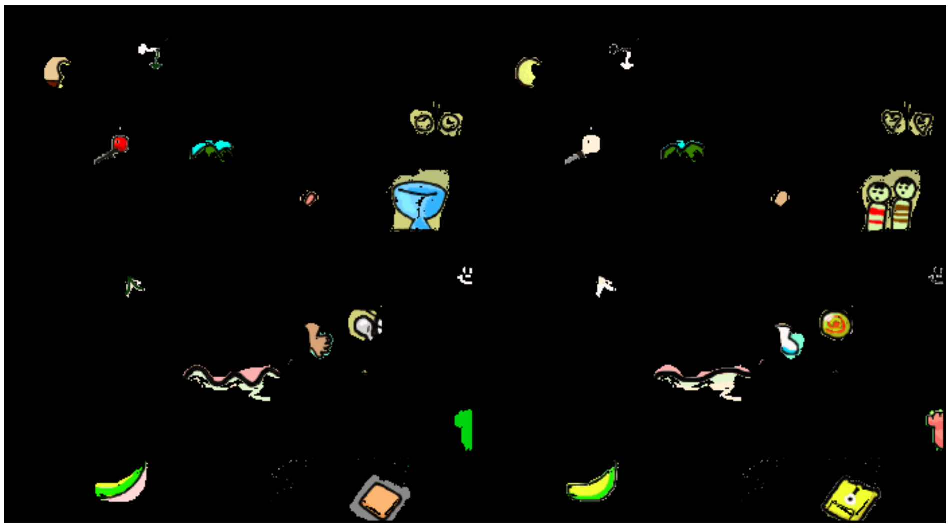
CV Assignment-1

By:- Kwanit Gupta (B19EE046)

Q.1 Given a stitched image containing two very similar scenes, find out the differences. (a) Submit your implementation. (b) Write down your algorithm in brief. (c) Show the image where differences are suitably marked. (d) Write down scenarios when your implementation may not work.

Solution:-

- Input the image, using OpenCV's imread command.
- Sliced the Numpy array and went with 2 approaches (one with an absolute difference and the other being a ratio one).
- The ratio approach didn't perform well, since the pixel variations and the segmentation output came near 1.0. And even with the thresholding and tolerance value, the suitable masks weren't prepared very well.
- So, I scrapped off and went with a difference approach. Noted down the positions of the changed pixels (applying the condition of having non-zero pixel change) and made a segmentation mask. Then utilized that mask to uncover the different places of change and visualized them accordingly.



- This set of approaches won't be robust to sensitive color palettes (i.e slight change in the consecutive pixels), hypersensitive to pixel orientations (even the slight change of row/column affects the outcome masks). Also, illumination and the corresponding noises/attenuations will be a bigger issue in the working of this.

Q.2 Given an image of the map of India, find out the pixel distance between the two states. [Hint: use off-the-shelf OCR] (a) Submit your implementation. (b) Write down the limitations of your approach.

Solution:-

- Input the image, using OpenCV's imread command.

- I went for 2 different approaches initially (pytesseract's functionality and easy-ocr). But pytesseract performed poorly in the whole image (even after fluctuating the confidence threshold, the resulting bounding boxes didn't improve).

- Better results started to come after incorporating the sharpening kernel (made in the codebase) for pytesseract and easy-ocr. Even cutting down the region and applying the pipeline was an idea (but it won't have been a generalized one).

- Also, suitable state prompts were given, so that mapping would be done and Euclidean distance metric to the bounding box's centroid would be applied and printed accordingly.

Q.3 Given an image of a circle, find out the area and perimeter in the pixel unit. Submit your implementation such that it takes the image file as an argument and prints the area and perimeter in new lines.

Solution:-

- Input the image, using OpenCV's imread command.

- Went for 2 approaches, in the following manner:-
    - From scratch (only accounting with respect to the image given), for-loops were incorporated so that black pixels would be identified and accounted for the pixel-wise measured perimeter. Afterward, with the mathematical formulation of the radius and area, I made the values as integers to approximate them closer to that of the number of pixels.
    - Using Hough Circle Transform, I obtained the center coordinates (x,y) and radius (r). Then I calculated the perimeter and area using the conventional mathematical formulation and integer approximation.

- With the 1st approach, the values fluctuated heavily with the incorporation of data preprocessing or noise. On the other hand, those fluctuations lowered with the usage of Hough Circle Transformation.

- Made the pipeline according to the requirements of the question (input and output).

Q.4 Given an image of a clock find out the angle between the hour and minute hands. (a) Submit your implementation. (b) Write down your approach to finding out the angle. (c) Write down the limitations of your approach.

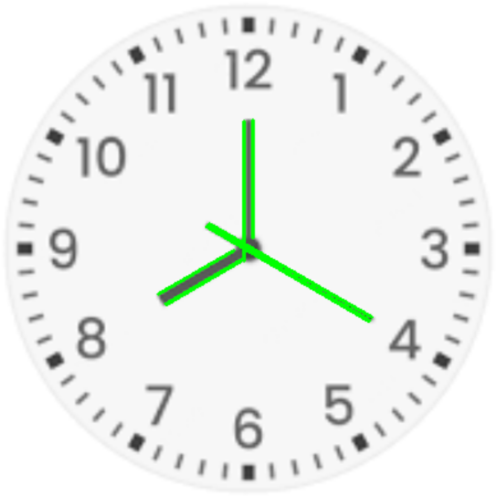
Solution:-

- Input the image, using OpenCV's imread command.

- Made gradients-based histogram (via Sobel Filter and Cartesian-to-Polar API) to understand the variations in the pixels and maybe figure out if some maximizing slopes can help to figure out the angle of minute hand (since it's the longest and most of the pixels point towards that direction).

- As studied in class, I applied Hough Line Transformation to figure out the extent of lines. With/without data preprocessing, the intermediate detections turned out to be different and the hyperparameter tuning was very sensitive since in some of the trials the detections weren't happening.

- Before that, I applied Canny Edge Detection (to formulate contours at a finer level). While receiving the lines via coordinates, my approach is to find the line with the highest magnitude alongside figuring out its index of it to identify the slope list's element. Then the 2nd highest's magnitude element will point out to the hour hand alongside its slope. So, with the help of both slopes, it will become easy to point out the angle between those hands.



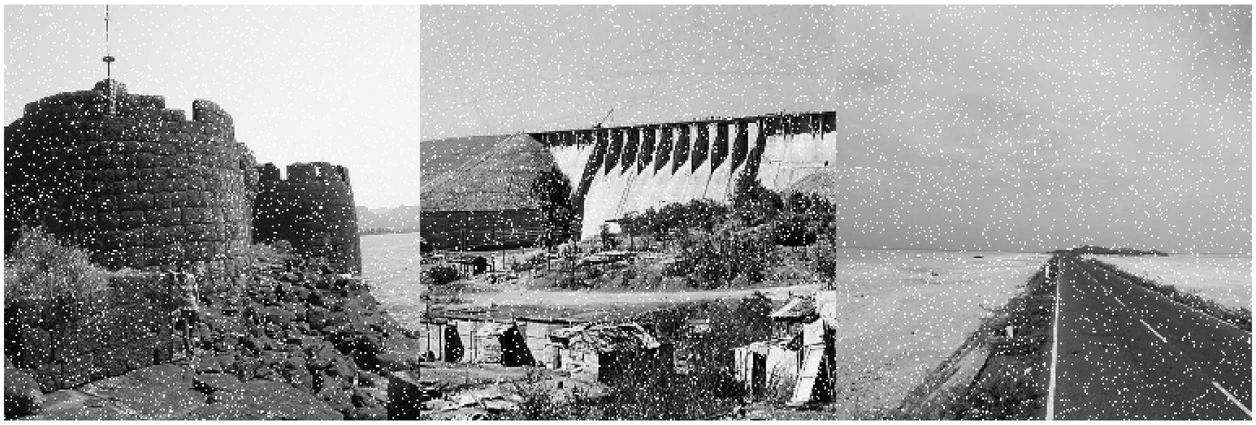
- Talking about the limitations, it is hyperparameter sensitive, data-preprocessing dependent (blurring/sharpening highly affects the outcomes and histograms), and hypersensitive towards pixel-wise distribution (results varied with variations in color/grayscaled images).

Q.5 Choose three images of a world landmark from the Google Landmark dataset (Link: https://storage.googleapis.com/gld-v2/web/index.htmlleapis.com/). The name of your chosen landmark should begin with the first letter of your first name. For example: If your name is Adhrit, you could choose Amarnath. (a0) Resize all images to 256 × 256. Convert it to gray. (a) Show the average of all three images. (c) Subtract Image 1 with Image 2. (d) Add salt noise with 5% probability in one of the images. (e) Remove the noise. (f) Use the following 3×3 kernel: {−1, −1, −1; 0, 0, 0; 1, 1, 1} for performing convolution in one of the images and show the output.

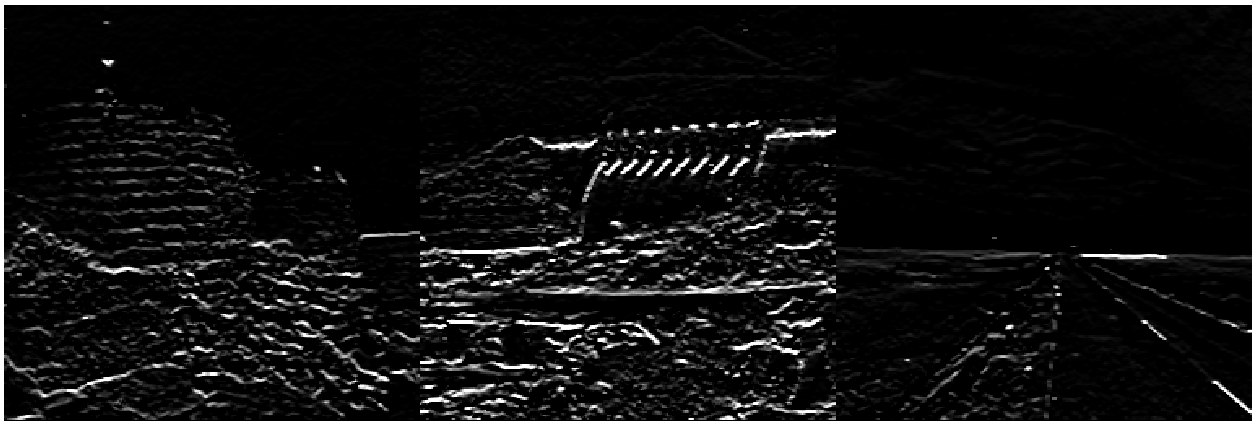
Solution:-

As my name is “Kwanit”, and my first letter is K, so I search the given website and did the following:-

- Fetch the images, using wget API.
- Input those, using OpenCV's imread command.
- Established comparative changes and studied while resizing, and grayscaling so that the confirmation of information loss can be given.
- Further operations are also done via the functioning paradigm (creating necessary DEFs), even for salt noise too with the flexibility of maneuvering percentage.



- Removed the noise using Median Filter and performed the convolution kernel on the original as well as the denoised image (turned out to be an edge detection filter).



Q.6 You will be given 100 handwritten images of 0 and 1. You have to compute horizontal projection profile features and use Nearest Neighbour and SVM classifiers to recognize the digits. Report accuracy and show some visual examples. Dataset (choose only 0 and 1): https: //github.com/myleott/mnist_png.git

Solution:-

- Cloned the repository into notebook/system using Github's API.

- Unzipped the tar file into the folder and then Input the image, using OpenCV's imread command for visualization.

- Made the matrix corresponding to the following feature variants:-
    - Flattening the whole image and inserting each sample as a row vector.
    - Calculating the Horizontal Projection Feature altogether and constructing a concise feature space (28 times lesser).

- Implemented KNN and SVM using sci-kit-learn as a primary framework and observed that the first approach resulted in 100% accuracy and others proved to provide 96-98% accuracy.

Q.7 Given a word image find out if the word is bright text on a dark background or dark text on bright background.

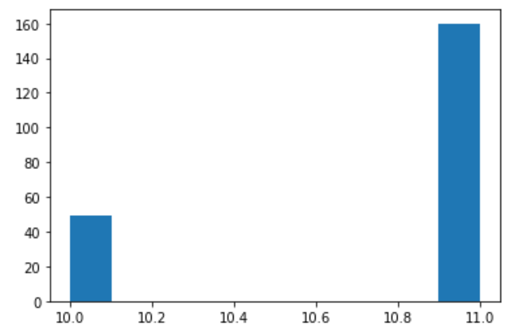
Solution:-

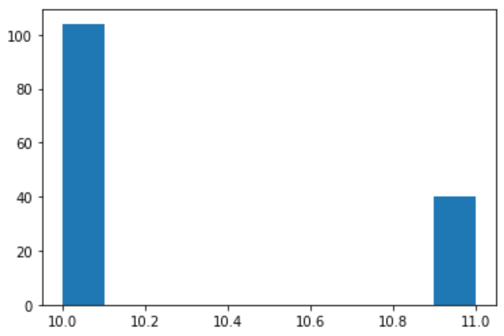
- Input the image, using OpenCV's imread command.

- Inspired by the idea of “Horizontal Projection Features”, I built a sliding window-based detection function, where column-wise features would be accumulated.

- After receiving the row vector, it was log-transformed since the values were way higher, and for simpler bifurcation, approximations were made to convert them to integer values.

- With the assistance of the background’s projection feature value (assuming that the background starts with the corner columns and rows), the verdict is made (if the frequency of fewer column values is more, then dark text on bright, otherwise vice versa)





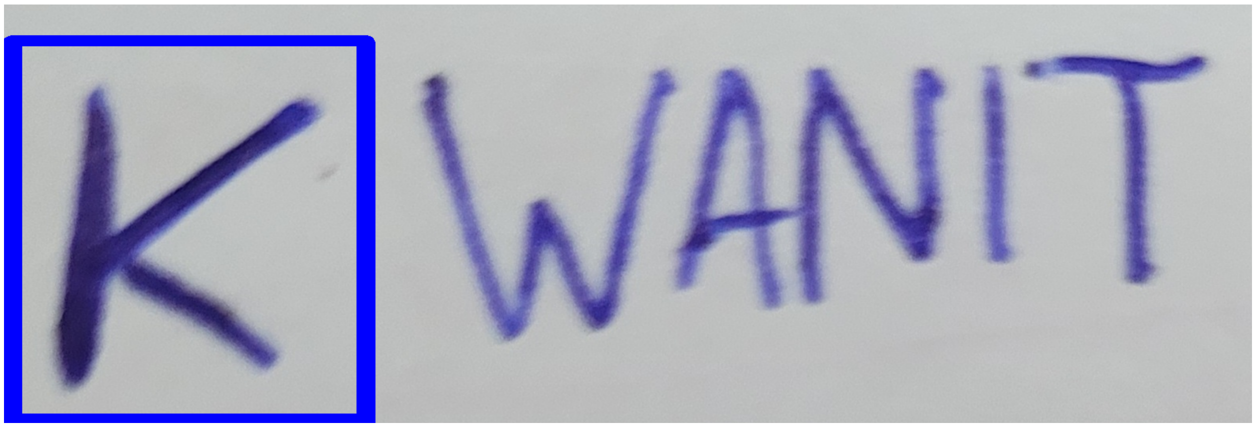
Q.8 Write your name in capital letters on a piece of white paper and a random letter from your name. Click photographs of these. Implement the Template Matching algorithm and discuss your observation.

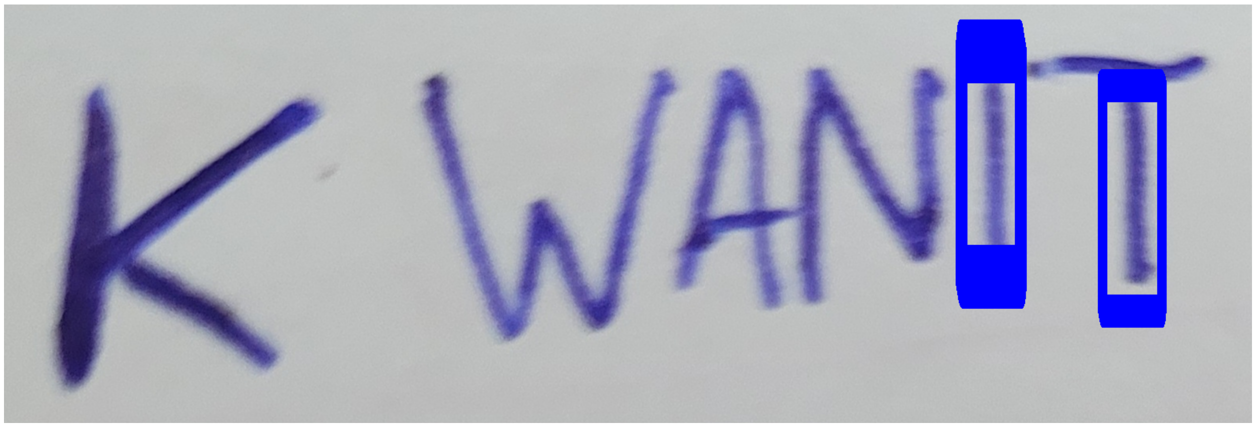
Solution:-

- Input the image and template, using OpenCV's imread command.

- As studied in class, 4 variations of the Template Matching algorithm were covered in the codebase (['cv2.TM_CCOEFF', 'cv2.TM_CCOEFF_NORMED', 'cv2.TM_CCORR', 'cv2.TM_CCORR_NORMED']).

- Since each variant differed with the working and its mathematical intuition, their behavior towards the template matching also differed with different confidence thresholds (The normed one performed way better than their originals).





- For normed ones, the thresholds were kept low since they provided good results. On the other hand, the original ones performed poorly even reaching the threshold of 0.999. These can be seen in the codebase and notebook too.

Q.9 Choose one image from Problem 5. Show histogram of pixel values with bin size 10. Perform histogram equalization and show the output image.

Solution:-

- Input the image, using OpenCV's imread command.

- Channel-wise histogram evaluation was done since the arrangement of pixel intensities can be simultaneously studied.

- Apart from that, the Gray-scaled image was also studied with a suitable histogram representation, since it didn’t overlap or become similar to those of channel histograms.

- To understand more about histogram equalization, 2 variations were covered in the code base (naming CLAHE and Global Histogram Equalization), alongside a suitable comparative plot.

Q.10 You will be given an image of a mobile number. Use off-the-shelf OCR and find out the last three digits of the mobile number.

Solution:-

- Input the image, using OpenCV's imread command.
- Utilize the pytesseract's Image_to_string API to fetch the text from the Numbered Image.
- While doing that, some empty string lines were accounted for. So, to deal with that, indexing was done for the last 3 elements of the detected numbers.
- It somehow made me understand the limitations and importance, created by the arrangement of text into the picture.

Link for the Notebook (Constitute all the Questions together, into a single Notebook, instead of individual notebooks)