinates were calculated.
  - The distance between these mean coordinates and other points are calculated.
  - 4 points with the largest distances were deleted. (These represent the points of A4 sheet).
  - Next the corners of the tag were determined using (argmin, argmax) for storing the points with maximum and minimum values of x and y coordinates.
- Function for determining the homography matrix between world and camera coordinates was defined.
- A blank image was created and the homography between the corners of the blank image and AR tag was calculated.
- The homography matrix is then multiplied with coordinates of blank image, which leads to the warping of the AR tag from video frame to blank image frame.
- Function for Tag ID determination:
  - Threshold is applied to the new image after homography.
  - After that, image is equally split into 8 squares.
  - Then median of each square is determined.
  - The position of the white square which is useful for determining the upright orientation of the tag is calculated.
  - The white square position has 4 possibilities (Fig 2) in blank image, for each possibility a new homography is performed by cyclically changing the corners of AR tag.
  - The above step will make the marker in upright position in the blank image.
  - After that the ID number is calculated, using the inner most 2 x 2 grid.

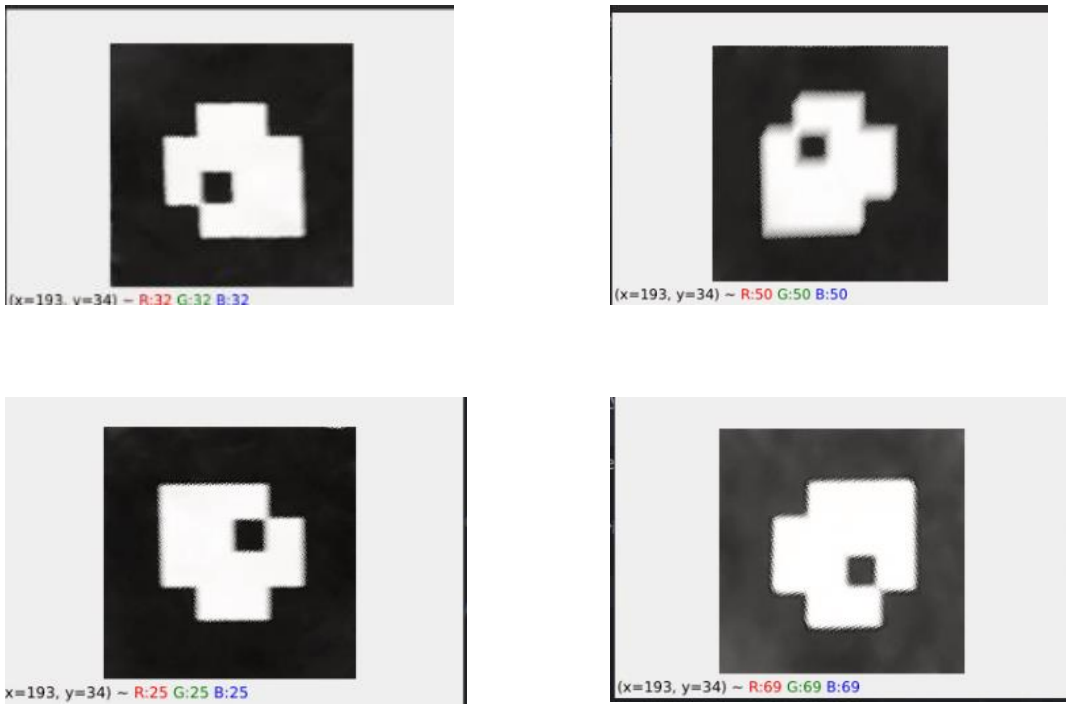

Fig. 2 : Different possibilities

CHALLENGES :

- Determination of the 4 corners of tag after Shi-Tomasi filter application:
  - As the number of points determined at any given time wasn't 8, the mean position of x and y coordinates would play around inside the tag. So the maximum distance method for getting 4 points was only applicable for A4 corners.
  - The solution to this challenge was using (argmin , argmax) for corner detection.
  - The (argmin , argmax) method fails and some points are lost when the AR tag's edges are parallel to the frame of the image.
  - To avoid this , a simple 'if' statement was made to check whether the given list of corners have 2 'Xmin' points. If true, some basic arithmetic operations were done to store the points.
- Getting the correct value of each square after 8 x 8 division of image:
  - In some frames, the AR tag wasn't perfectly warped in the blank image, which made checking for 255 (for white) in (5,5) position extremely difficult.
  - Np.median was used to work around this problem.

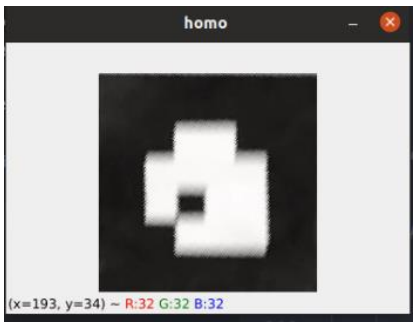

Fig. 3: Not perfectly dividable

- Some were corners of the Tag would miss at rare occasions and homograhy matrix would throw an 'out of index bound' error:
  - Solved it including "try: and except:" statements in the program.

VIDEO

Solution 2.a:

- *Same steps as of (1.b) before 'Function for Tag ID determination'.*
- The testudo image was read and its dimensions determined.
- After warping of AR tag to blank image, the image and AR corners were passed to another function to determine the orientation of the tag.
- The function splits the tag image into 8 x 8 parts and locates the position of white square.
- For the 4 possibilities of white square, the AR tag corners were cyclically changed so that the testudo image can be upright at all times.
- Homography matrix was determined with these new AR corners and testudo image corners.

- Then the homography for the testudo image was generated.

CHALLENGES :

- Some frames were not rotating the testudo as expected. (Still couldn't tackle this issue).

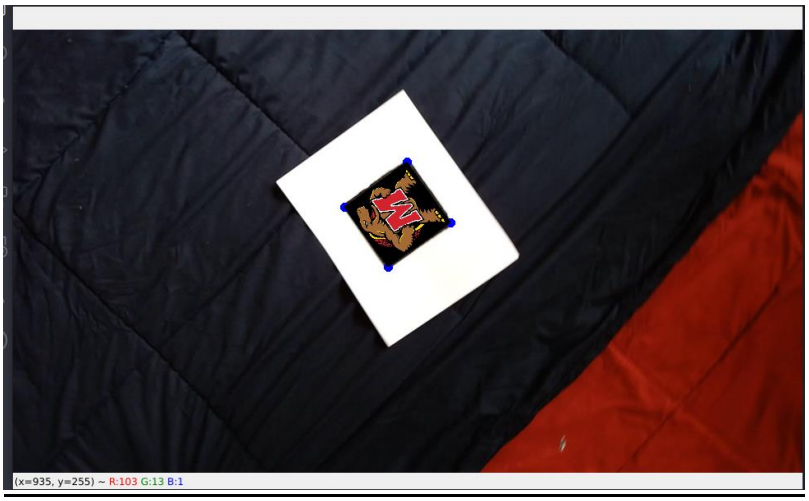

Fig. 4 : Testudo template

VIDEO

Solution 2.b:

- The video is read, one frame after another.
- Corner Detection :
  - The image is converted into grayscale.
  - To get rid of noises outside the paper, a low pass filter (Gaussian Blurr) was applied.
  - Thresholding was done to make image into binary form.
  - As the inside of the AR tag wasn't important for corner detection of tag, the image was again smoothened out using Gaussian Blur.
  - After all the parameters of the functions were tuned out, Shi-Tomasi corner is implemented.
  - The maximum number of corners used in 'goodfeaturestotrack' takes into account the 4 corners of the paper, followed by corners of tag and some unavoidable corners inside tag.
- Function for choosing the required corners:
  - The mean of x coordinates and y coordinates were calculated.
  - The distance between these mean coordinates and other points are calculated.
  - 4 points with the largest distances were deleted. (These represent the points of A4 sheet).
  - Next the corners of the tag were determined using (argmin, argmax) for storing the points with maximum and minimum values of x and y coordinates.

- Function for determining the homography matrix between (x,y) coordinates of unit square and corner coordinates of AR tag was defined.
- Using this homography matrix(H), and (K) intrinsic matrix, the projection matrix is determined using the formulae from supplementary material
- This projection matrix is then passed into another function for calculating the coordinates of upper surface of the cube in image frame.
- Then lines are plotted using cv2.line command between these points.

CHALLENGES :

- In some frames the top square was not displaying properly. (Still haven't found solution).

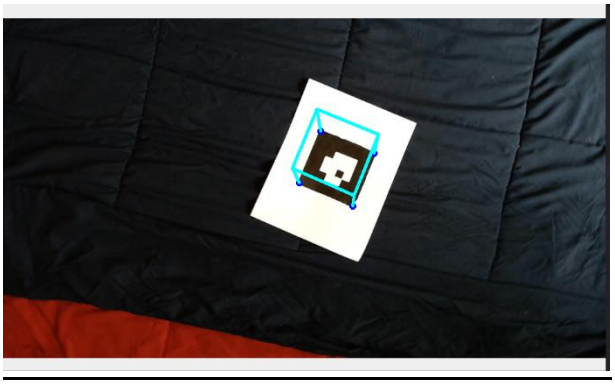

Fig. 5 : Cube generated

VIDEO

EXTRA CREDIT :

DexiNed is designed to allow an end-to-end training without the need to weight initialization from pre-trained object detection models, like in most of Deep Learning based edge detectors.

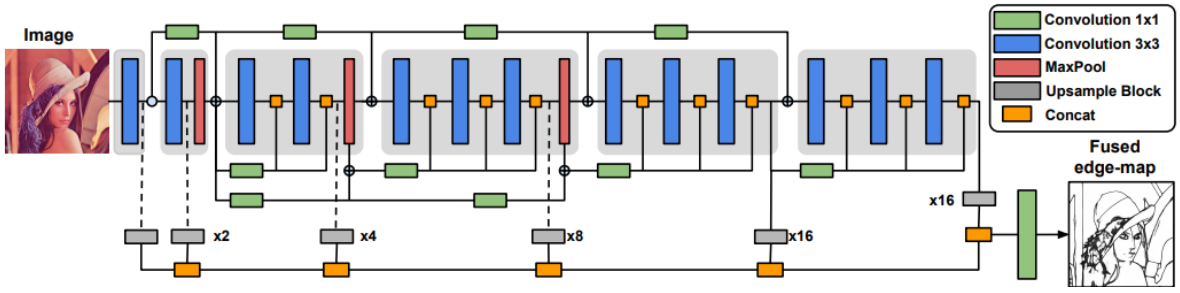

Fig. 6 : Proposed Architecture

Architecture:

- Dense Extreme Inception Network, consists of an encoder composed by six main blocks (showed in light gray). The main blocks are connected between them through 1x1 convolutional blocks.

- Each of the main blocks is composed by sub-blocks that are densely interconnected by the output of the previous main block.
- The output from each of the main blocks is fed to an upsampling block that produces an intermediate edge-map in order to build a Scale Space Volume, which is used to compose a final fused edge-map.
- All the edge-maps resulting from the upsampling blocks are concatenated to feed the stack of learned filters at the very end of the network and produce a fused edge-map..
- The blocks in blue consists of a stack of two convolutional layers with kernel size 3 × 3, followed by batch normalization and ReLU as the activation function (just the last convs in the last sub-blocks does not have such activation). The max-pool is set by 3 × 3 kernel and stride 2.

Upsampling Block:

The UB consists of the conditional stacked sub-blocks. Each sub-block has 2 layers, one convolutional and the other deconvolutional; there are two types of sub-blocks. The first subblock (sub-block1) is feed from Dexi or sub-block2; it is only used when the scale difference between the feature map and the ground truth is equal to 2.

The other sub-block (sub-block2), is considered when the difference is greater than 2. This sub-block is iterated till the feature map scale reaches 2 with respect to the GT. The sub-block1 is set as follow: kernel size of the conv layer 1 × 1; followed by a ReLU activation function; kernel size of the deconv layer or transpose convolution s × s, where s is the input feature map scale level; both layers return one filter and the last one gives a feature map with the same size as the GT.

The last conv layer does not have activation function. The subblock2 is set similar to sub-block1 with just one difference in the number of filters, which is 16 instead of 1 in subblock1. For example, the output feature maps from block 6 in Dexi has the scale of 16, there will be three iterations in the sub-block2 before fed the sub-block1.

Fig. 7: DexiNed

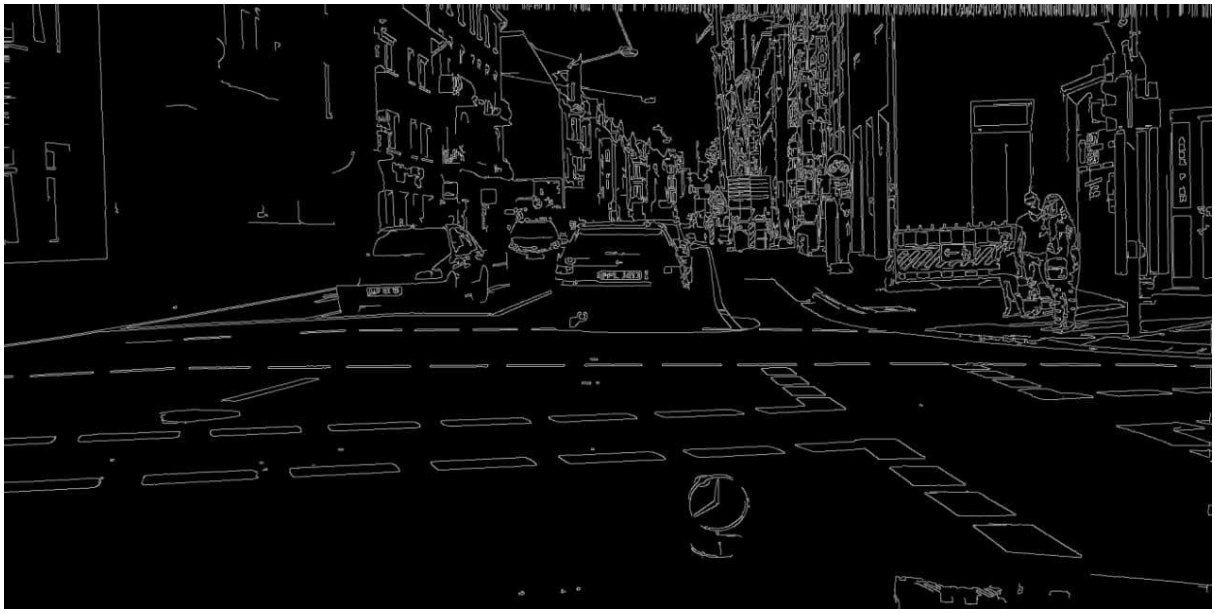

Fig. 8 : Canny Edge

Even though canny edge detection provides thinner edges, the output from Dexined is more preferred. Fig.8 has more details than dexined but some of the vital information such as cars bonet and rear bumper is indeterminant from fig.8 . The clear bifurcation of objects makes the dexined output much useful for computer vision application for this scenario.