Queen Mary
University of London

Introduction to Computer Vision

Coursework

Submission

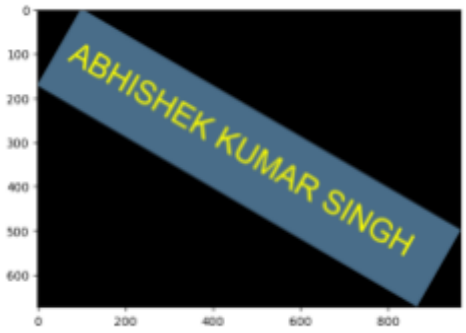
Name: Abhishek Kumar Singh

Student number: 220982845

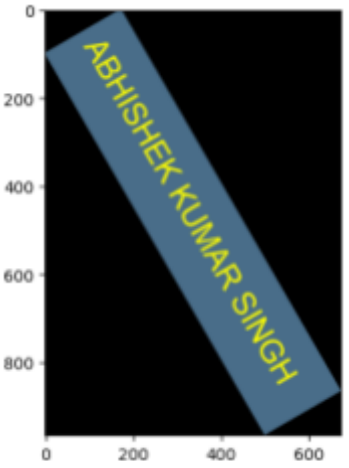
Transformations
Question 1(b):

Rotated images:

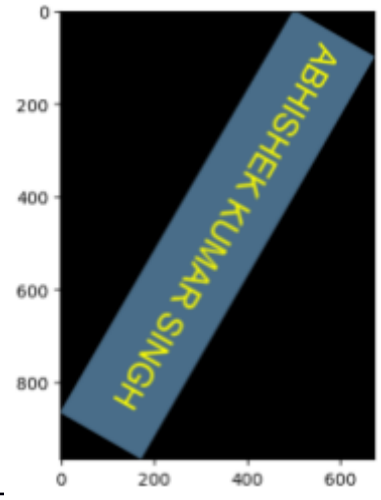
θ = 30 deg



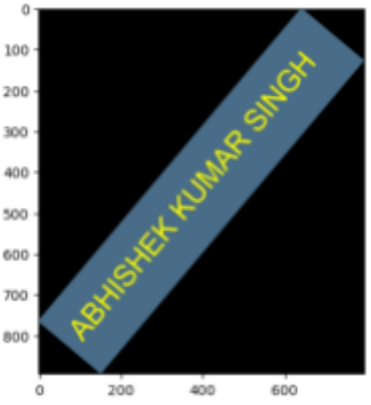
θ = 60 deg



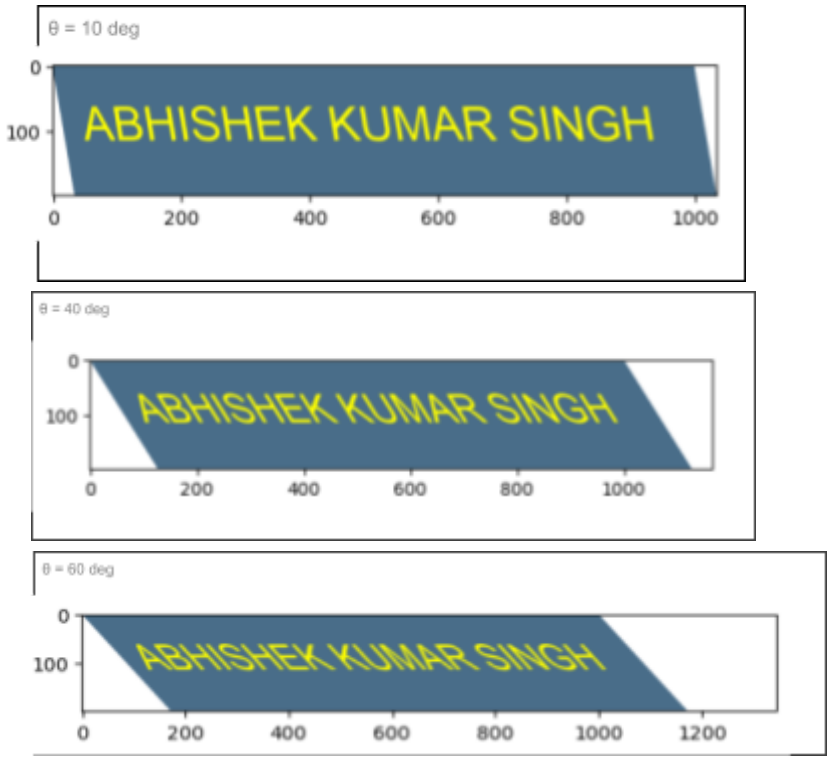
θ = 120 deg



θ = -50 deg



Skewed images:



Your comments:

Understanding:

The first task of Submission 1 covers the topic of Geometric Transformations. The task challenges us and motivates us to refrain from using any of the built-in functions for geometric transformations in PIL or OpenCV, helping us bolster our understanding of the underlying mathematical operations during these image transformations.
We will be developing these functions from scratch and operating on individual pixels of the image and rotating and shearing the image by creating a transformation matrix and observing how the order of these operations affects the image.

Analysis:

i) The first part of the task asks us to input an image and rotate it by a given angle and then skew the rotated image horizontally with a different angle. In the problem, Firstly we have to manipulate the axis of rotation for which we will be translating the image's origin to the center point of the image. We will then rotate the image about this point and then after rotation, we have to translate it back to the original origin of the image.
For rotation, we need to take the pixel value at every (x,y) location, rotate it according to the given angle, and then write these values of the new location with respect to the origin we assumed in the new image, which will be our new rotated image.

We also need to find out the new height and width of the image after rotation, since the frame's dimensions would change after the rotation so we use basic trigonometry to give us the new coordinates (polar) in the newly rotated frame.

ii) We will use ImageDraw and ImageFont libraries in PIL to write our name on the image and then we have to rotate and skew the image with our name by the given angle.
After rotation and shearing, we notice that there are certain artifacts and holes in the images. This is because when we round off the values after multiplications with sines and cosines, this could lead to some locations being indexed more than once when writing these address locations onto the new image and sometimes they are missing which leads to holes. To solve this problem, we need to adopt Inverse Mapping Technique.

Forward Mapping

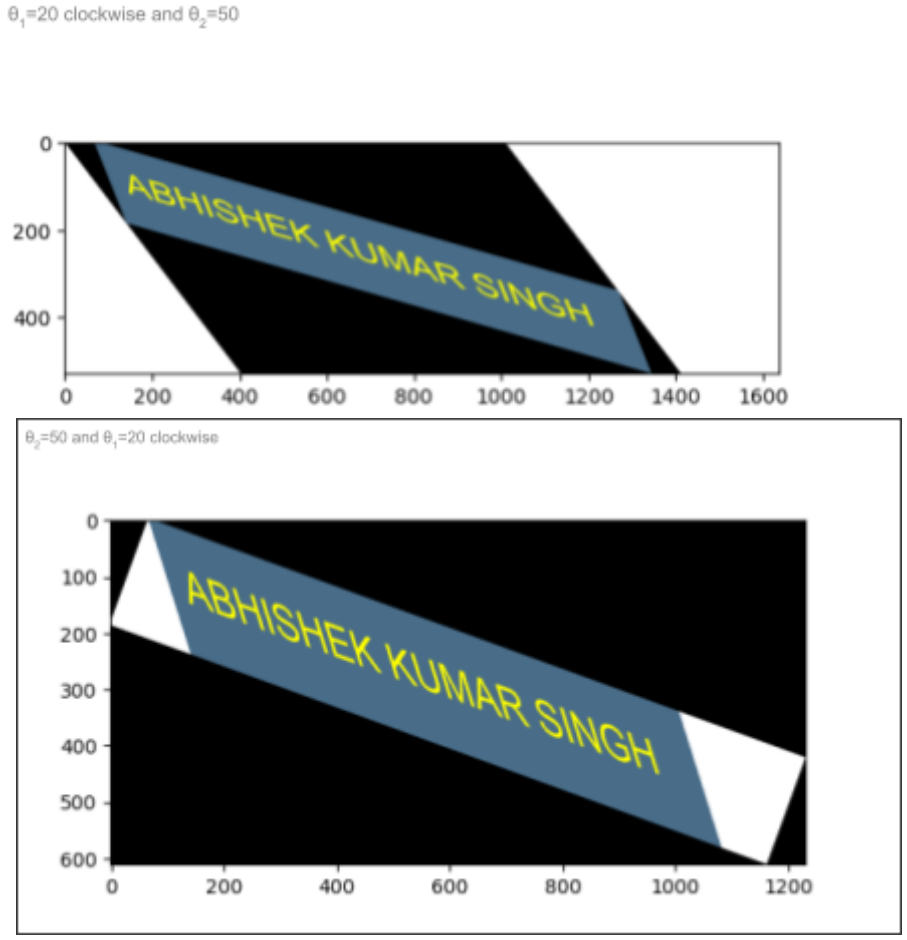
This technique consists of scanning all the pixels in the original image and then computing their positions in the new image. We apply the transformations on pixels of the original/source image, then we take this transformed matrix and copy these pixel values onto the output pixels. This is called forward mapping.

The main disadvantage with forward mapping is that there could be some output pixels that did not receive any input image pixels, which leads to holes. There could also be some output pixels that received more than one input image pixel, which leads to artifacts

Inverse Mapping

This technique scans each pixel of the new image and applies the mapping to find the pixel in the original image, after which we need to use some interpolation techniques on the surrounding pixels to compute the new intensity.
We use inverse mapping to overcome the two main disadvantages of forward mapping.

Question 1(c):



Your comments:

This problem shows how the order of the operations affects the resultant image, in this, we take the image and first rotate it by 20 degrees and shear it by 50 and compare it with another image which is the resultant of the shear first and then rotates by the same angle. We observe that both the resultant images are not the same, that is because these transformations are essentially matrix multiplications. We know that matrix multiplications are not commutative. Hence, when we change the order of operations on the image, i.e when we apply skew before rotation we are basically performing a series of matrix multiplication of the image with the shearing matrix and then multiplying the resultant with the rotation matrix. If we change the order we would be getting very different results.

Challenges:

The major challenge in this problem for me was to gain an understanding of the coordinate system of the image and then apply a bounding condition for mapping the results back to the original image. Initially, In the rotate and the shear function, I used forward mapping in which my index was going out of bounds for the x and y-axis.

Another difficulty I faced was understanding in what order the height and width of the image are stored in the image shape. I was confused if the coordinates of the image would be y, x (Height, Width) or, x, y(Width, Height).

Another challenge I faced during this task was to make sure the image does not get cropped/cut and goes out of the dimensions of the image window.
For that, I had to adjust the new height and new width after the transformation accordingly so the image fits properly in the new frame.

I was incorrectly translating the origin of the image to the actual center of the image. Instead of taking the translation and the rotation step by step, I constructed an incorrect matrix that translates the image's coordinates to the center and then rotates the image about the point.

Mistakes:
There were a lot of mistakes that I made during the formulation of this code, but I gained meaningful insights from this and gained a better understanding of the concepts.

One of the major mistakes I made was considering the order of image coordinates to be (x,y) instead of (y,x), this mismatch between my first loop and second loop and mapping at the later stages was causing the index to go out of bounds for x.

The second mistake I committed was in the order of matrix multiplication, I was not using the proper order for matrix multiplication for translation and rotation, which was giving me incorrect results it wasn't until I did the entire math on a pen and paper I realized where I was going wrong.

Discoveries:

The discoveries I made along the way mainly helped me realize how important the clarity of what we are doing mathematically is before actually sitting down to code.

I got clarity on the basic concepts of image transformations and cropping of the image by manipulating their coordinates. For eg., the midpoint of the rotated image would no longer be height//2 and weight//2, but they will become new_height//2 and new_width//2 which is their new heights and width w.r.t to the rotation/shear frame.

I discovered what the disadvantages of forward mapping are and how we overcome those shortcomings by using inverse mapping.

I also discovered that the order of the operation of transformation on an image leads to different results, this is because essentially we are doing matrix multiplications that do not follow the property of commutativity.

Convolution
Question 2(b):

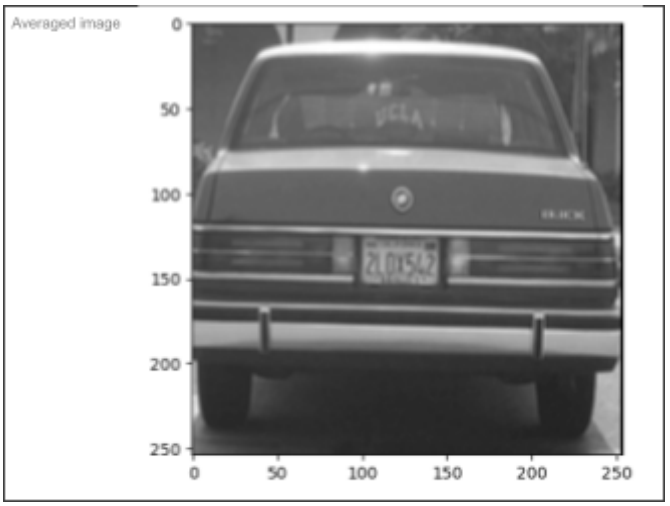
Designed kernel:

Unnormalized Kernel

```
([[1 ,1 ,1],
 [1, 1 ,1],
 [1 , 1, 1]])
```

Normalized Averaging Kernel

```
[[0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111]]
```



Your comments:

Understanding:

The Second question of the coursework introduces us to convolutions which is a mathematical operator that expresses the amount of overlap of one function ‘g’ as it is shifted over another function ‘f’. In 2D convolution, we move a small matrix called Kernel over 2D Image and multiply it element-wise over each sub-matrix, then sum elements of the obtained sub-matrix into a single pixel of the Feature map. We move it from the left to the right and from the top to the bottom. At the end of convolution, we cover the whole image surface.

Analysis:

a) For simplicity, we convert the input image into grayscale before applying the convolution operator to it. Since we know convolution is an operator that multiplies the sub-element of a matrix with a kernel and sums the elements. This would reduce the size of the resultant image. Hence, we have to find what the shape of our resultant image becomes after convolution.

It can be easily calculated using the formula:

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

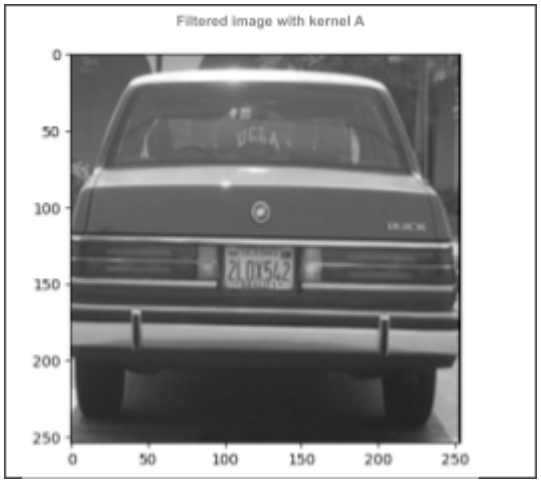
n_{in}: number of input features
n_{out}: number of output features
k: convolution kernel size
p: convolution padding size
s: convolution stride size

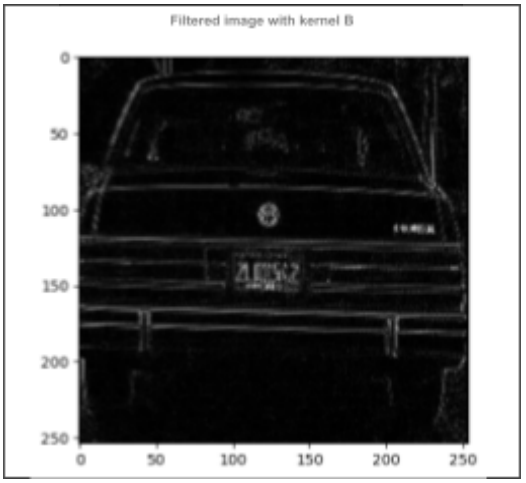
Next, we have to flip the kernel, we have to flip the kernel both about the x-axis and y-axis. It is really necessary to flip the kernel because if we don't flip it before performing the operation we would end up calculating the cross-correlation between them instead of convolution.
Another important step is to normalize the kernel before convolving so that this will ensure the average pixel in the modified image is as bright as the average pixel in the original image.

For convolution, we create small windows and multiply kernels and the windows together. After experimenting I realized, there was a need to take the absolute value of the convolution result because after multiplication with the kernel certain values. I also noticed that with certain kernels I was getting really overexposed, white images. This meant that after convolution values were going over 255, to solve this we have to use NumPy clip so the values fall between the range 0 to 255.

b) The second part asked us to average the image and apply a sharpen filter. The average filter used is also called a box blur.
After applying the average filter on the image, each pixel would become the average of the surrounding pixels. The overall effect of averaging the values is to make the individual features of an image less noticeable.

Question 2(c):

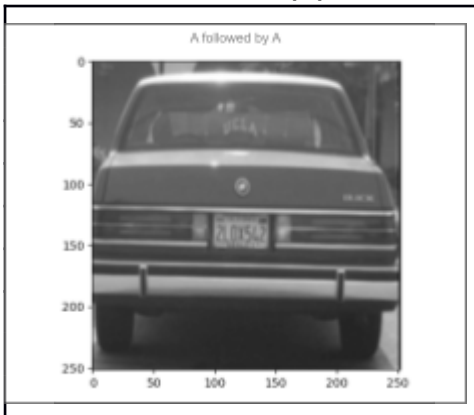


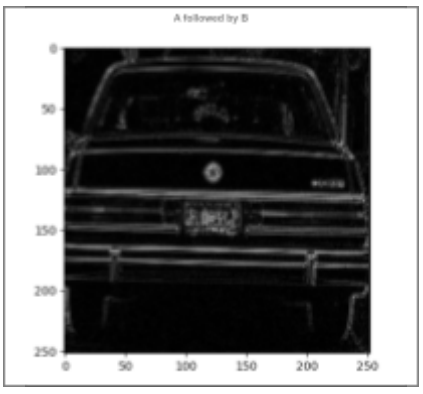


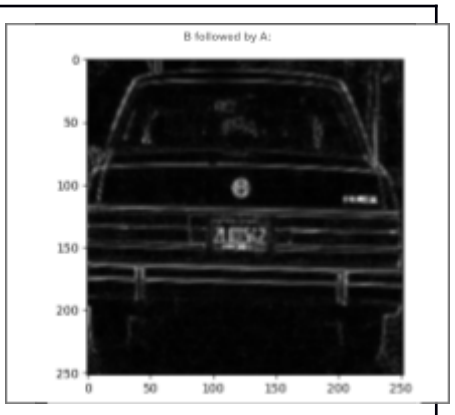
Your comments:

c) In this part, we have two kernels A and B. Kernel A is a smoothening filter, which is also called the Gaussian filter. It is used to `blur' images and removes detail and noise.
Kernel B is an edge-detection kernel which is also called a Laplacian operator. The effect of this kernel is that the central pixel value is multiplied by 4 and subtracted while the surrounding pixel values are added.

Question 2(d):







Your comments:

d) In this part, we will observe the effect of having multiple convolutions on the same image by varying the order or sequence of the convolution. In the first part, we convolve the image twice with the Gaussian filter, and we get an even more blurred output.

Next, we convolve the image with kernel A and then followed by kernel B. In this result as a consequence of blurring the image, the noise is lost and we can detect edges better using kernel B.
Lastly, we convolve the image with kernel B and then by Kernel A, in this case, we will first detect edges, and then perform gaussian blurring.

Both the results of A then followed by B and B followed by A are not the same. This shows us that the order of convolution gives us different results. This is because the pixel values change after the convolution operation at various pixel locations according to the kernel.

Challenges:
The challenge I faced during the task was in converting the image to grayscale, after using a weighted average over the color channels, I still got a green image. I realized this was due to matplotlib having various colour maps. With the grayscale colour map, I got perfect results.
Another challenge I faced was the values of my pixel values after convolution was going beyond 255, or were negative.

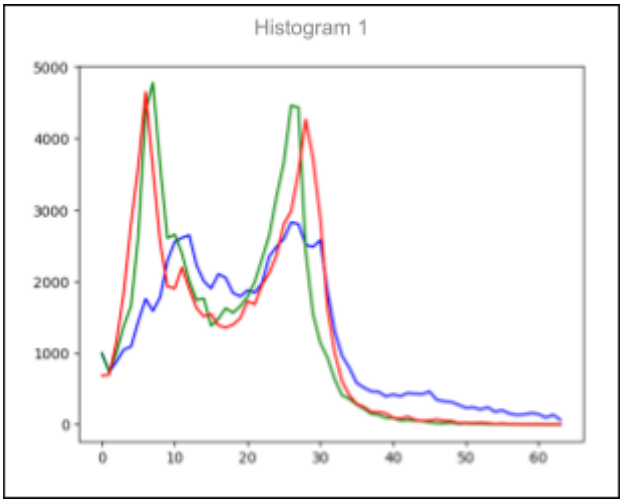
Mistakes:
Initially, I was using an RGB image for this problem, but I later switched to Grayscale for better results. I didn’t clip the convolution values, I also didn't take the absolute values of the convolution. This gave me a lot of errors while experimenting with different kernels.

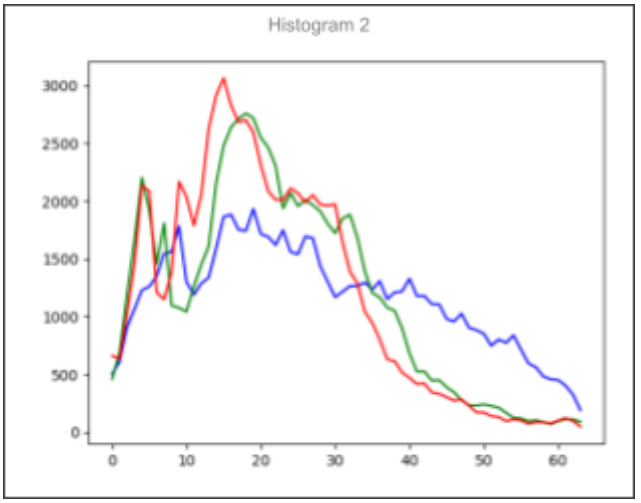
Discoveries:
I discovered how cross-correlation and convolution coincide with each other if the kernel is symmetrical.
I also discovered how various kernels/filters when convolved together with an image give us such different results and how most of the editing filters we use for image editing are essentially convolutions.

Histograms
Question 3(a):

Two non-consecutive frames:

Corresponding colour histograms:





Your comments:

Understanding:

The third question for this assignment asks us to learn about colour histograms, understand how binning works and histogram intersection can be used to find a scene change in a video.

Analysis:

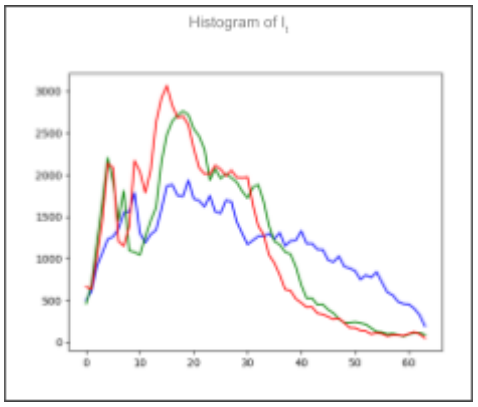
For generating a histogram we first need to do binning, for that we need to count the number of pixel values in each channel in the image and make a dictionary using these values.

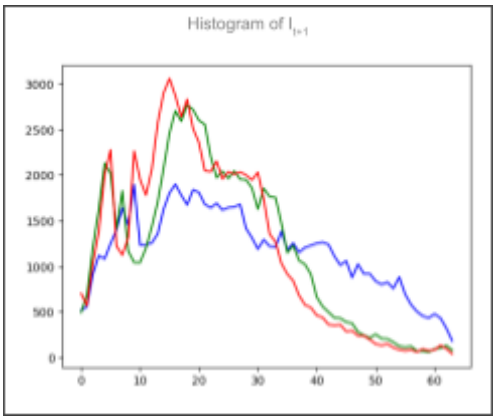
Using the bins we created and the dictionary we create a histogram for each respective channel. The above function is then used to create a histogram for every frame in the video. We extract all the frames from the video and then append them to an array, after which we loop over every frame in this array and calculate their histogram.

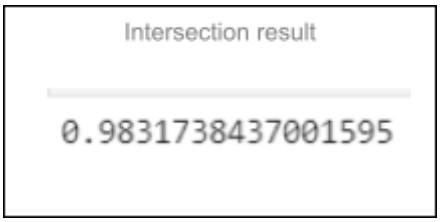
Question 3(b):

Two consecutive frames:

Histograms:

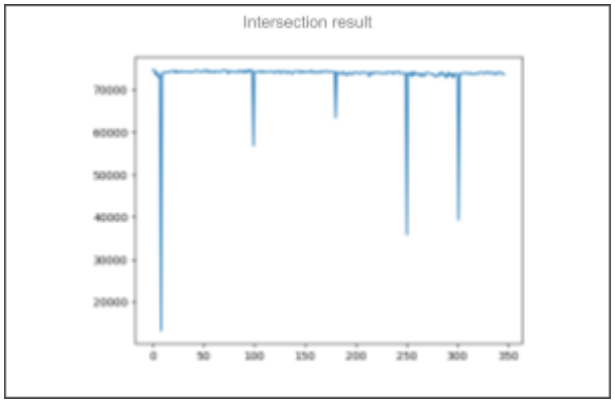


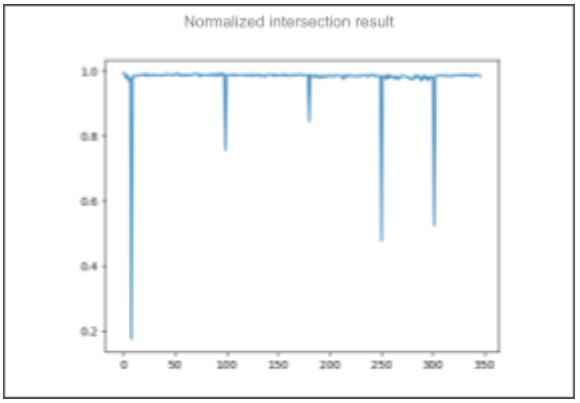




Question 3(b):

Intersection result for a video sequence:





Your Comments:

b) The intersection of a histogram is given by taking the minimum of the two histograms and taking a sum over the total number of bins. After this, we will go over the frames array and compute the histograms for every i^{th} and i^{th+1} (consecutive frames) and take their intersections.

We observe after taking an intersection that there are 5 different scene changes in this video. The intersection returns a singular value which tells us the amount of overlap between each consecutive frame.

We notice that there is a need for normalization for the histograms, hence we will divide the intersection of the histograms by the sum of the model(Second histogram).

After normalization, we observe that there is only a change in the scale of the plot of the intersection. The scale after normalization is reduced from 0 to 1.

Question 3(c):
Comments:

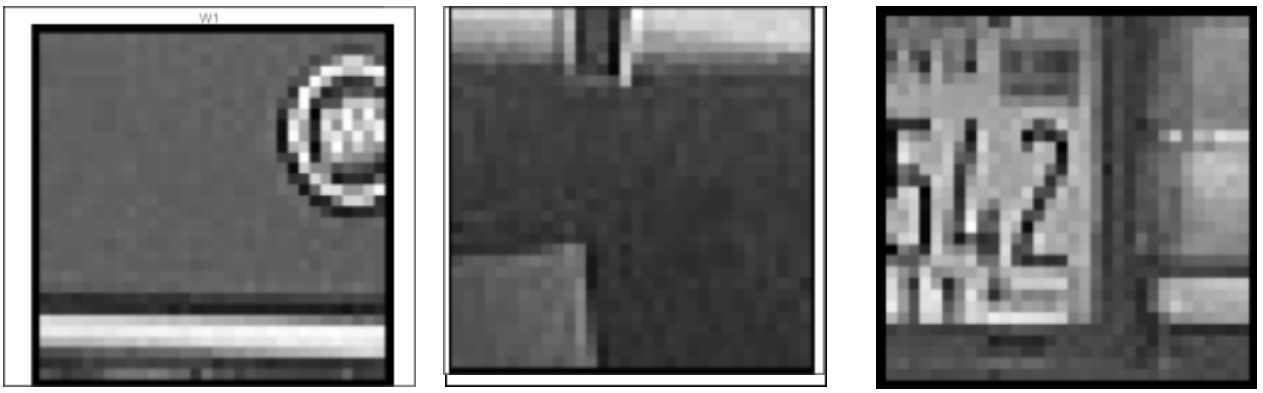
c) The intersection of the histograms for the respective frames gives us information about the scenes in the video and when the scene drastically changes in the video which is represented by a sharp drop in the plot.
Yes, we can use the histogram intersection to comment on the scene change in this video but in most cases, this method is not very robust.
The histogram is a many-to-one mapping, hence very different scenes could have the same colour histograms.
There is a high chance of faulty scene change detection if we just rely on histogram intersection to observe a scene change.

Challenges:
The challenge I faced during this task was mostly in figuring out the way to generate a histogram and do binning.

Mistakes:
The order of colour channels I used was [r,g,b] instead of [b,g,r], even though we were reading the images using OpenCV

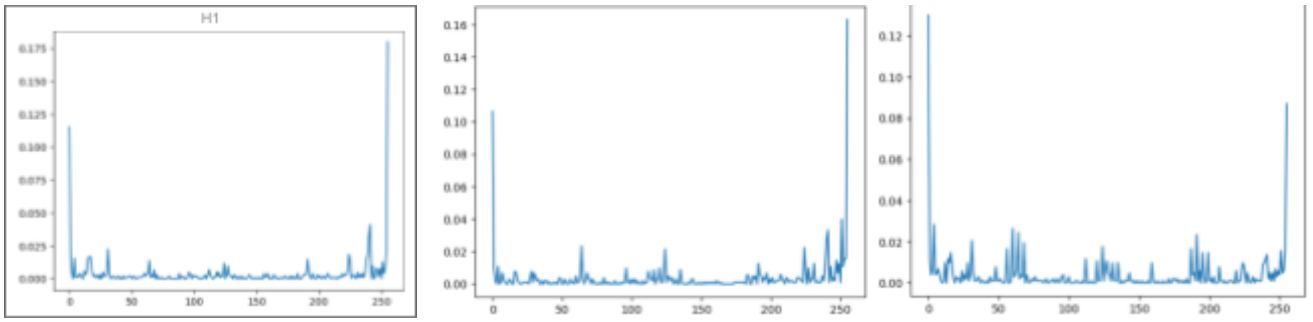
Texture Descriptors and Classification
Question 4(a)

Three non-consecutive windows



LBP of windows

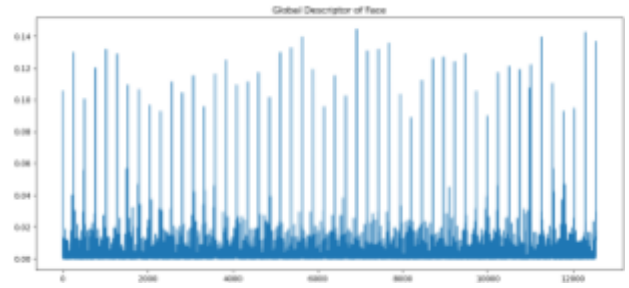
Histograms of LBPs



Question 4(b)

Two example images:

Descriptors:



Your comments:

Understanding:

The fourth question for this assignment asks us to perform texture classification using the Local Binary Operator which is a good measure of texture analysis in an image. It goes over the image in a 3*3 window and calculates a binary value using the center pixel value and comparing it with its 8 neighbours and then converting the binary value to decimal gives us the features in the image. With the features created by the LBP texture operator, we can tell the texture of the objects in image

Analysis:

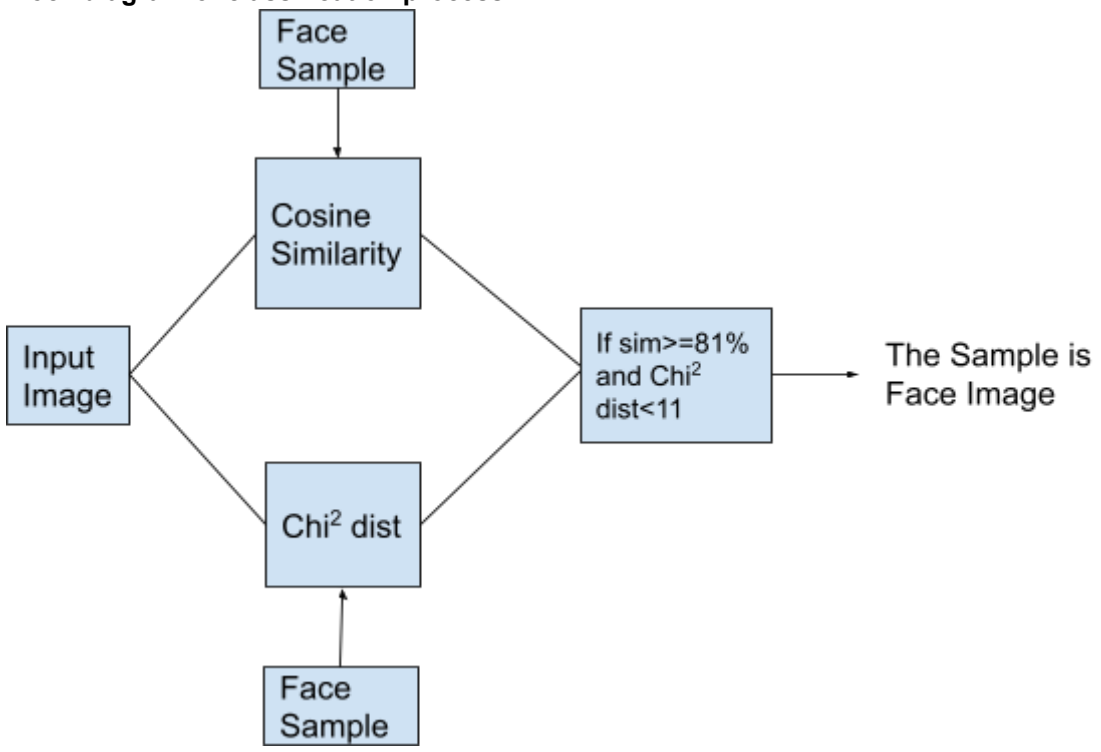
For LBP Analysis on the dataset we will convert it to grayscale and then we have to find the optimum number of non-overlapping windows in which we can divide the images into, for this I calculate the factors of the Image shape.

We split the images into the above mentioned non overlapping windows, after that we need to perform LBP operator which can be considered to be going over the image windows in 3*3 kernel. Hence, the 3*3 windows for LBP are constructed, and their binary values are calculated.

In LBP for converting the values to decimal we have to go in the 3*3 resultant matrix in a clockwise or anticlockwise manner.
We then plot the histogram of LBP values over the non-overlapping windows.

Question 4(b)

Block diagram of classification process



Your comments:

b) We have to create a global descriptor which tells us about the texture of the entire image. There are multiple approaches for this, one of which I used involved concatinating the histograms(values) of all windows using np.concatenate and then plotting the histogram of the concatenated array.
This is our global descriptor for texture analysis.
Alternate approaches, One of the approach can be considering the whole image as a window and computing the LBP of the entire image, this will be our global descriptor.

Second approach for this could be creating an empty array and appending the LBP values of all windows into the array and then plotting the histogram of that array.

For classification process, I used cosine similarity and chi^2 distance between two histograms(Global Descriptors) and if both the resultant values are meeting a certain threshold, I can safely classify them belonging to a certain class.
Cosine similarity gives us the measure of similarity between two histograms and the chi^2 distance gives us the dissimilarity between two histograms.
For my classifier, I was able to classify the images to their respective classes.
The classifier failed to classify only in one or two cases, which can be attributed to the other factors affecting the texture of an image.

Question 4(c)

Your comments:

c) When I decrease the window size the histogram becomes more descriptive and has more values in it and the classifier works, as expected for the given task Face v/s Non-Face.

Question 4(d)

d) After Increasing the window size to 64 from 32, the histograms became more general and less descriptive and hence the classifier I used because the two histograms became really similar. I took the cosine similarity and the chi2 for a face and non face image and the measure of their dissimilarity was really low.

Question 4(e)

e) Dynamic textures are sequences of images in a video that exhibit some properties in time. They are statistically similar and temporally stationary. If LBP operator can be made reliable using Uniform-LBP can be improved. Here all non uniform patterns are grouped into a single bin and hence the reliability is partially solved. Since, Dynamic textures recognition mainly relies on spatial information, we can use LBP for the Dynamic Texture recognition.

Object Segmentation and Counting
Question 5(a)

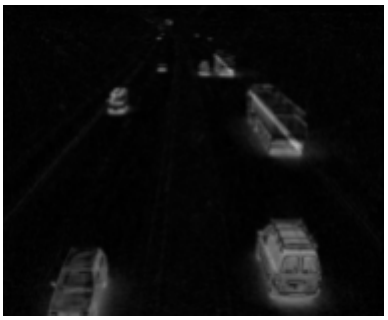
Original frames:

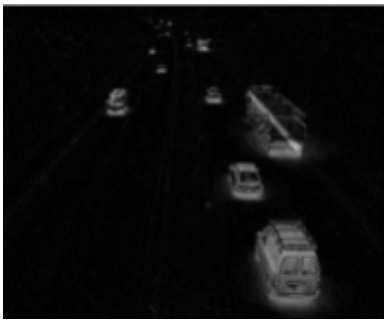
Frame 0

Frame 50

Frame 120

Frame differencing:





Threshold results:

Question 5(b)

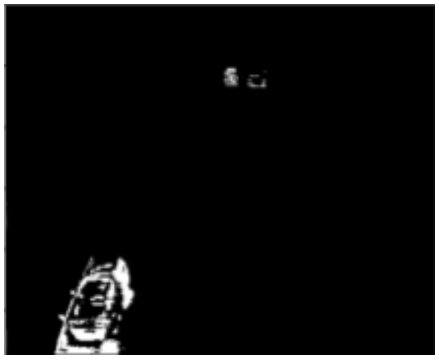
Original frame:

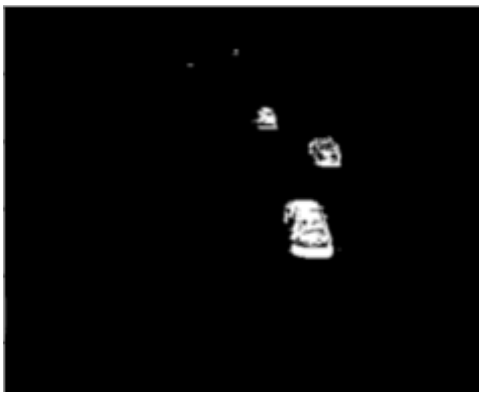
Frame 50

Frame 120

Frame differencing:

Threshold results:





Your comments for 5a,5b:

Understanding:

The fifth question of the task we have to perform object counting. There are multiple ways in which we can calcuate the difference between the frames, for eg. we can use absolute difference between two frames or use euclidian distance between the two frames. The difference between two frames gives us an idea of motion in the frames. Hence we can use the frame differencing approach to identify moving objects. We can then give the different objects a binary value and turn them into a blob and count the number of objects in each frame.

Analysis:

I chose to perform this task in the grayscale because when working on BGR images i was getting a lot of artifacts in my difference between frames which challenged me further when counting the number of objects.
After converting all frames in the video into grayscale, I calculated absolute difference between the frames.

For the first part a) I chose the first frame of the sequence a my reference image and performed frame differncing over the entirety of frames. After which I set a threshold on the frame differences to convert them into grayscale.
Using this I could figure out a vague boundary about the moving objects of all frames with respect to the first frame.

b) Now instead of fixing the reference frame we will change the reference frame for every other frame by taking their absolute difference between the previous frame and the current frame.
This gives us the measure of change between each of the frames. Using the thresholding we get a better tracking of objects which are moving in every frame.
In contrast, The first approach gave us all moving objects between the first frame and the current frame.

Question 5(c)

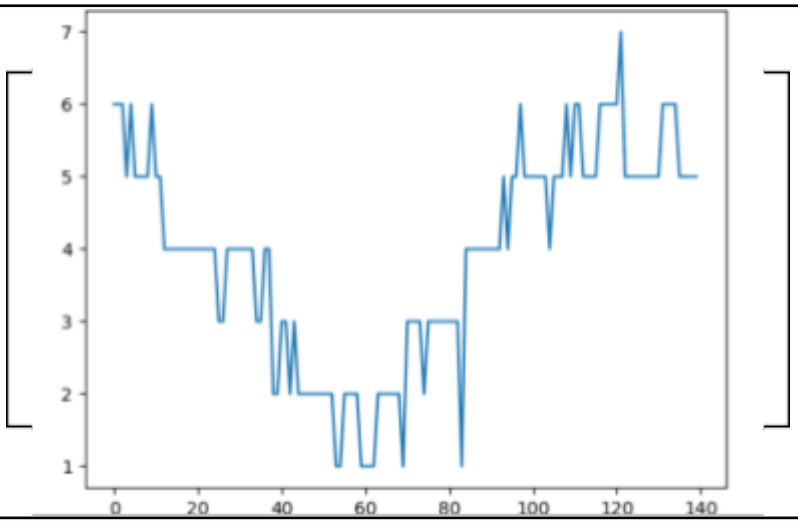
Your comments:

c) In this part we had to generate a reference frame/background frame from the given frames.

For background estimation I used median to calculate median of each pixel in a range of random frames (For.eg 50 frames) using this I successfully generated a background frame for our case.

This approach would fail in giving a clean background if there were alot of cars or heavy traffic in the frames.

Question 5(d)



Your comments:

d) For this part we have to count the number of moving objects in the video, This can be done by using the frame differences between all frames and the reference frame we generated in part c).
After calculating the frame difference between the frames and converting them into a binary image we have to fill all the holes in the masks of the objects.
For this I used k-nearest neighbours to fill in the holes with 1 for a n*n window if their sum is approximately equal the the n^2.
Then I smoothened the values I got from this knn using the max pixel value in the given window and making the entire video take the value 1.

This resulted in different blobs in the image, now all we have to do is count the different objects in each frame to get an approximate number of moving objects in the entire video.