CV Assignment-1 Report

Note: All the links for the code and data are provided at the end of the report.

Task: To give solutions and approaches to the given problems.

Objectives:

Question 1.

- (Spot the diff)
- 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.

Procedure:

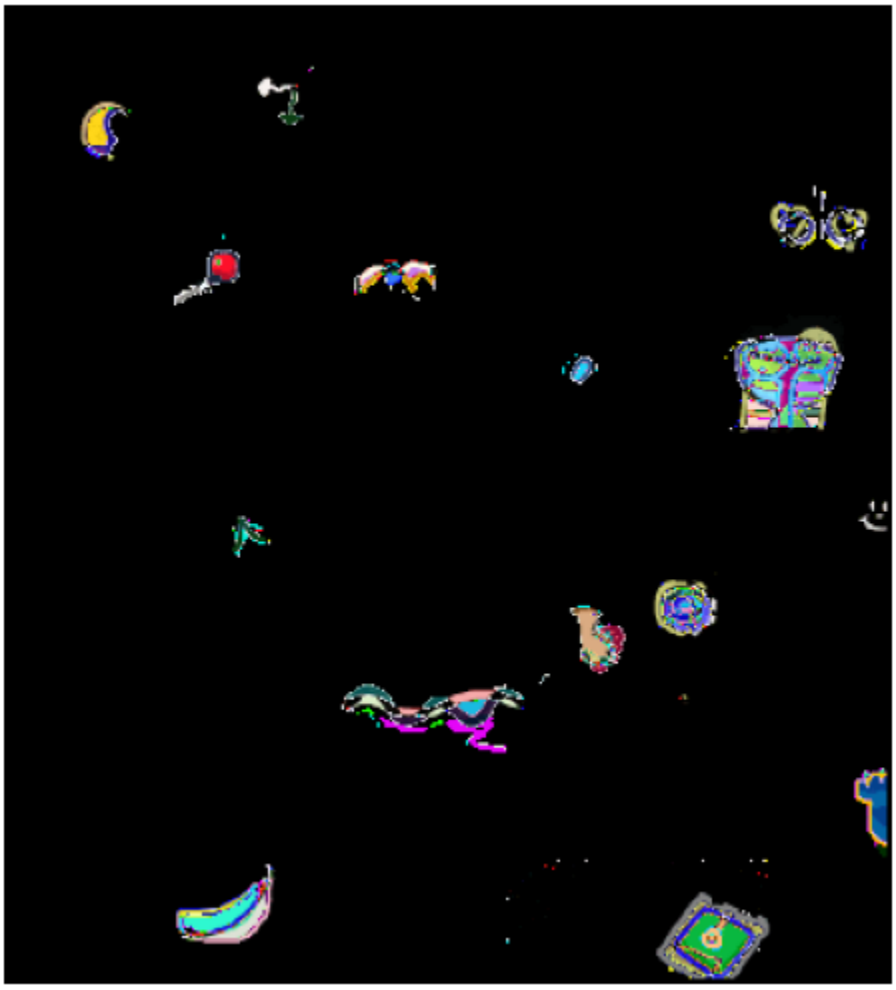
- Import required packages. (OpenCV, NumPy, Matplotlib, …etc.)
- Upload the data, The stitched image.
- Load the image with open-cv.
- Splitting the image into two similar photos.
- Plotting original images and split images.

Image 1

Image 2

- Taking differences in the images will show the difference b/w the photos.

Differences



- Adding an image with a difference from the above step (image).

image 1

image 2

marked Images

- Scenarios when your implementation may not work
    - If the stitching on the images was not done in 2 halves.
    - If the scale of the image were different.
    - If any transformation is applied to the image.

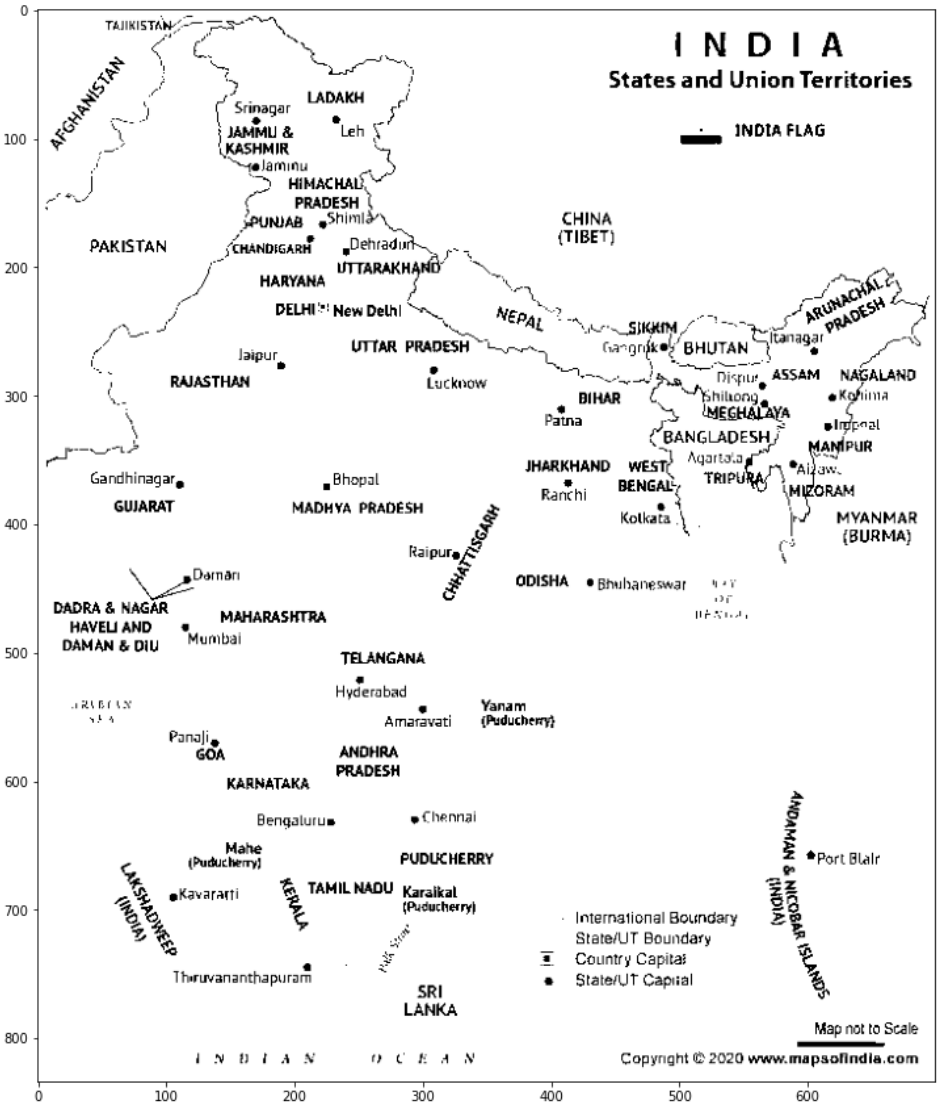
Objectives:

Question 2.

- (Distance in images-1)
- 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.

Procedure:

- Import required packages. (OpenCV, NumPy, Matplotlib, …etc.)
- Upload the data, The map of India.
- Installing system dependencies by apt install tesseract-ocr -y;
- Installing pytesseract with pip.
- Restart the runtime because the new dependencies are installed.
- Load the image with open-cv.
- Convert to grayscale and apply thresholding for ocr.



- Apply OCR on the image, and find the coordinates of the texts (state names).
- Apply norm / euclidean distance between coordinates and get the distance in units.
- `the distance between Gandhi nagat to Hyderabad is: 200.0 units`
- limitations of the approach
    - Dependents on the OCR tool.
    - Finds the distance between image pixels
    - It needs more processing to get all the entries from the map.

Objectives:

Question 3.

- (Distance in images-2)
- 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.

Procedure:

- Import required packages. (OpenCV, NumPy, Matplotlib, …etc.)
- Upload the data, The map of India.
- Load the image with open-cv.
- Note:- the image was in PNG format with alpha channel.
- Converting image to grayscale.

Image in png

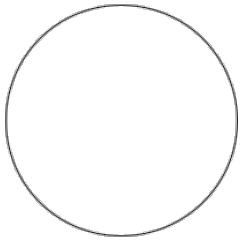
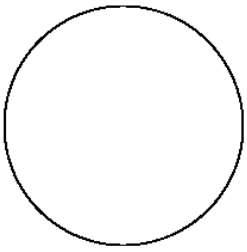


Image in gray



- Take max chords || to the x and y axis, respectively, and find the circle's radius.
- Results:-
    - The area of the circle is: 31259.04 $units^2$
    - The circle's perimeter is: 626.75 $units$

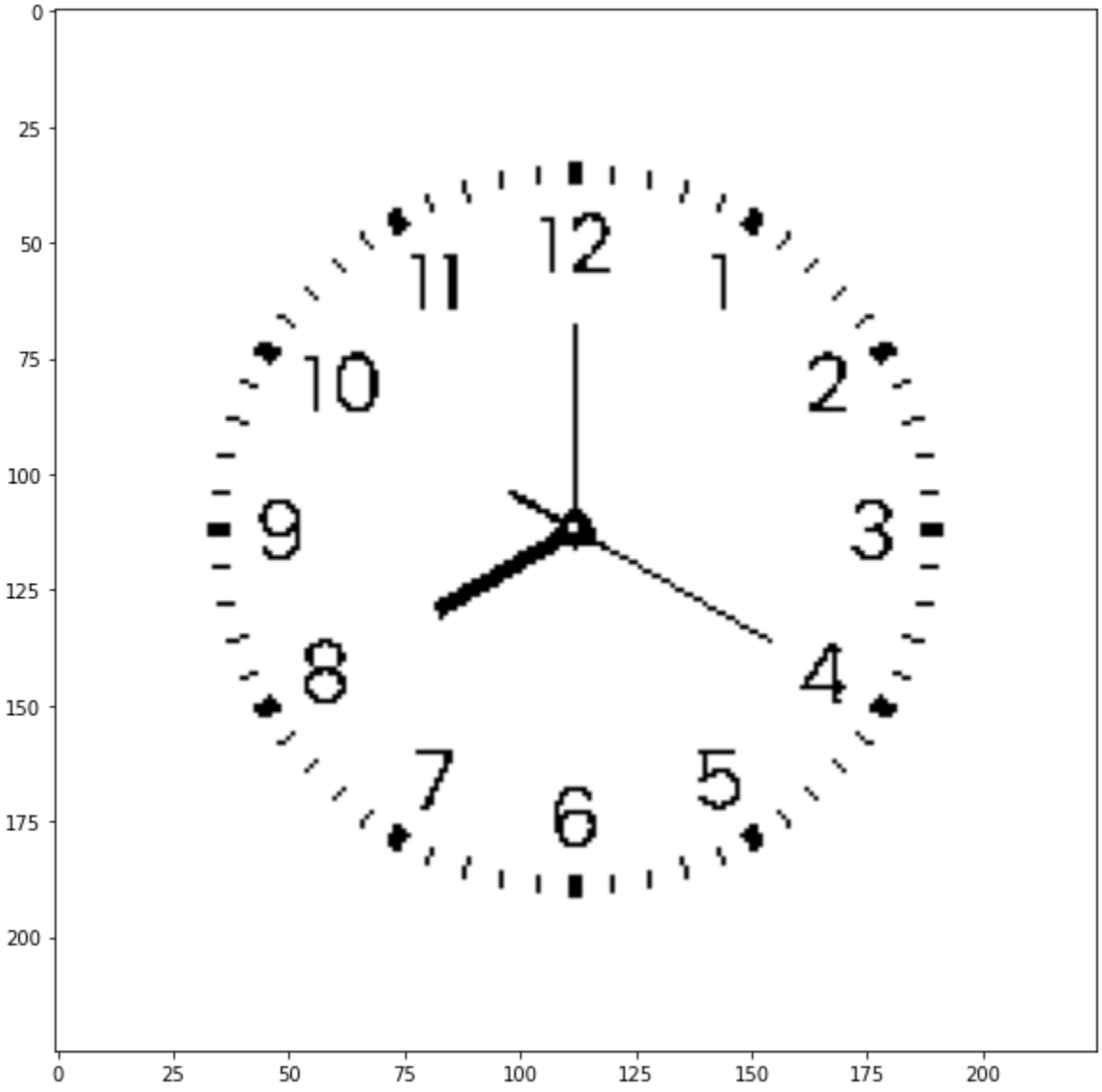
Objectives:

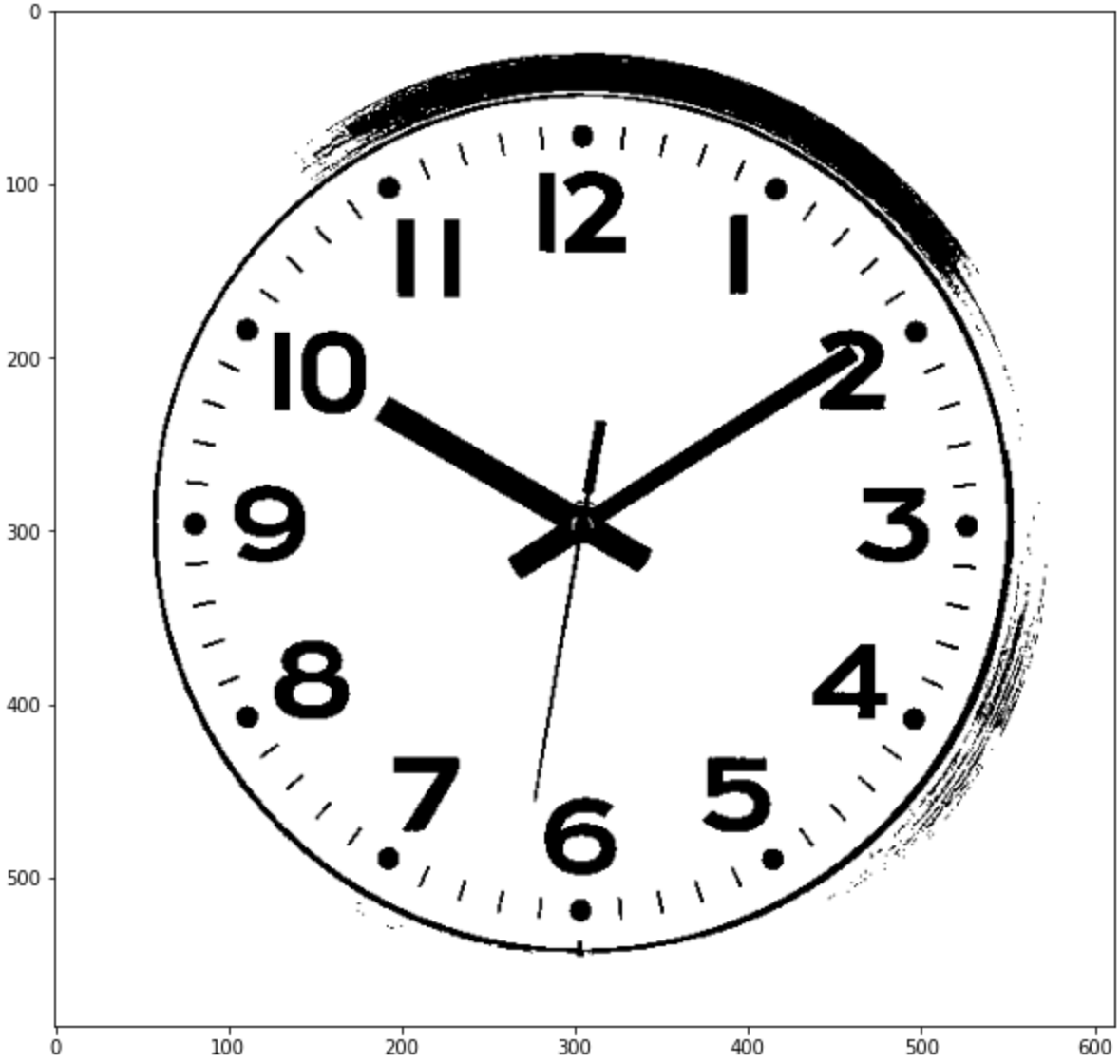
Question 4.

- (Towards reading time)
- 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 for finding out the angle.
- (c) Write down the limitations of your approach.

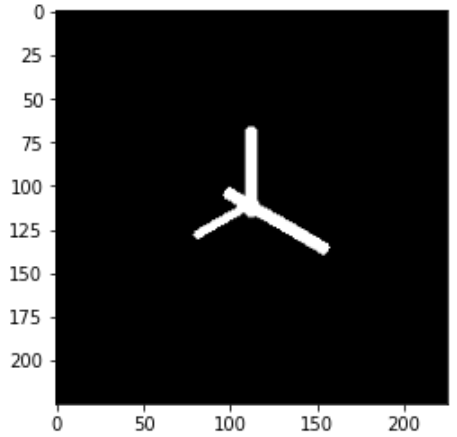
Procedure:

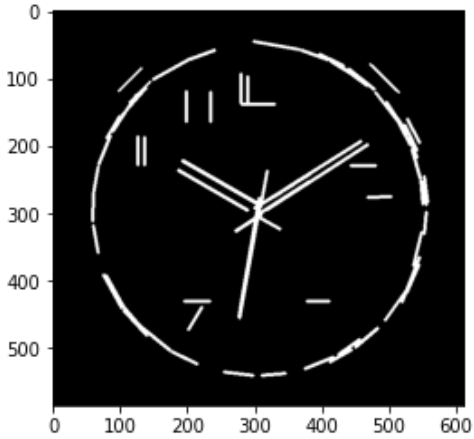
- Import required packages. (OpenCV, NumPy, Matplotlib, …etc.)
- Upload the data, The clock images.
- Load the image with open-cv.
- Convert the image to grayscale images.
- Apply thresholding to get better edge detection.





- Apply Canny Edge detection, followed by HoughLines Probabilistic line detection.
- Ploting the image after detection.





- Finding the coordinates of the minute and hour lines from the above images.
- Calculate the Slop of lines (m1,m2)
- Find the angle with the help of the formula: $tan^{-1}|(m2 - m1)/1 + m1m2|$
- Results:-
    - the angle is: 150.06
    - the angle is: 117.85
- limitations of the approach:-
    - This approach will work for almost all the clocks but need a little manual intervention of identifying the minute and hours hand lines.
    - Results are not very accurate for clock image one, but pretty close for clock two.
    - Requestion pre-processing of the images in order to give correct results.

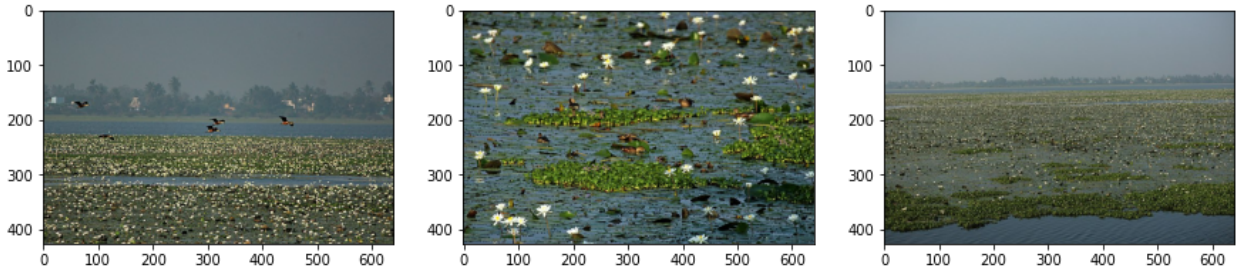
Objectives:

Question 5.

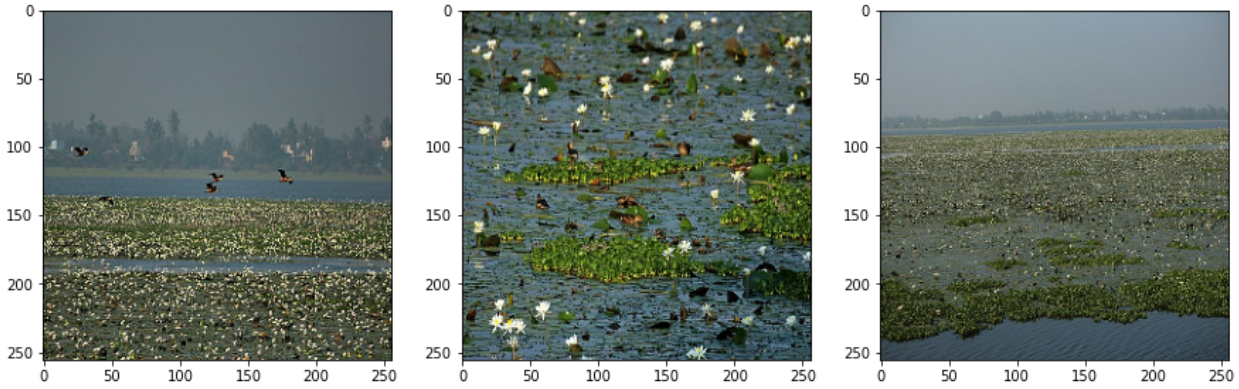
- (Fun with Landmarks)
- Choose three images of a world landmark from the Google Landmark dataset (Link: https://storage.googleapis.com/ gld-v2/web/index.html).
- The name of your chosen landmark should begin with the first letter of your first name.

Procedure:

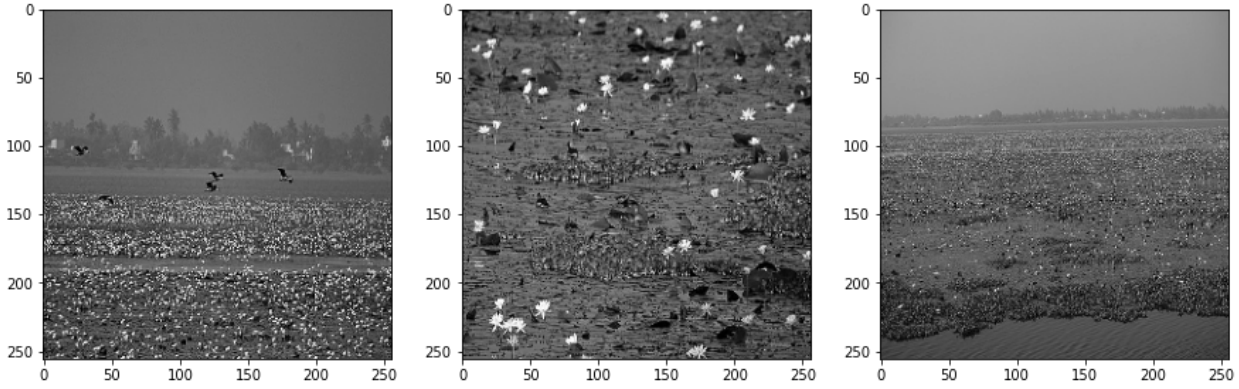
- Import required packages. (OpenCV, NumPy, Matplotlib, …etc.)
- Downloading the data, The images of landmarks with the help of a small script.
- Load the image with open-cv.
- Plotting the images.



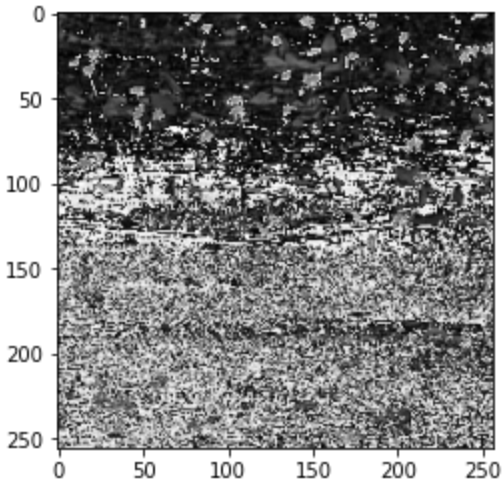
- Applying Resize operations and plotting the results.



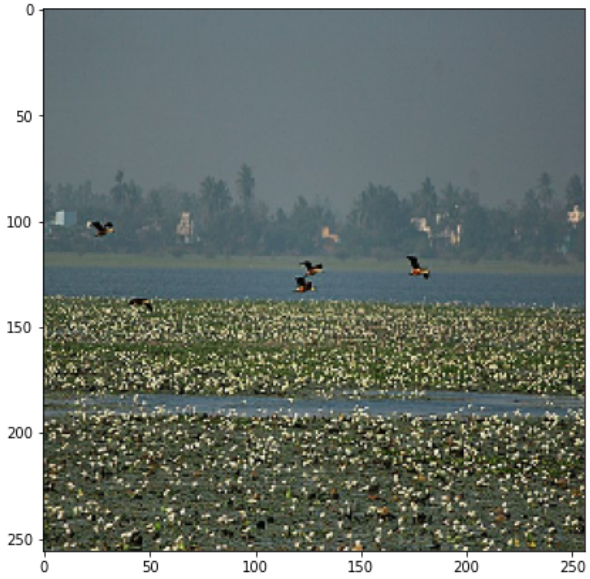
- Note the scale for the images in the above plots has been changed.
- Converting the images to grayscale images and plotting the results.

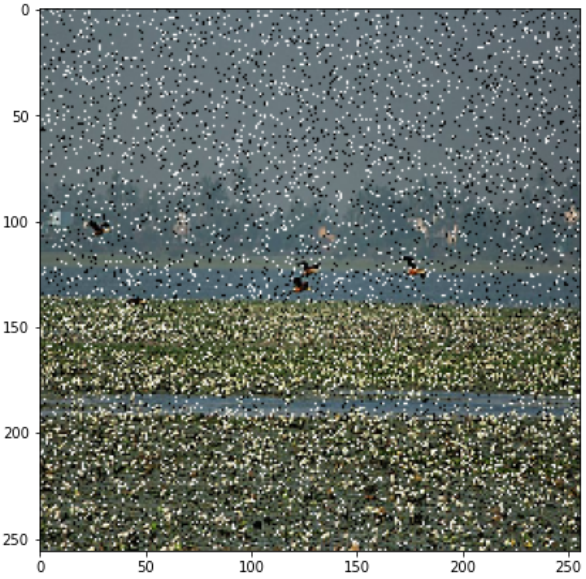


- Subtracting image two from image one.

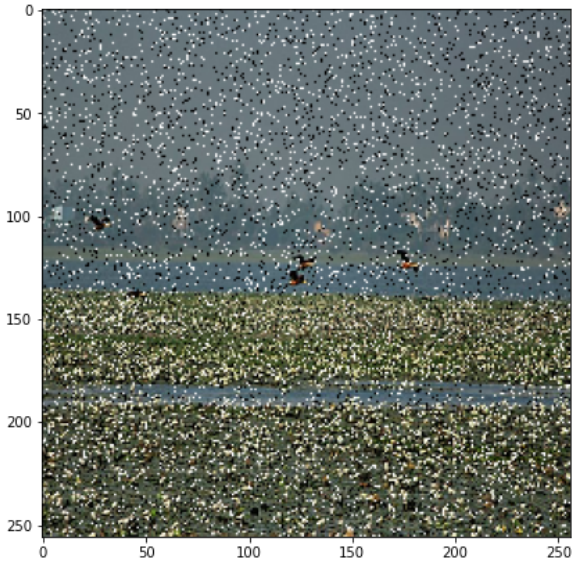


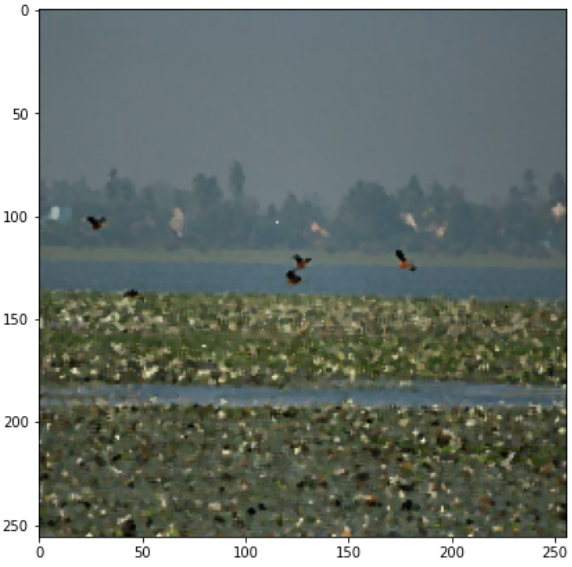
- Adding Noise (salt and pepper) to image one.



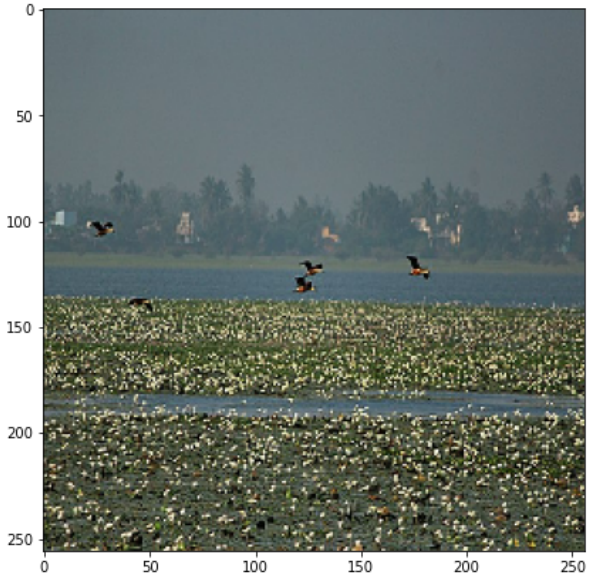


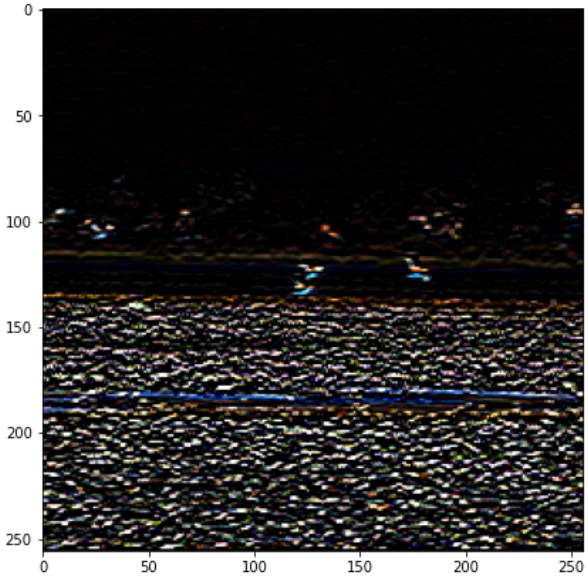
- Removing the Noise from the above image with the help of median Blur.





- Apply custom kernel = $([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])$ to the image.





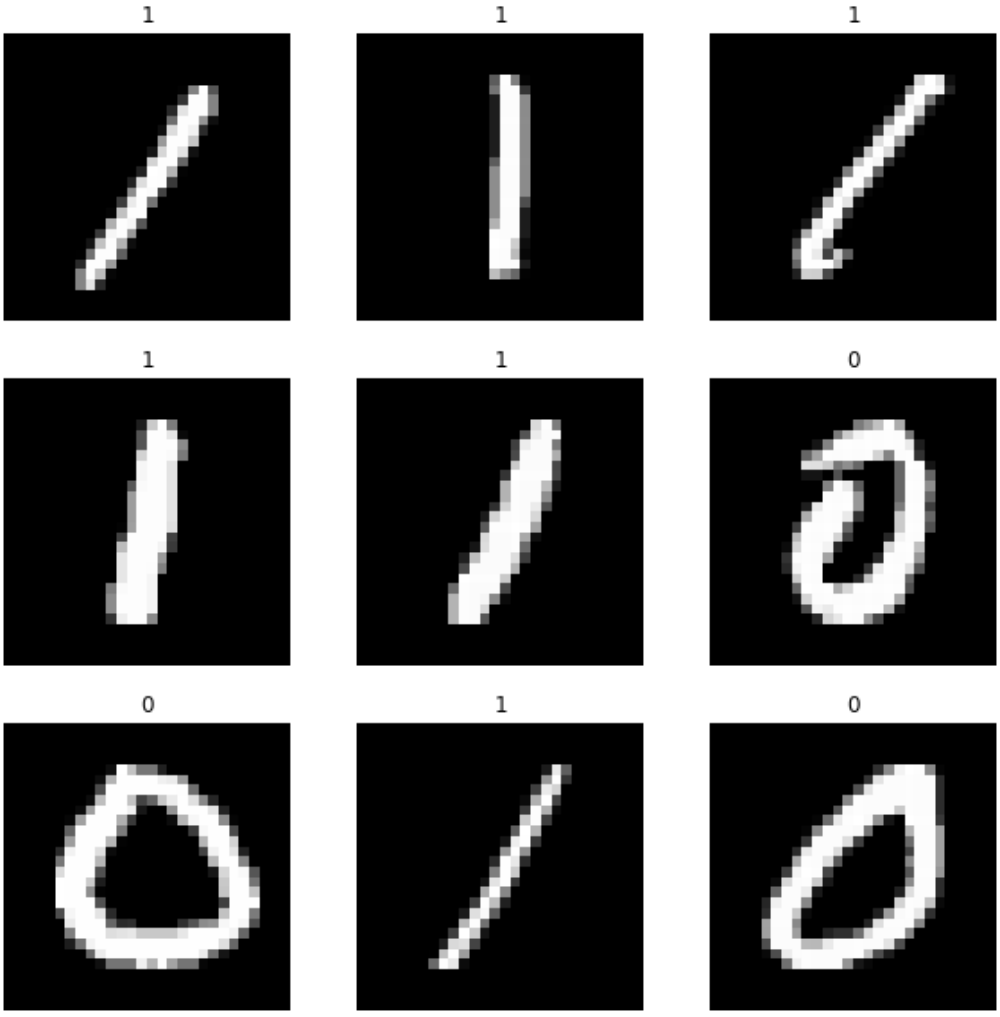
Objectives:

Question 6.

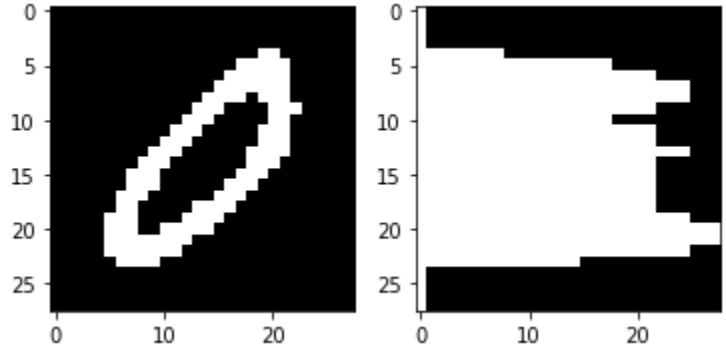
- (Digit Recognition)
- You will be given 100 handwritten images of 0 and 1.
- You have to compute horizontal projection profile features and use Nearest Neighbour and SVM classifiers to recognize the digits.
- Report accuracy and show some visual examples. Dataset (choose only 0 and 1): https: //github.com/myleott/mnist_png.git

Procedure:

- Import required packages. (OpenCV, NumPy, Matplotlib, sk-learn, …etc.)
- Downloading the data, The images of landmarks with the help of a small script in tar.gz format.
- Extracting the zip file downloaded from the above step.
- Removing extra data, like images from 2 to 9, with the help of regular expression.
- Keeping only 100 samples for zeros and one.
- Making dataset with the help of pathlib module of python.
- Plotting random samples from the dataset.



- Computing Horizontal projection of one random sample from the dataset.
- Plotting the projection.



- Making Datasets of the projection images of size (28 x 28)
- Plotting random samples from the projection dataset.

- Making the KNN classification model with n_neighbors=3.
- Training the model with horizontal projection dataset.
- Testing the model with horizontal projection dataset.
- Result:-
    - Acc. with knn n_neighbors=3, is: 89.57%

- Making the SVM classification model with gamma=0.01.
- Training the model with horizontal projection dataset.
- Testing the model with horizontal projection dataset.
- Result:-
    - Acc. with svm gamma=0.01, is: 95.26%

Objectives:

Question 7.

- (White on Black or Black on White)
- Given a word image, find out if the word is bright text on a dark background or dark text on bright background.

Procedure:

- Import required packages. (OpenCV, NumPy, Matplotlib,, …etc.)
- Upload the data, The Texts images.
- Load the image with open-cv.
- Convert the image to grayscale images.
- Apply thresholding to remove noise from the images.
- Applying Gradient on the images

- Counting the number of -ve and +ve gradients for respective images.
- If -ve gradients in an image are greater than +ve gradients, then
    - TEXT IS ON WHITE BACKGROUND"
- If -ve gradients in an image are less than +ve gradients, then
    - TEXT IS ON BLACK BACKGROUND"
- Else
    - Can’t determine

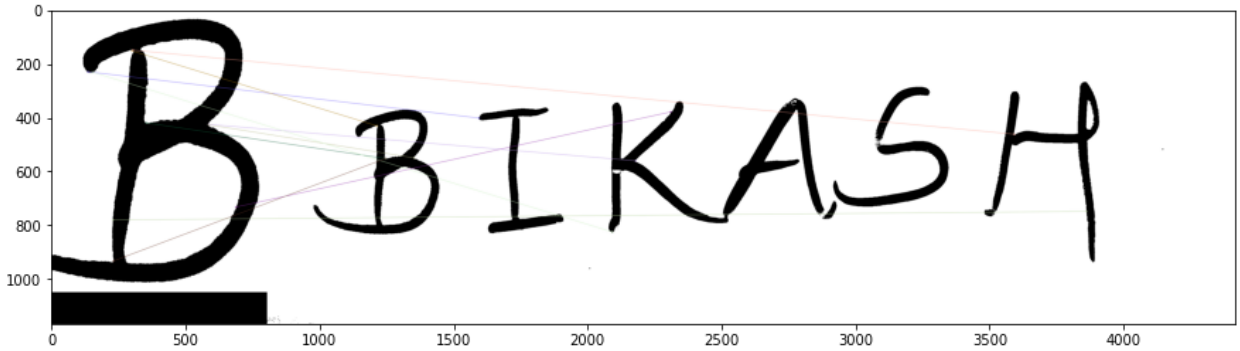
Objectives:

Question 8.

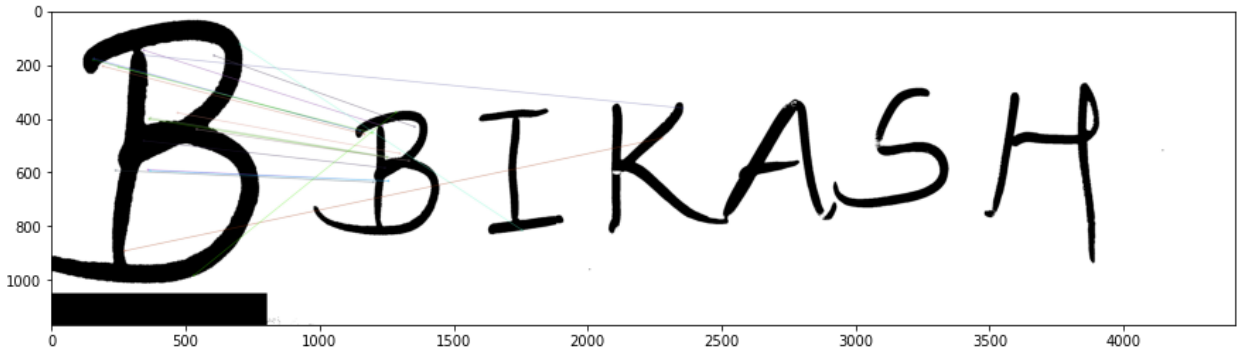
- (Template Matching)
- Write your name in capital letters on a piece of white paper and a random letter from your name.
- Click photographs of these.
- Implement the Template Matching algorithm and discuss your observation.

Procedure:

- Import required packages. (OpenCV, NumPy, Matplotlib,, …etc.)
- Upload the data, The Texts images.
- Load the image with open-cv.
- Apply Binarization on the images.
- Create orb instance.
- Create a BFMatcher instance for brute force matching.
- Get matches sorted by distance.
- Plotting the matches found by the orb detector.



- Creating Swift Detector instance.
- Computing key points with swift object.
- Using brute force matcher with knn matching.
- Applying checks for good matches only.
- Plotting the matches found by the swift detector.



- Observations:
  - It is not perfect.
  - Any transformations leads to ambiguous matches with other alphabets.
  - Many key points are matched unnecessarily.

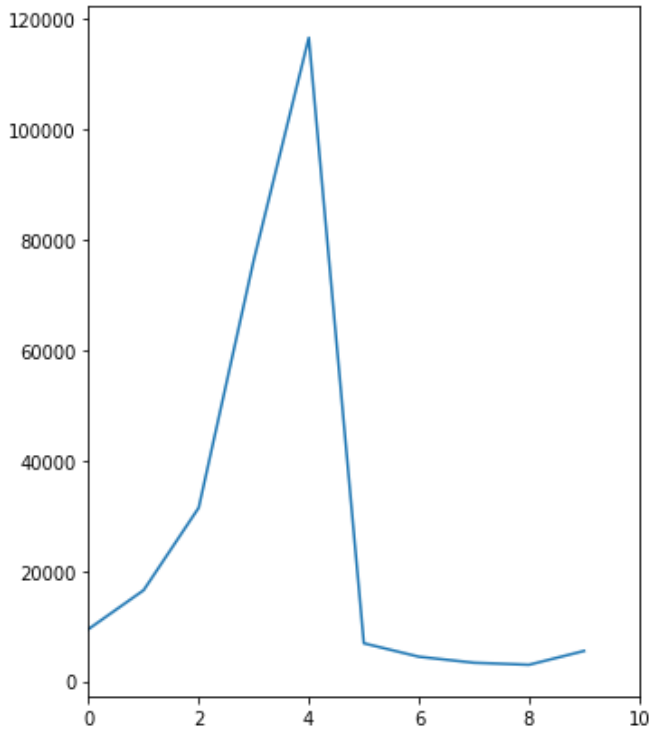
Objectives:

Question 9.

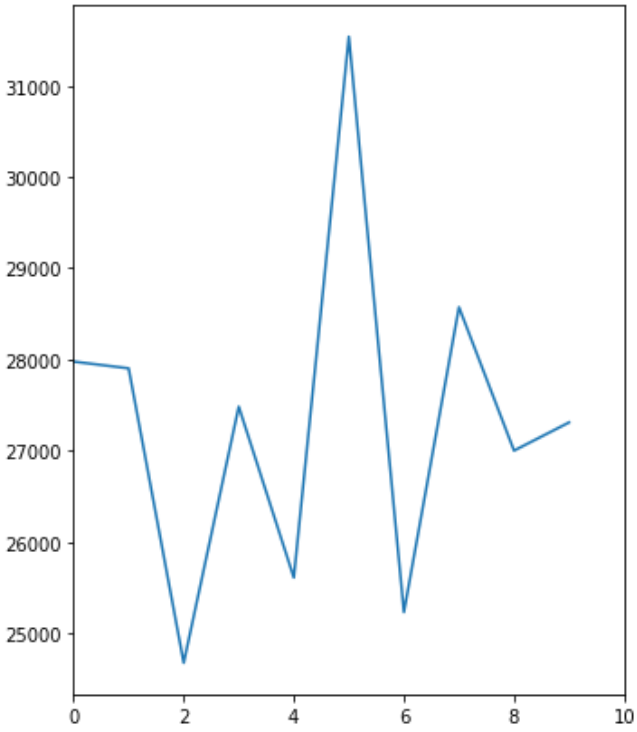
- (Histogram Equalization)
- Choose one image from Problem 5.
- Show histogram of pixel values with bin size 10.
- Perform histogram equalization and show the output image.

Procedure:

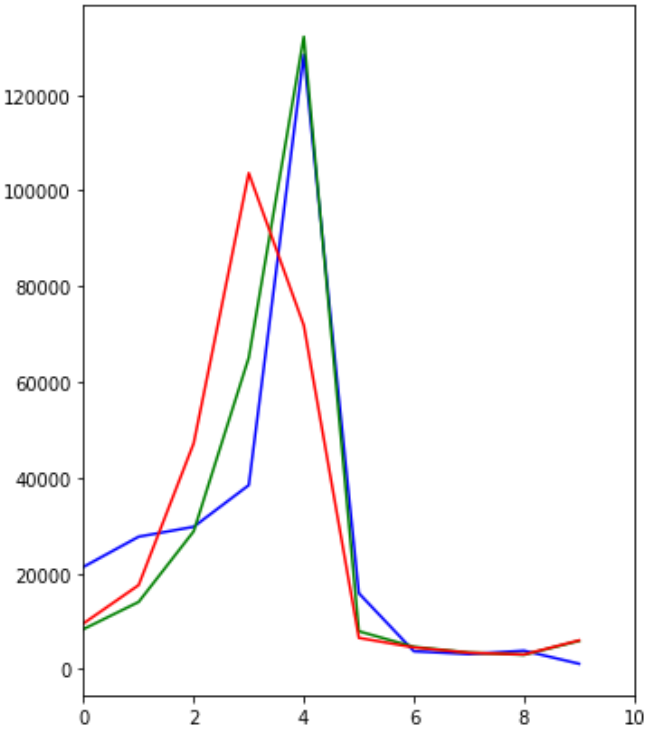
- Import required packages. (OpenCV, NumPy, Matplotlib,, …etc.)
- Downloading the data, The images of landmarks with the help of a small script, same as question 5.
- Converting the image to grayscale.
- Calculate the histogram with bin = 10.
- Plotting the image and histogram.



- Applying Equalization on the image.
- Plotting the Result of the equalization.



- Computing Histogram for Color Image for RGB Channels.



- Note: Equalization can’t be done with all three channels conceding once, it needs to take care of one channel at a time.

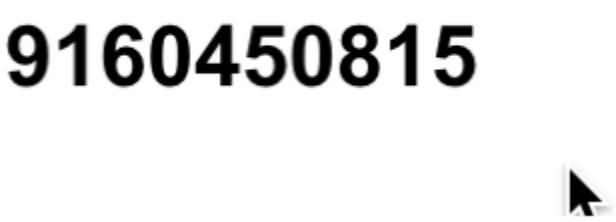
Objectives:

Question 10.

- (Reading Mobile Number)
- You will be given image of a mobile number.
- Use off-the-shelf OCR and find out the last three digits of the mobile number.

Procedure:

- Import required packages. (OpenCV, NumPy, Matplotlib,, …etc.
- Upload the data, The Mobile numbers images.
- Installing system dependencies by apt install tesseract-ocr -y;
- Installing pytesseract with pip.
- Restart the runtime because the new dependencies are installed.
- Load the image with open-cv.
- Convert to grayscale images.
- Applying OCR to the images.
- Printing the last three digits of the phone numbers.



The last 3 digits of the phone number are 815.

The last 3 digits of the phone number are 925.

References:

https://docs.opencv.org/4.x/
https://docs.opencv.org/4.x/d9/df8/tutorial_root.html
https://numpy.org/doc/stable/reference/index.html
https://stackoverflow.com/
https://www.geeksforgeeks.org/opencv-python-tutorial/

Links:

ColabNotebooks + Data:- CV A1

Note:

Use institute id only for accessing the above link.