Optical Character Recognition for Cursive Handwriting

By

Milind Bachani
14bce009

Akash Chaudhary
14bce016

DEPARTMENT OF COMPUTER ENGINEERING
Ahmedabad 382481

Optical Character Recognition

Minor Project - 1

Submitted in fulfillment of the requirements

For the degree of

Bachelor of Technology in Computer Engineering

By

Milind Bachani
14bce009

Akash Chaudhary
14bce016

Guided By
Prof. Anitha Modi
[DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING]

DEPARTMENT OF COMPUTER ENGINEERING
Ahmedabad 382481

CERTIFICATE

This is to certify that the Project entitled "Optical Character Recognition for Cursive Handwriting" submitted by Milind Bachani (14bce009) and Akash Chaudhary (14bce016), towards the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Engineering of Nirma University is the record of work carried out by them under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Prof. Anitha Modi
Assistant Professor
Department of Computer Science & Engg.,
Institute of Technology,
Nirma University,
Ahmedabad

Dr Sanjay Garg
H.O.D. (Dept. of Computer Engineering)
Institute of Technology,
Nirma University,
Ahmedabad

ACKNOWLEDGEMENT

This is to acknowledge that there was involvement of few people whom we would like to give credits and appreciate, one of the main is our own guide Prof Anitha Modi, who not only trusted us but also provided motivation whenever needed, also we got help from our few classmates who have already worked or are working on OCR. So we would give a formal vote of thanks to everyone, who directly/indirectly have helped us.

ABSTRACT/ Outline

The work presented here is about the Optical Character Recognition of handwritten characters using python, which has currently progressed a lot due to its multiple applications in every area possible. The aim is code can detect typed/ handwritten characters and gives output in machine-editable text. Also, we have mentioned in detail the procedure of OCR, its working and few techniques for preprocessing of raw image.

CONTENTS

Appendix – A List of Useful Websites

1. https://docs.opencv.org/2.4/modules/ml/doc/k_nearest_neighbors.html

2. https://docs.opencv.org/3.3.0/d7/d4d/tutorial_py_thresholding.html

3. https://docs.opencv.org/2.4/modules/ml/doc/k_nearest_neighbors.html

4. https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html

REFERENCES

1. Timoshenko, S. and Woinowsky-Krieger, S., *Theory of Plates and Shells*. Second Edition, McGraw-Hill, 1959.
2. Belytschko, T., Lu, Y.Y. and Gu, L., "Element-free Galerkin Methods". *International Journal for Numerical Methods in Engineering*; Vol. 37, 1994, pp. 229-256.
3. Ma, H.T. and Kanok-Nukulchai, W., "On the application of assumed strain methods", *Proceedings of the second East Asia-Pacific Conference on Structural Engineering & Construction (EASEC-2)*, Chiang Mai, Thailand, January 11-13, 1989.
4. Cough, R. W. and Penzien, J., "Response to a general dynamic loading", Chapter 7, *Dynamics of Structures*, McGraw-Hill, Singapore, 1975.

(The titles of books, journals and conference proceedings should be in italic)

Chapter 1 | Introduction

1.1 General

Optical Character Recognition (OCR) is procedure of converting images which maybe handwritten/typed to be recognized digitally by machine learning concepts.

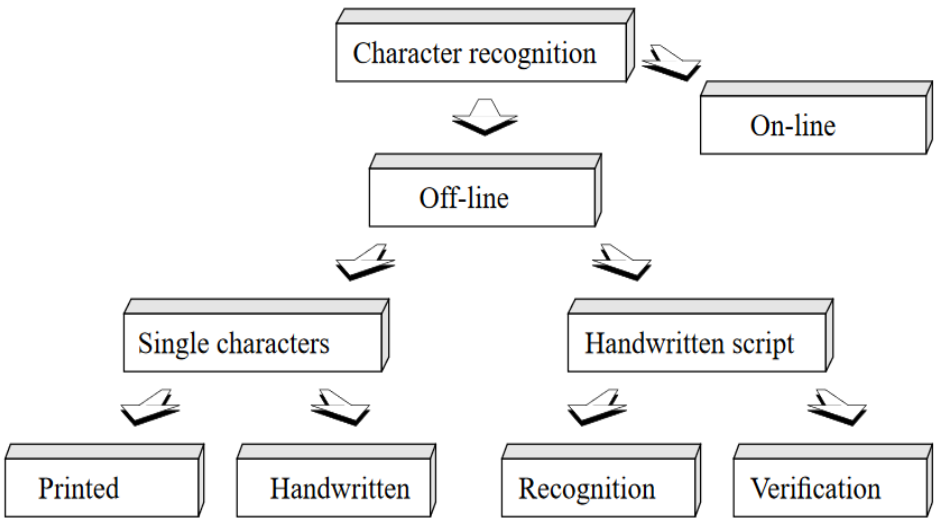


Fig-1.1 Areas of Optical Character Recognition

1.2 About OCR

2 Optical character recognition is an application of computer vision field. Traditional way of entering a piece of information is to type the document with help of keyboard. The best alternative is automatic identification. Different technologies for the recognition purposes exist. Following are the applications of computer vision.

1.3 Implementations of Automatic Recognition

- Recognition of Speech
- Frequency of Radio operation
- Visio System
- Magnetic Strip
- Scanning of Bar codes
- Magnetic Ink
- Reading of optical mark
- Recognition of optical characters

1.4 Objective of Study

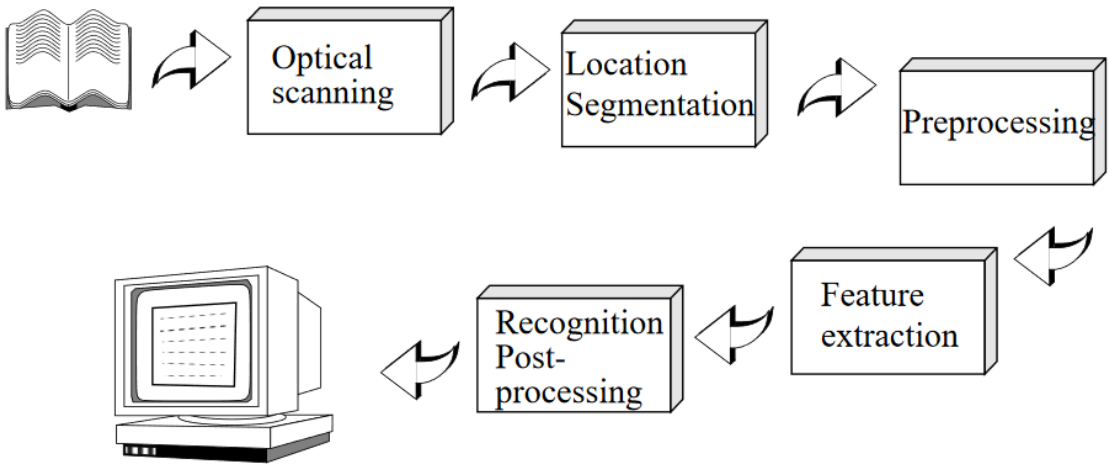
The objective of the report is to give a brief introduction to OCR and our work is about how to implement OCR in python with sequence of steps which are described in detail in further chapters.

1.5 Scope of work

The implemented code is capable to recognize handwritten characters on a piece of paper with considerable accuracy and output is given by the machine. We have also surveyed about work carried on till today, analyzed the flaws in the different implementations and techniques being used to achieve it.

Chapter 2 | Literature Survey

2 Components of OCR



2.1 Optical Image Scanning

- In this process digital image of a document is captured and preprocessing is done which are mentioned in the further section.
- Scanning can be done by any mechanical machine which is capable to click pictures.
- In OCR optical scanners are used which has the ability to convert the image directly to grayscale. Hence lessens the work of a software to do so.
- Converting the image into black and white image i.e. bi-level format is known as thresholding.
- A threshold means the limiting value which categorizes the properties. In this the intensities of the image's pixel is judged and based on the decided threshold are decided whether to assign a 0 intensity or 255 intensity value to that pixel.

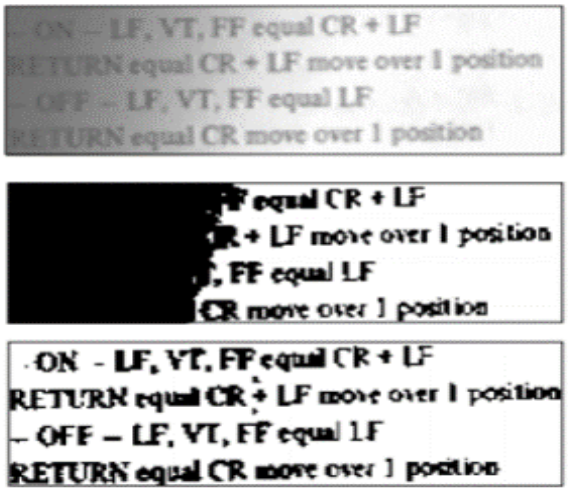


Fig-2.1 Optical Scanning

2.2 Segmentation

- To segment means to isolate characters or words in a given document.
- Most of scanners segment the words by characters.
- Separation is done usually by separating each connected component.
- Problems:

1. To extract characters which are touched and fragmented.
2. Separate the noise from the text.
3. Considering graphic or geometry as text, by mistake.
4. Considering text as graphic or geometry, by mistake.

2.3 Pre-processing

- The image scanned by the scanner may have some defects.
- So to filter out the noise and defects pre-processing is applied to the scanned image.
- Smoothing of an image implies both filling the voids and thinning the character width. Filling eliminates small breaks gaps and holes in the digitized characters and thinning reduces the width of the character.
- Whereas thinning reduces the width of the line.
- Whereas thinning reduces the width of the line.
- Normalization is applied to the scanned image to rotate the image at an appropriate angel.

Fig-2.2 Normalization

2.4 Feature Extraction

- The objectives of feature extraction is to highlight the features which are useful for identifying symbol. This comes under the subject called pattern recognition in computer science.
- Techniques for extraction of such features are often divided into three main groups.

1. Point distribution.
2. Transforming and expanding.
3. Analyzing Structure.

1.1.1 2.4.1 Point Distribution

- In this technique, the statistics of point distribution is considered which help in extraction of features.
- There is negligible effect of distortions and making variations in style. Some of the typical techniques within this area are listed below.

1. Zoning: The rectangle delineating the character is isolated into a few covering, or non-covering, districts and the densities of dark focuses inside these locales are figured and utilized as highlights.

2. Moments: The snapshots of dark focuses about a picked focus, for instance the focal point of gravity, or a picked organize framework, are utilized as highlights

2.4.2 Crossing and Distances

- In this features of a symbol are identified by noting the number of times a vector crosses the symbol. This vector should be in specific direction and should be in same direction for all the data.

This technique is oftenly used because it's complexity is less and it's fast as it creates a feature vector of less dimensionality.

2.4.3 Characteristic Loci.

- This technique is some what the same as previous one. It uses two lines, horizontal and a vertical line.
- The number of times the line segments of the symbol are intersected with these vectors are noted as feature of the image.

Fig-2.3 Characteristic Loci.

2.4.4 Transformation and Series

- As we can observe that these technique generates feature vector with less dimensionality. Only 2 dimensional feature vector which stores the number of intersection eg. <234,567>
- Features of this kind remains unaffected from the skewing of images.
- The transformations used may be Fourier, Walsh, Haar, Hadamard, Karhunen-Loeve, Hough.

2.5 Post-Processing

- After segmentation we need to extract the features out of the image.
- So there are two ways to extract the features.

1. Boundary based: In this method features of an image are extracted using the boundary of an object in images.
2. Region based: In this method the features are extracted based on the region within the boundary of the image. i.e colour ratio at different region and all.

- We have already seen the methods for feature extraction from region of images.
- Now we may focus on feature extraction methods using boundary of the image.

2.5.1 Border based extraction- Chain code

- Chain code is the technique used to extract the features from an image using the border of the image.
- In this technique the pixels are marked around the border.
- Then the segment connecting this pixels are given a code according to the direction in which the segment is facing.
- Following are the two conventions in which codes are assigned to the line segment in which direction the line is faced to.

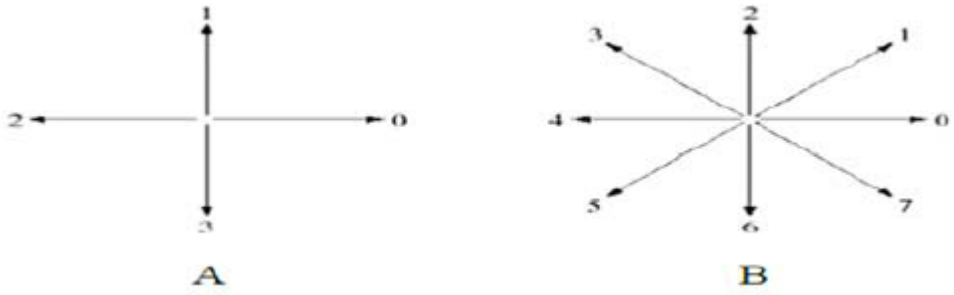
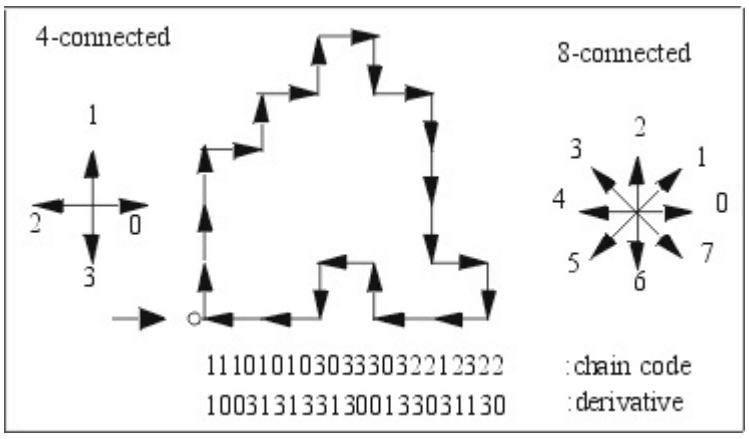


Fig-2.3 Border based extraction

- In the figure of previous slide the diagram A shows the representation of line segments in four different directions.
- In the B figure the line segments with extra directions like south-east, south west and all are added to the set and numbers are assigned to them.
- Now whenever the pixels of the outlined images are connected, one of the possible configuration of the line segment is used and an vector is considered in which the code of the line used is noted down.
- The traversal of the pixels can be in clock wise or anti clock wise.

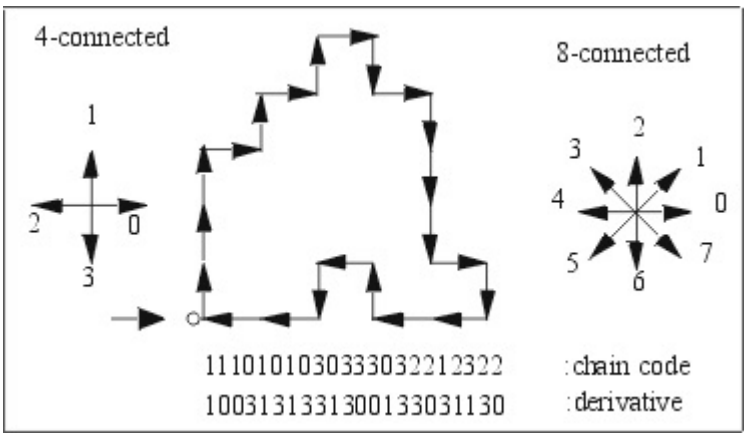


2.5.1.1 Problems in the above method

- If the start point of the image Is varied then the vector formed may be different.
- Hence classifier will not be able to identify the given figure.
- So we used the circular representation of the vector. In this the max value possible with the given vector is drawn out. And store in the knowledge base.
- And the same technique is applied to the images which are to be classified.
- Ex. We have feature vector <1, 2, 3> then we may store the vector <3, 1, 2> as the feature vector.
- Another problem in chain coding is that if we change the orientation of the shape again it will give the wrong vector.
- Hence object with the same shape are not identified if their orientation is varied.
- So, solution to this problem is the differential code chain.

2.5.2 Differential code chain

- In this technique we consider the number of rotations required to bring a certain direction segment to certain orientation and store it as a form of vector.
- First the consecutive direction vectors are found and then the rotations to be performed for the previous one to the current one is done and considered as a vector. And this rotations are in clock wise or anti – clock wise.



- As shown in the figure the shape is produced by four direction line segments. And below the figure there is the vector representation of it.
- First vector representation is of simple chain code.
- Second one is of differential chain code. Observe that we start at the position shown by the arrow. The direction is encoded with 1.
- Then after the next point is also in the direction of 1. So the rotation needed from the previous vector is 0 as they both are same.
- Notice at the third point the direction gets changed and the previous direction was 1 and to get the direction 0 we need to rotate the direction 3 times in anti-clock wise.
- Hence this is the method through which we can eliminate the problem of orientation of an object.

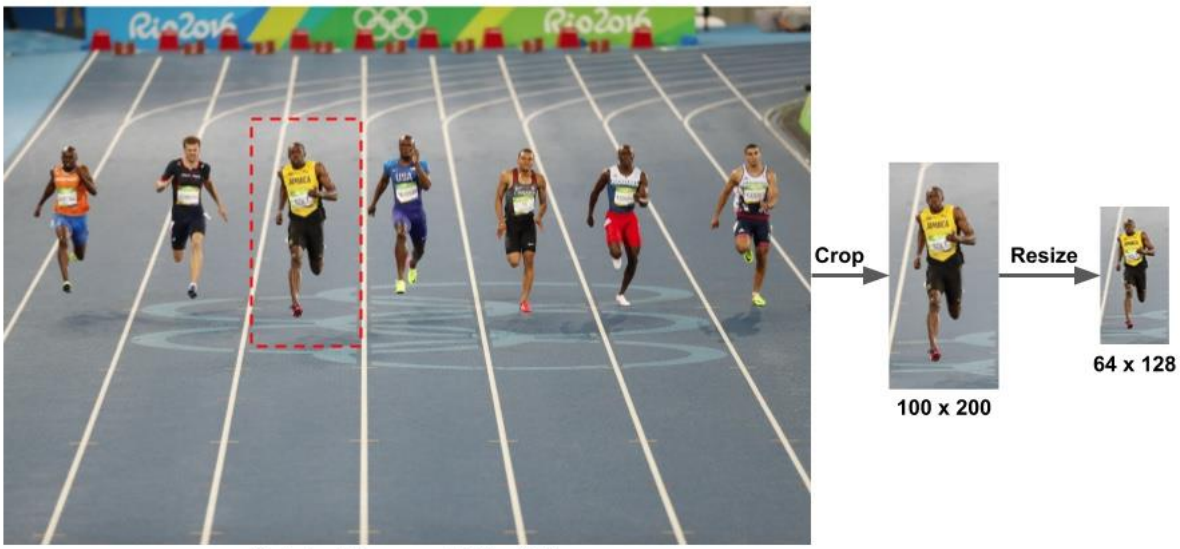
Chapter 3 | Histogram of Oriented Gradients

3.1 Feature Descriptor

- An image or a patch of image is represented in its simplified form by extraction of features and ignoring the extraneous information.

3.2 Calculating HOG

3.2.1 Pre-Processing



- Selecting the patch size and cropping the required area.
- Cropped image to be resized to 64 x 128p (acc. To requirement).

3.2.2 Calculate Gradient Images

-1	0	1

-1
0
1

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \arctan \frac{g_y}{g_x}$$

- The above kernels are used to filter and achieve horizontal and vertical gradients in order to calculate the hog.

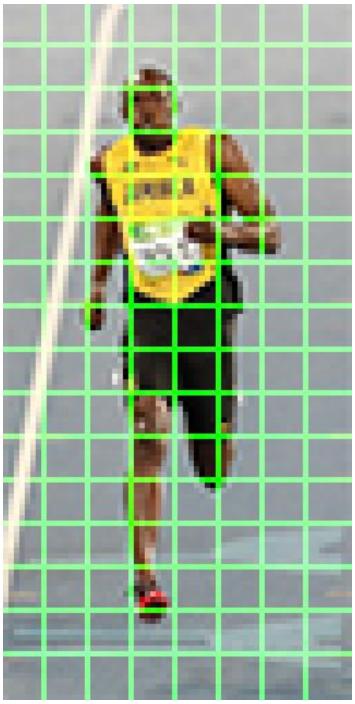
- The formulae to find the direction and magnitude of gradient.

3.2.3 Gradients for the patch

X gradient absolute value	Y gradient absolute value	Gradient Magnitude

The gradient image removes a lot of non-useful information which may not help in determination of objects.

3.2.4 HOG in 8 x 8 cells

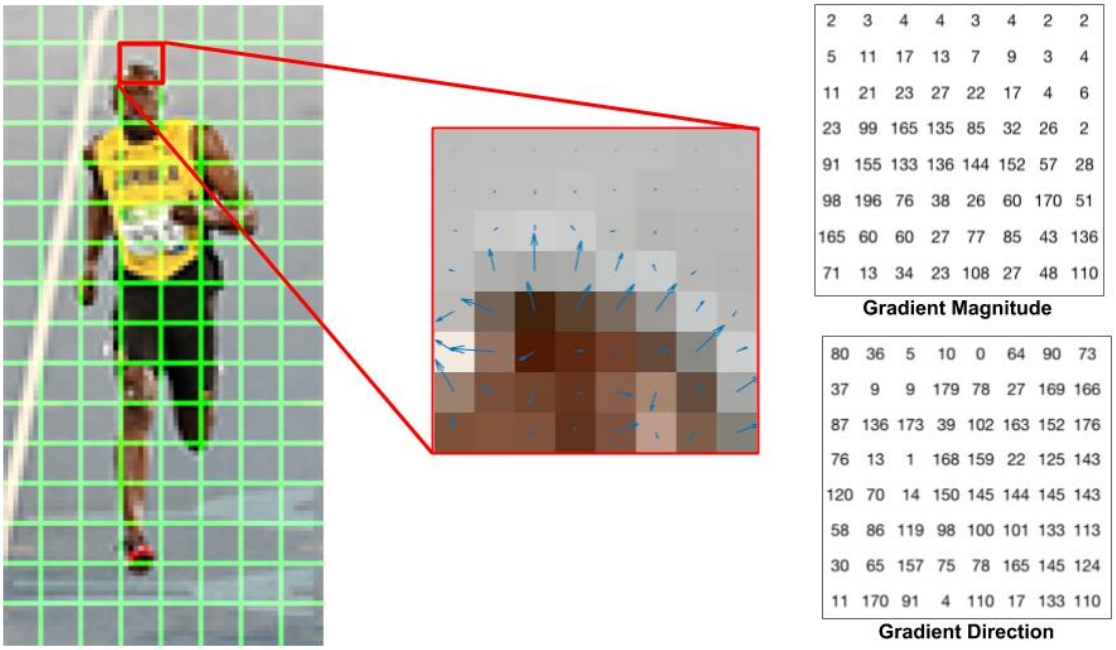


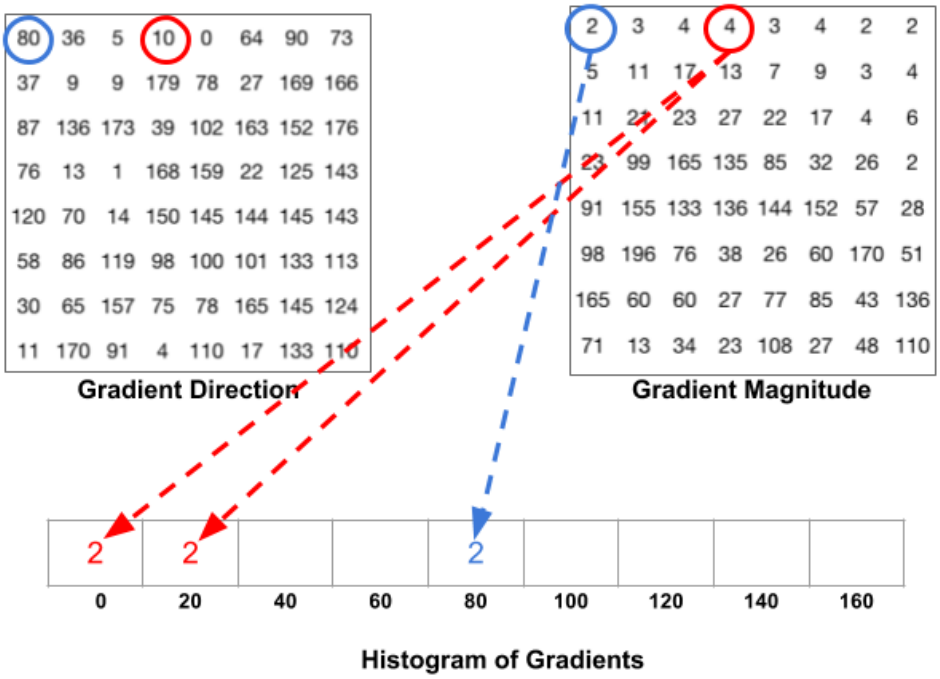
WHY 8x8?

- ✓ Feature descriptor provides compact information for the patches.
- ✓ 8x8x3=192 pixels.
- ✓ Gradient (vector) contains 2 values per pixel- magnitude and direction.
- ✓ Hence, 8x8x2=128 numbers.
- ✓ Represented in 2 8x8 grids.

Now we apply normalization, considering 2 x 2 blocks consecutive and find out new average of hog values.

By applying normalization , the histogram is not affected by change in the lightning and therefore making it more stable.





- The gradient at the pixel shown using red colored circle has an angle of 10 degrees and magnitude of 4. Since 10 degrees is between 0 and 20, the vote by the pixel, splits evenly into the two bins.

- If the value is greater than 160 and less than 180, we can wrap it around and value contributes to histogram of gradients accordingly by getting the difference value as shown in below figure.

3.2.5 Calculate HOG feature vector

1. To figure the last element vector for the whole picture fix, the 36×1 vectors are connected into one monster(giant) vector.
2. What number of places of the 16×16 pieces do we have? There are 7 even and 15 vertical positions making an aggregate of 7 x 15 = 105 positions.
3. Each 16×16 square is mapped to a 36×1 vector. So when we connect them all into one mammoth vector we get a 36×105 = 3780 dimensional vector.

Chapter 4 | Implementation in python

4.1 Image Blurring(Image Smoothing)

It can be achieved by applying a low-pass filter kernel. Noise removal is the main task to be done by this process. Below mentioned are the Blurring techniques in openCV:-

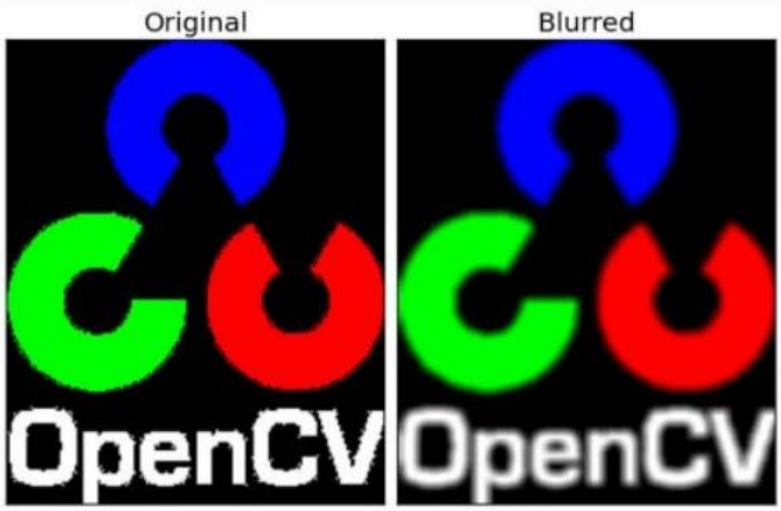
4.1.1 Averaging

Done by applying a normalized box filter on the image and it does simple averaging of the pixel values covered under the kernel.

A 3 x 3 filter looks like :

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

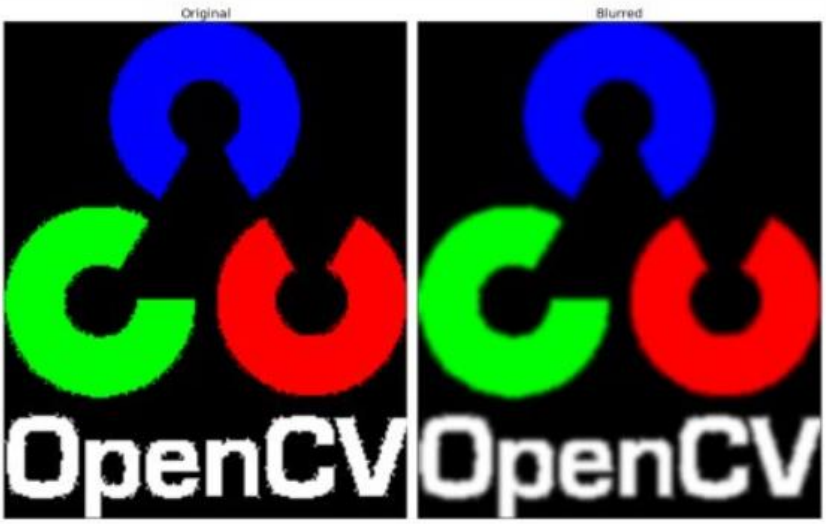
Result:



4.1.2 Gaussian Filtering

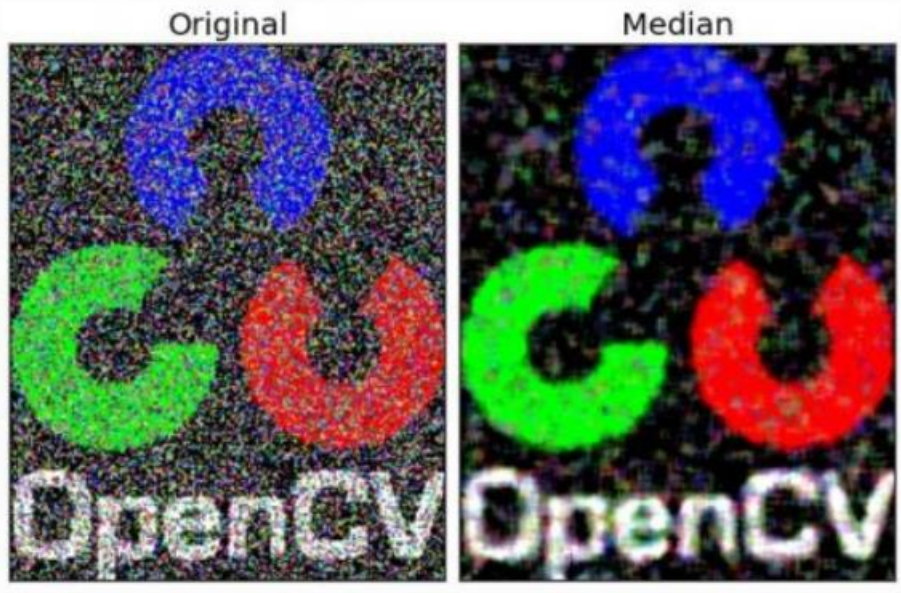
The height and width of the kernel is specified which should be positive and must be odd. Standard deviations in X and Y are to be inputted. If only S.D(X) is given S.D(Y)= S.D(X).

Result:



4.1.3 Median Filtering

In this type of filtering, median is calculated and central values of pixel is replaced with the median values. Most effective in removing salt and pepper noise.

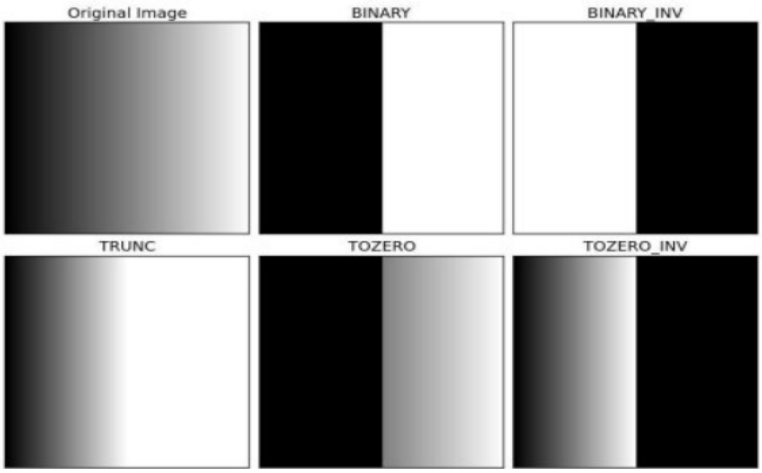


4.2 Image Thresholding

4.2.1 Simple Thresholding

A threshold value is decided, and the values greater than that threshold are assigned some value (say white) and less than

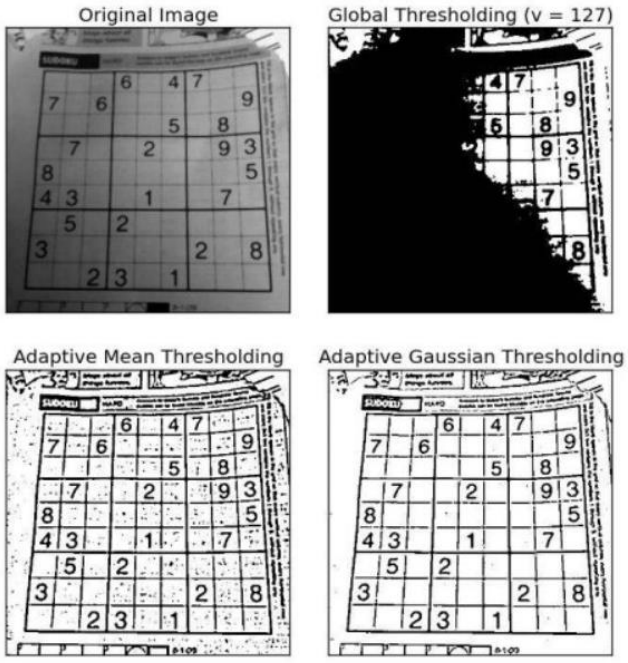
threshold are assigned different value (say black). Cv2.threshold function is used for thresholding having parameters as grayscale image and a user defined threshold value.



image

4.2.2 Adaptive Thresholding

In the above method, we use a user defined threshold value which may not prove to be a good method, if the image has different lightning small areas and are diverse. In adaptive thresholding, different threshold for different regions, which yields better result.



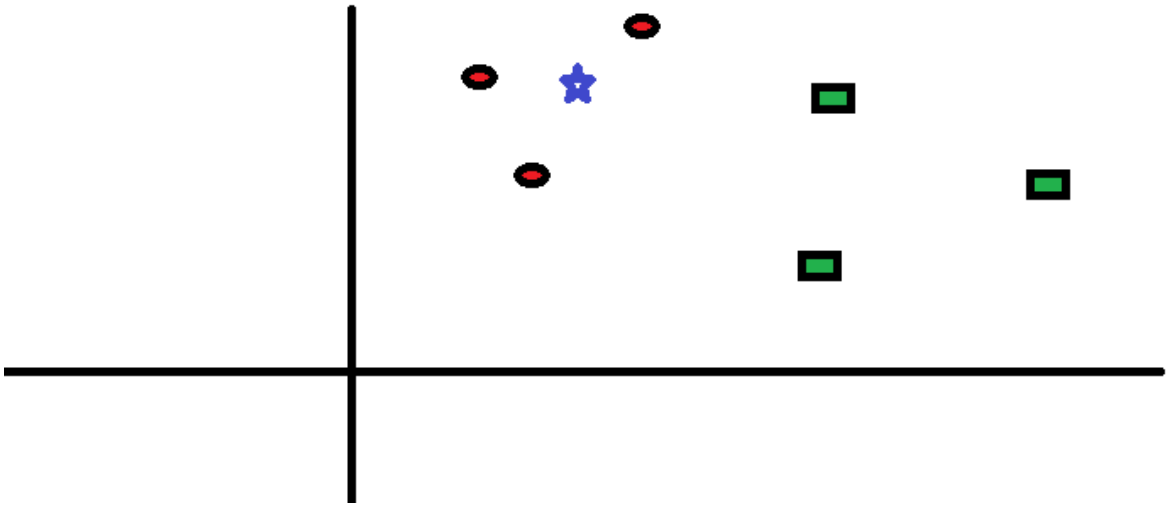
image

4.3 K Nearest Neighbor

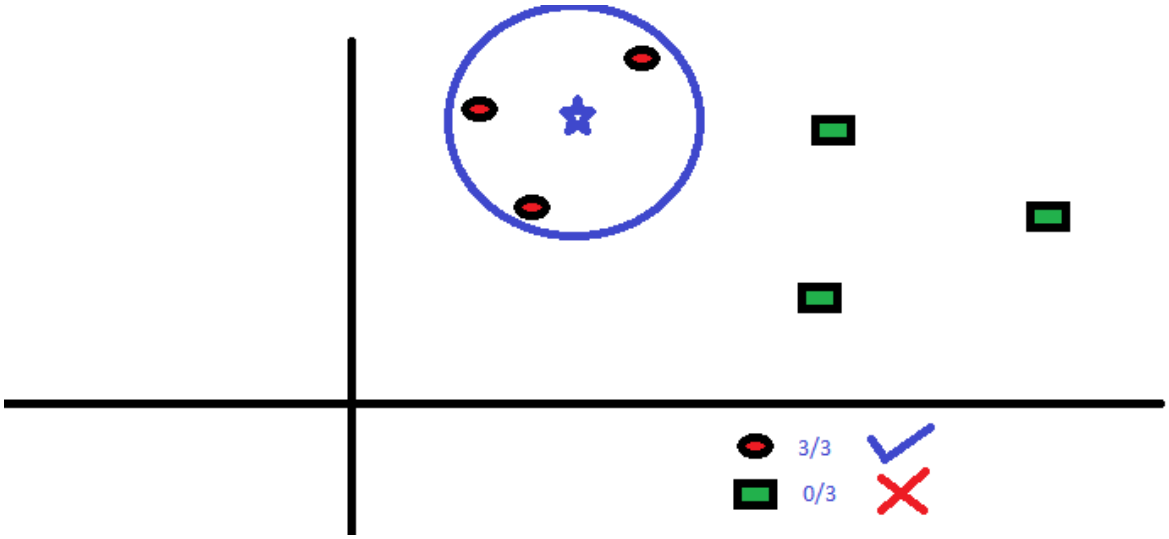
kNN can be used both for classification and prediction, here we will use it for prediction.

4.3.1 How it works?

Consider below figure of Red and Green objects.

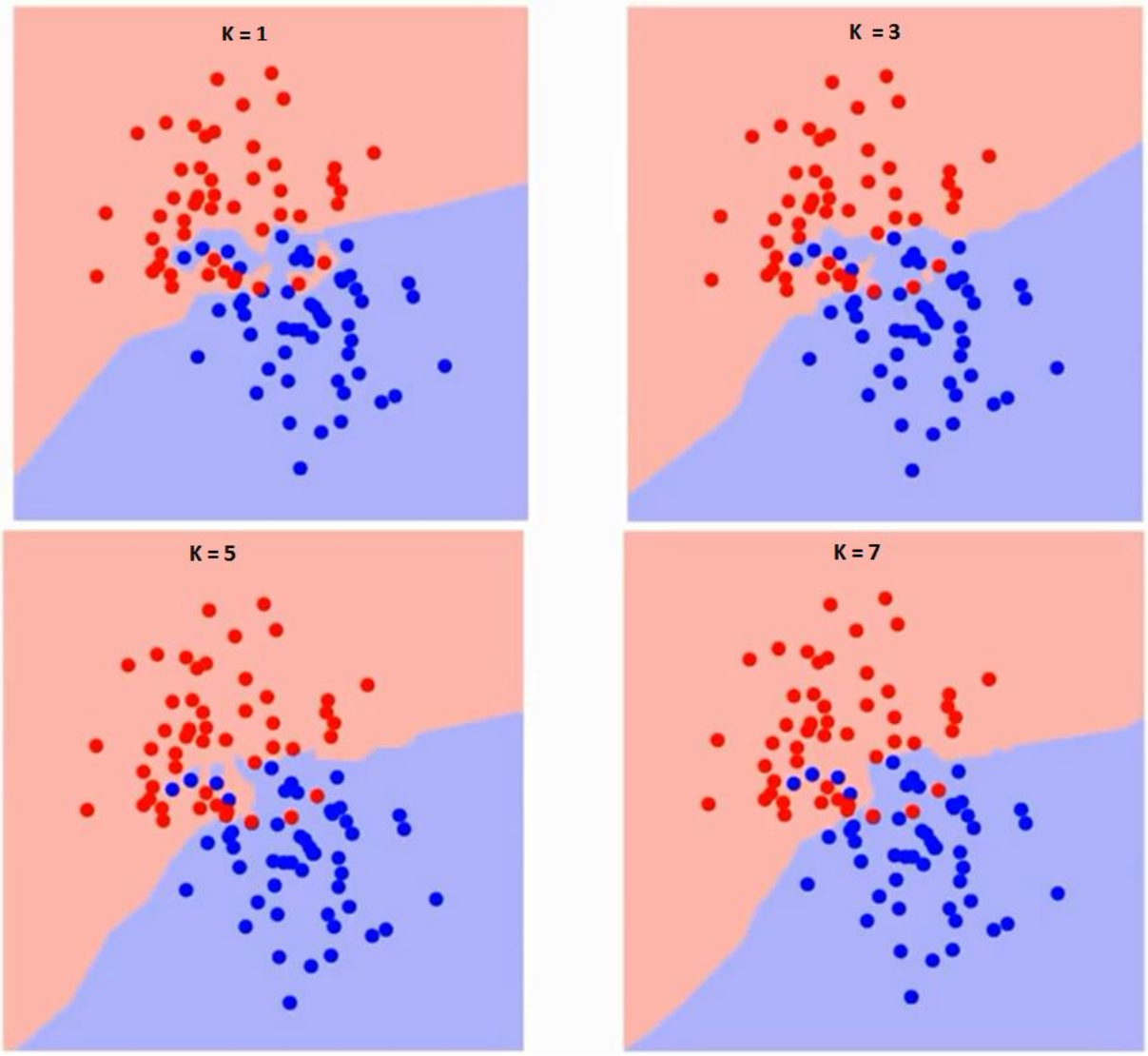


The blue star is to be classified, around that blue star is all reds so, using kNN it will be classified to class of reds.



4.3.2 How to choose k-factor?

If K value is given we can make boundary of each class. The below figure depicts the variations by making change in value of k.



Chapter 5 | About code

```
imgGray=cv2.cvtColor(trainingimg,cv2.COLOR_BGR2GRAY)     #get grayscal
imgBlurr=cv2.GaussianBlur(imgGray,(5,5),0)       #get blurred image

imgThresh = cv2.adaptiveThreshold(imgBlurr,
                                  255,
                                  cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                  cv2.THRESH_BINARY_INV,
                                  11,
                                  2)
```

The above code snippet are functions of openCV used to apply GaussianBlur and adaptiveThreshold which were discussed in earlier chapters.

```
56      npaClassifications=npaClassifications.reshape((npaClassifications.size,1))
57
58      kNearest=cv2.ml.KNearest_create()       #create the object of KNearest node.
59
60      kNearest.train(npaFlattenedImages, cv2.ml.ROW_SAMPLE, npaClassifications)
```

The above snippet is used to apply kNearestNeigbour classifier available in machine learning package for openCV.

```
cv2.rectangle(testimg,                                          # draw rectar
              (contourWithData.intRectX, contourWithData.intRectY),     # u
              (contourWithData.intRectX + contourWithData.intRectWidth,
               contourWithData.intRectY + contourWithData.intRectHeight),
              (0, 255, 0),              # green
              2)                        # thickness
```

The above snippet is used to create a rectangle around the character which is being scanned and has area >100, according to our conditions of it being a valid character.

Chapter 6 | Summary and Conclusion

6.1 Summary

We would like to summarize our work here, we implemented an OCR for handwritten characters, using kNN. Thus we successfully implemented OCR using python considering all the preprocessing and post processing before and after its application on image respectively

6.2 Conclusion

We hereby conclude to end of our Minor project and you can find the code on our github accounts:-

www.github.com/milindbachani45/Optical-Character-Recognition
www.github.com/AkashChaudhary1996/Optical-Character-Recognition