OPTIC DISC DETECTION

TAHERA AKTAR LASKAR (16/368)

TASNIM AHMED (16/371)

TAUSEEF CHOUDHURY (16/361)

HIRAK JYOTI BUNGRUNG (16/378)

ABHIJIT DEURI (16/376)

Dissertation submitted in partial fulfilment of the requirements for the degree of

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE ENGINEERING

Of GAUHATI UNIVERSITY

24th JUNE 2019 – 19TH JULY 2019

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

ASSAM ENGINEERING COLLEGE

GUWAHATI - 781013

OPTIC DISC DETECTION

TAHERA AKTAR LASKAR (16/368)

TASNIM AHMED (16/371)

TAUSEEF CHOUDHURY (16/361)

HIRAK JYOTI BUNGRUNG (16/378)

ABHIJIT DEURI (16/376)

Dissertation submitted in partial fulfilment of the requirements for the degree of

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE ENGINEERING

Of GAUHATI UNIVERSITY

24th JUNE 2019 – 19TH JULY 2019

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

ASSAM ENGINEERING COLLEGE

GUWAHATI - 781013

OPTIC DISC DETECTION

Bona fide record of work done by

TAHERA AKTAR LASKAR	**16/368**
TASNIM AHMED	**16/371**
TAUSEEF CHOUDHURY	**16/361**
HIRAK JYOTI BUNGRUNG	**16/378**
ABHIJIT DEURI	**16/376**

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE AND ENGINEERING

of Gauhati University

June -July 2019

.....	.....
Mr. Prakash J	**Dr. Sudha Sadasivam G**
Mentor	Head of the Department

Certified that the candidate was examined in the viva-voce examination held on …….

.....	.....
(Internal Examiner)	(External Examiner)

CONTENTS

CHAPTER **Page No.**

ACKNOWLEDGEMENT

The internship opportunity we had with PSG College of Technology was a great chance for learning and professional development. Therefore, we consider ourselves very lucky as we were provided with an opportunity to be a part of it. We are also grateful for having a chance to meet so many wonderful people and professionals who led us through this internship period.

A very sincere thanks to Dr. K. Prakasan, Principal, PSG College of Technology for providing a wonderful opportunity to develop and complete the project.

A sincere thanks to Dr. Sudha Sadasivam G., Professor and Head, Department of Computer Science and Engineering for her constant support and help.

Special thanks to Mr. J Prakash for mentoring us throughout the course of this project and providing invaluable support and advice on how to proceed through each step of the project development lifecycle.

We perceive this opportunity as a big milestone in our career development. We will strive to use gained skills and knowledge in the best possible way, and will continue to work on their improvement, in order to attain desired career objectives.

SYNOPSIS

An efficient detection of Optic Disc in retinal images is the fundamental step in an automated retinal image analysis system. Optic Disc detection and segmentation helps in identification of diabetic retinopathy and glaucoma in earlier stages. In this project, four different clustering algorithms are used for the detection and segmentation of Optic Disc from retinal images Clustering is a powerful technique that has been reached in image segmentation. The cluster analysis is to partition an image data set into a number of disjoint groups or clusters. Input image is first pre-processed using resizing, dilation and blurring on the green color band image. The cluster with maximum intensity is filtered using connected component to segment out the optic disc.

These clustering methods are tested on **High Resolution Fundus Dataset** which contains 45 retinal images, out of which 15 are healthy, 15 are affected by diabetic retinopathy and 15 are affected by glaucoma. These methods offer a successful detection of Optic Disc which may help in diagnosis of various retinal abnormalities.

CHAPTER 1

INTRODUCTION

Image Processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics or features associated with that image.

Image processing basically includes the following three steps:

- Importing the image via image acquisition tools;
- Analyzing and manipulating the image;
- Output in which result can be altered image or report that is based on image analysis.

There are two types of methods used for image processing namely, analog and digital image processing. Analogue image processing can be used for the hard copies like printouts and photographs. Digital image processing techniques help in manipulation of the digital images by using computers. Digital image processing allows the use of much more complex algorithms, and hence, can offer both more sophisticated performance at simple tasks, and the implementation of methods which would be impossible by analog means. It allows a wide range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing.

Applications of image processing:

The field of image processing has significantly improved in recent times and extended to various fields of science and technology. Some of the applications of image processing are:

1. **Intelligent Transportation Systems**: This technique can be used in Automatic number plate recognition and Traffic sign recognition.

2. **Remote Sensing**: For this application, sensors capture the pictures of the earth's surface in remote sensing satellites or multi – spectral scanner which is mounted on an aircraft. These pictures are processed by transmitting it to the Earth station. Techniques used to interpret the objects and regions are used in flood control, city planning, resource mobilization, agricultural production monitoring, etc.

3. **Moving object tracking**: This application enables to measure motion parameters and acquire visual record of the moving object. The different types of approach to track an object are:
    - Motion based tracking
    - Recognition based tracking

4. **Defense surveillance:** Aerial surveillance methods are used to continuously keep an eye on the land and oceans. This application is also used to locate the types and formation of naval vessels of the ocean surface.

5. **Biomedical Imaging techniques**: For medical diagnosis, different types of imaging tools such as X- ray, Ultrasound, computer aided tomography (CT) etc. are used.
6. **Automatic Visual Inspection System:** This application improves the quality and productivity of the product in the industries.
    - Automatic inspection of incandescent lamp filaments – This involves examination of the bulb manufacturing process.
    - Automatic surface inspection systems – In metal industries it is essential to detect the flaws on the surfaces.
    - Faulty component identification – This application identifies the faulty components in electronic or electromechanical systems.

CHAPTER 2

2.1 SOFTWARE REQUIREMENTS

- Language: Python
- IDE: Spyder

Additional Libraries:

- **OpenCV** (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. It was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and machine learning algorithms.

- **NumPy** is a library for the Python Programming Language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

- **Scikit-learn** (formerly **scikits.learn**) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

- **Matplotlib** is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

- **Pandas** is an open source library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

CHAPTER 3

LITERATURE REVIEW

3.1 INTRODUCTION

As the world's population has drastically increased, the number of people suffering from glaucoma, or those suspected to have glaucoma, has increased too. Glaucoma is the second leading cause of blindness after cataract, with ~60 million cases reported worldwide in 2010. It is estimated that by 2020, about 80 million people will suffer from glaucoma. Glaucoma is a chronic eye disease, in which the optic nerve is gradually damaged. If glaucoma is left untreated loss of vision occurs gradually, potentially leading to blindness. Therefore, diagnosing glaucoma at early stages is extremely important for proper management and successful treatment and control of the disease. The detection and diagnosis of glaucoma are related to tracing the changes in the optic cup which is a portion of optic disc (OD). The optic disc is a point in the eye where the optic nerve fibres leave the retina. It is a vertical oval with average dimension of 1.76 mm horizontally and 1.92 mm vertically. Segmenting the optic disc (OD) is an important and essential step in creating a frame of reference for diagnosing glaucoma. Segmentation can be defined as the classification of all the picture elements or pixels in an image into different clusters that exhibit similar features. It involves partitioning an image into groups of pixels which are homogeneous with respect to some criterion. The groups are called segments. The main components of the retina are blood vessels, optic disc and optic cup. The blood vessels, disc and cup merge in the image, making the segmentation more demanding. Determining the cup– disc ratio (CDR) is essential for detecting

the disease for which segmentation of disc and cup from the retinal images is necessary. Generally there is no unique method or approach for image segmentation. Clustering is a powerful technique that has been reached in image segmentation. Cluster analysis is to partition an image data set into a number of disjoint groups or clusters. The clustering methods such as K means, Fuzzy c mean (FCM), Density-based spatial clustering of applications with noise (DBSCAN) and Agglomerative hierarchical have been proposed.

Segmentation of optic disc and cup from the fundus retinal image helps in the detection of glaucoma. In addition to the visual field test and intra-ocular pressure measurement, precise measurement of the disc and cup areas as well as the cup to disc ratios is important for accurate diagnosis of glaucoma. The evaluation of the appearance of optic disc is central to the diagnosis and treatment of glaucoma. Glaucoma which is in most cases associated with an increase in intra-ocular pressure, often produces additional pathological cupping of the optic disc. As glaucoma advances, the cup enlarges until it occupies most of the disc area. The cup-to-disc (CDR) ratio is a measurement used in ophthalmology to assess the progression of glaucoma. The optic disc can be flat or it can have a certain amount of normal cupping. The CDR compares the diameter of the cup portion of the optic disc with the total diameter of the optic disc. The normal CDR is 0.3. A large CDR ratio may imply glaucoma or other pathology. In normal eye horizontal CDR is greater than vertical CDR and in glaucomatous vertical CDR is greater than horizontal CDR.

3.2 CLUSTERING

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them. Clustering is a type of unsupervised learning method. An unsupervised learning is a method in which we draw references from datasets consisting of input data without labeled responses.

Clustering is very important as it determines the intrinsic grouping among the unlabeled data present. There are no criteria for a good clustering. It depends on the user, what is the criteria they may use which satisfy their need.

CLUSTERING METHODS

1. **Density based methods:** density based methods has played a vital role in finding nonlinear shapes based on density. These methods consider the clusters as the dense region having some similarity and different from the lower dense region. These methods have good accuracy. Density- Base Spatial Clustering of Applications with Noise (DBSCAN) is the most widely used density based algorithm.
2. **Hierarchical method:** Hierarchical clustering involves creating clusters that have a predetermined ordering. The clusters formed in this method forms a tree type structure based on the hierarchy. New clusters are formed using the previously formed one. There are two types of hierarchical clustering:
    - Agglomerative (bottom up)
    - Divisive(top up)

3. **Partition methods:** The main objective of partition clustering algorithm is to divide the data points into K partitions. Each partition reflects one cluster. For example *K-means, CLARANS (Clustering Large Applications based upon randomized Search) etc.*
4. **Grid-based methods**: In this method the data space are formulated into a finite number of cells that form a grid-like structure. All the clustering operation done on these grids are fast and independent of the number of data objects example *STING (Statistical Information Grid), wave cluster, CLIQUE (Clustering In Quest)* etc.

3.2.1. DBSCAN CLUSTERING TECHNIQUE

Density-based spatial clustering of applications with noise (DBSCAN) is a density-based clustering algorithm. This algorithm finds core samples of high density and expands clusters from them. It is good for data which contains clusters of similar density.

Algorithm :

Input: Image to be clustered and global parameters Eps, MinPts.

Output: Clusters of objects.

- Arbitrary select a point P.
- Retrieve all points density-reachable from P with respect to Eps and MinPts.
- It P is a core point, a cluster is formed.

- If P is a border point, no points are density-reachable from P and DBSCAN visits the next point of the database.
- Continue the process until all of the points have been processed.

3.2.2. AGGLOMERATIVE CLUSTERING

Agglomerative clustering is a hierarchical clustering technique in which initially each data point is considered as an individual cluster. At each iteration, the similar clusters merge with other clusters until one cluster or K clusters are formed.

Algorithm:

- Compute the proximity matrix
- Assign each data point as a cluster
- Merge the two closest cluster and update the proximity matrix
- Repeat step 3 until k clusters are formed

3.2.3. K means Clustering

Kmeans algorithm is an iterative algorithm that tries to partition a dataset into *K* pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group**. It tries to make the inter-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum.

Algorithm:

1. Specify number of clusters K.
2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.
    - Compute the sum of the squared distance between data points and all centroids.
    - Assign each data point to the closest cluster (centroid).
    - Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

3.2.4. Fuzzy C-Means Clustering

In fuzzy clustering, every point has a degree off belonging to clusters, as in fuzzy logic, rather than completely belonging to just one cluster. Thus, points on the edge of a cluster, may be in the cluster to a lesser degree than points in the center of the cluster.

The fuzzy c-means algorithm attempts to partition a finite collection of elements X, into a collection of c fuzzy clusters with respect to some given criterion. The algorithm is based on the minimization of the following objective function:

$$J(U,V) = \sum_{i=1}^{n}\sum_{j=1}^{c}(\mu_{ij})^{m}\left\|\mathbf{x}_i - \mathbf{v}_j\right\|^2$$

Where, m (the Fuzziness Exponent) is any real number greater than 1. N is the number of data. C is the number of clusters. U_{ij} is the membership value which tells the degree to which the element x_i belongs to the j-th cluster. $\| x_i - v_j \|$ is distance of data point I to the current cluster center j.

Algorithm:

1. Assign an initial random centroid to each cluster (Group).
2. Compute the distance between each point and the cluster centre using a simple algorithm.
3. Based on distance between each point and the cluster centre, re-compute the membership function.
4. Based on the new membership function, re-compute the centroid.
5. If the difference between the original centroid and the next one is below a certain threshold value say, €, then the algorithm stops, else it continues till this condition is true.

CHAPTER 4

SYSTEM DESIGN

This chapter covers the system of the image pre-processing, clustering and segmentation of the Optic Disc.

4.1 SYSTEM ACHITECHTURE

The block diagram of the proposed system, which explains step by step implementation, is shown in the figure below. The input retinal images are taken from High Resolution Fundus (HRF) image dataset.

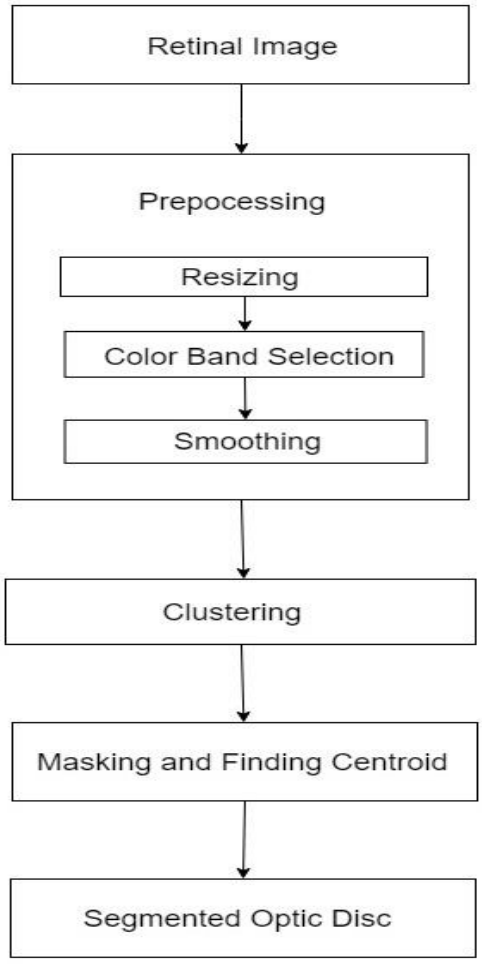


Fig: Block diagram of the implemented system

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1. Pre-processing the retinal image

The retinal images provided as input may contain variations of brightness or color information known as noise. To cope with such degraded images, the images are enhanced which improves visibility and perceptibility of image. The goal of pre-processing includes: removing the noise, enhancing contrast, sharpening or smoothing, elimination or retaining certain features in an image.

- Working with the original size of the retinal images could use high computational power and may not fit into certain space on a screen. So resizing is done to alter the size of the original image without cutting anything out.
  img = cv2.resize(img,(350,233))
  This function resizes the original image.
- Each pixel of the retinal images contains three color values, R, G, and B. They can be any numeric value between 0 and 255. It is not advisable to process with the colored image as such because it requires large computational power and it does not own distinct facts about anatomical and pathological structure in the retinal images. So the initial step is to separate the color bands from the retinal images. The green color band is better for the segmentation of OD as compared to red and blue color band. In the green color band, the image is of good contrast, so it is used for further processing.
  _, g,_ = cv2.split(img)
  This function splits the image and stores the green color band image.

- The green colour band image contains blood vessels and other noise. In order to remove these noise, Dilation and Gaussian Blur operations are used for smoothening the image and removing the blood vessels.

  kernel = np.ones((25,25), np.uint8)

  dilated_img = cv2.dilate(g, kernel, iterations=2)

  blur = cv2.GaussianBlur(ie, (23, 23), 10)

  These functions perform the Dilation and Gaussian Blur operations.

5.2 CLUSTERING FOR SEGMENTATION OF THE OPTIC DISC

After pre-processing the input retinal image, clustering is done for detecting the optic disc. Clustering divides the number of pixels into groups based on their similarities. Since optic disc is the brightest region of the input retinal image, so all the pixels forming the optic disc will possess similar properties and hence will fall in the same cluster. And this cluster is later segmented as the optic disc.

For clustering four algorithms are used –

- Agglomerative (Hierarchical clustering)
- K means (Partition clustering)
- Fuzzy C Means (partition clustering)
- DBSCAN (Density based clustering)

Agglomerative Clustering

Agglomerative clustering is accomplished using

sklearn.cluster.AgglomerativeClustering(*n_clusters=2, affinity='euclian', connectivity=None*)

Parameters:

- n_clusters- the number of clusters to find.
- Affinity- (default-“euclidean”) Metric used to compute the linkage
- Connectivity- Connectivity matrix. Defines for each sample the neighboring samples following a given structure of the data. This can be a connectivity matrix itself or a callable that transforms the data into a connectivity matrix, such as derived from k neighbors graph.

K means Clustering

K-means clustering can be accomplished via **cv2.kmeans()** function in OpenCV.

compactness,labels,centers = cv2.kmeans(z,2,None,criteria,10,flags)

Input parameters

1. **Samples**: It should be of **np.float32** data type, and each feature should be put in a single column.
2. **n clusters(K)** : Number of clusters required at end
3. **Criteria**: *It is the iteration termination criteria. When this criteria is satisfied, algorithm iteration stops. Actually, it should be a tuple of 3 parameters. They are (type, max_iter, epsilon):*
    - type - type of termination criteria
    - max_iter - An integer specifying maximum number of iterations.
    - epsilon - Required accuracy
4. **Attempts**: Flag to specify the number of times the algorithm is executed using different initial labellings. The algorithm returns the labels that yield the best compactness. This compactness is returned as output.

5. **Flags**: This flag is used to specify how initial centers are taken. Normally two flags are used for this. They are:
    - **cv2.KMEANS_PP_CENTERS**
    - **cv2.KMEANS_RANDOM_CENTERS**.

Output parameters:

1. **Compactness**: It is the sum of squared distance from each point to their corresponding centers.
2. **Labels**: This is the label array (same as 'code' in previous article) where each element marked '0', '1'.....
3. **Centers**: This is array of centers of clusters.

Fuzzy C Means Clustering

Fuzzy c-means clustering is accomplished via:

skfuzzy.cluster.**cmeans** **(***data***,** *c***,** *m***,** *error***,** *maxiter***,** *init=None***)**

Parameters :	**data : 2d array, size (S, N)** Data to be clustered. N is the number of data sets; S is the number of features within each sample vector. **c : int** Desired number of clusters or classes. **m : float**

Array exponentiation applied to the membership function u_old at each iteration, where U_new = u_old ** m.

error : float

Stopping criterion; stop early if the norm of (u[p] - u[p-1]) < error.

maxiter : int

Maximum number of iterations allowed.

init : 2d array, size (S, N)

Initial fuzzy c-partitioned matrix. If none provided, algorithm is randomly initialized.

Returns:

cntr : 2d array, size (S, c)

Cluster centers. Data for each center along each feature provided for every cluster (of the c requested clusters).

u : 2d array, (S, N)

Final fuzzy c-partitioned matrix.

u0 : 2d array, (S, N)

Initial guess at fuzzy c-partitioned matrix (either provided init or random guess used if init was not provided).

d : 2d array, (S, N)

Final Euclidian distance matrix.

jm : 1d array, length P

Objective function history.

p : int

Number of iterations run.

fpc : float

Final fuzzy partition coefficient.

DBSCAN Clustering

Implementation

class sklearn.cluster.**DBSCAN**(*eps=5.5*, *min_samples=37*, *metric='euclidean'*, *al gorithm='auto'*)

Parameters:

eps : *float, optional*

The maximum distance between two samples for one to be considered as in the neighborhood of the other. This is not a maximum bound on the distances of points within a cluster. This is the most important DBSCAN parameter to choose appropriately for your data set and distance function.

min_samples : *int, optional*

The number of samples (or total weight) in a neighborhood for a point to be considered as a core point. This includes the point itself.

metric : *string, or callable*

The metric to use when calculating distance between instances in a feature array. If metric is a string or callable, it must be one of the options allowed by **sklearn.metrics.pairwise_distances** for its metric parameter. If metric is "precomputed", X is assumed to be a distance

matrix and must be square. X may be a sparse matrix, in which case only "nonzero" elements may be considered neighbors for DBSCAN.

algorithm : ***{'auto', 'ball_tree', 'kd_tree', 'brute'}, optional***

The algorithm to be used by the Nearest Neighbors module to compute point wise distances and find nearest neighbors. See Nearest Neighbors module documentation for details.

5.3 MASKING

After clustering, the optic disc must be in the brightest region of the clustered image, so we mask out all other pixels of the image except the brightest pixels. Now, we find the centroid of the masked image using the function cv2.moments(). Using this centroid we draw a circle of fixed radius in the original image to segment out the optic disc.

Chapter 6

RESULTS

The dataset used in this project is HRF(High Resolution Fundus) dataset, which contains 15 images of healthy patients, 15 images of patients with diabetic retinopathy and 15 images of glaucomatous patients. Binary gold standard segmentation values are available for each image. Based on these gold standard values, number of hits and misses were calculated.

The performance of OD detection methods is assessed by comparing the OD center difference between manually labeled coordinates (X_{OD}, Y_{OD}) and detected coordinates (X_C, Y_C). The detected OD center considers it a hit if it satisfies the equation given below:

$$\sqrt{(X_{OD}-X_C)^2+(Y_{OD}-Y_C)^2} \leq R_{mean}$$

Where R_{mean} is the average of all the gold standard radii of the dataset. For the HRF dataset the average radius was 187.5 units in length.
If the equation is not satisfied, then it is considered as a miss.

The result of k means, Fuzzy C Means, Agglomerative and DBSCAN clustering is shown below. Fig 1 is the input image and fig 2 is the output image of the respective algorithms. The green circle in fig 2 represents the portion of the optic disc detected by the respective algorithm.

1. K Means

Fig 1

Fig2

2. Fuzzy C Means

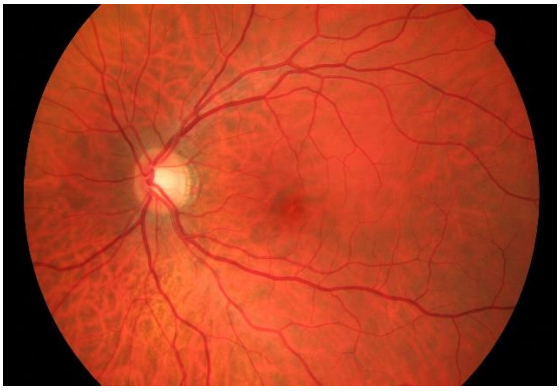


Fig 1

Fig2

3. Agglomerative

Fig 1

Fig 2

4. DBSCAN

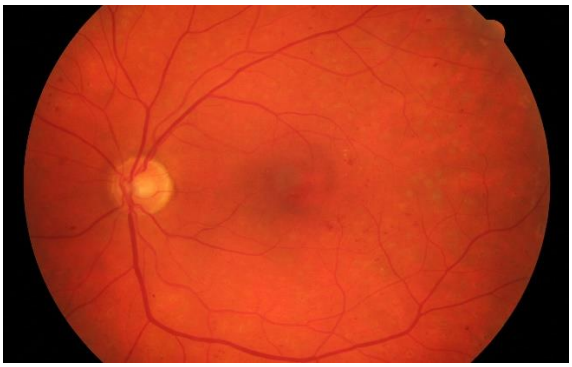


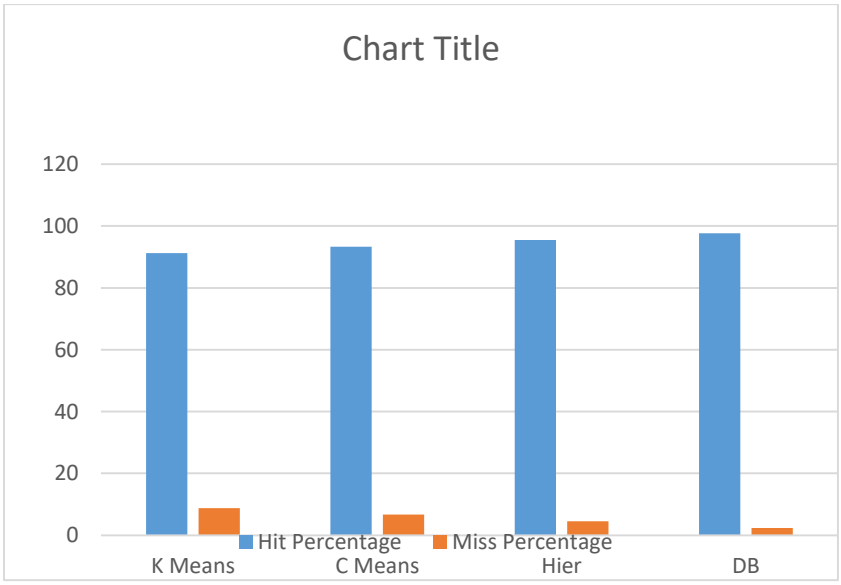
Fig 1

Fig 2

Comparative Analysis of Clustering Algorithms

Hit & Miss Evaluation:

ALGORITHMS	Hit Percentage	Miss Percentage
K Means	91.2	8.8
Fuzzy C Means	93.3	6.7
Agglomerative	95.5	4.5
DBSCAN	97.7	2.3



CHAPTER 7

Conclusion & Future Enhancement

In this project the K-means (KM), Fuzzy C-means (FCM), DBSCAN and Agglomerative algorithms were compared for their computing performance and clustering accuracy on High Resolution Fundus images.

The essential difference between fuzzy *c*-means clustering and standard *k*-means clustering is the partitioning of objects into each group. Rather than the hard partitioning of standard *k*-means clustering, where objects belong to only a single cluster, fuzzy *c*-means clustering considers each object a member of every cluster, with a variable degree of "membership".

K-Means is very sensitive to noise in the dataset whereas Hierarchical Clustering Algorithm is less sensitive to noise in the dataset.

From the implementations we can claim that the k-means can be used for its simplicity of implementation and for its convergence speed. K-means also produces relatively high quality clusters considering the low level of computation required. Fuzzy C-means gives better result for overlapped data set and comparatively is better than k-means algorithm. And lastly both hierarchical and DBSCAN produces the best results but it takes a higher computational time. So it is advisable to use these two.

Overall, we would describe our internship as a positive and instructive experience. This internship has been an excellent and rewarding experience. We have been able to meet and network with so many people that it will surely help us with opportunities in the future.

CHAPTER 8

Bibliography

[1] B. Vinoth Kumar, G.R. Karpagam, Yanjun Zhao, "Evolutionary Algorithm With Memetic Search Capability for Optic Disc Localization in Retinal Fundus Images", Intelligent Data Analysis for Biomedical Applications: Challenges and Solutions, Chapter 9, 15 January 2019.

[2] Neeraj Sharma, Dr. Amandeep Verma, "Segmentation and Detection of Optic Disc Using Kmeans Clustering", International Journal of Scientific & Engineering Research, Volume 6, Issue 8, August-2015

[3] Devarshi Naik , Pinal Shah, "A Review on Image Segmentation Clustering Algorithms", International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 3289 – 3293

[4] B.Sathya, R.Manavalan, "Image Segmentation by Clustering Methods: Performance Analysis", International Journal of Computer Applications (0975 – 8887), Volume 29– No.11, September 2011

[5] https://opencv-python-tutroals.readthedocs.io/en/latest/index.html

[6] https://www.analyticsvidhya.com/

[7] https://www.learnopencv.com/.

CHAPTER 9

APPENDIX

1. K-means

```
import cv2
import numpy as np
import glob
import time as time

path = 'C:/Users/DELL PC/python/opencv/kmeans/images/*.*'
no = 1
for file in glob.glob(path):
    start = time.time()
    print(file)
    im = cv2.imread(file)
    im = cv2.resize(im,(350,234))
    im1 = cv2.imread(file)
    im1 = cv2.resize(im1,(350,234))
    img = cv2.imread(file,0)
    img = cv2.resize(img,(350,234))
    _,g,_ = cv2.split(im)
    kernel = np.ones((25,25),np.uint8)
    blur = cv2.dilate(g, kernel, iterations = 2)
    blur = cv2.GaussianBlur(blur,(55,55),20)
    #blur = cv2.medianBlur(blur,35)
    z = blur.reshape(-1,1)
    z = np.float32(z)
    criteria = (cv2.TERM_CRITERIA_EPS +
cv2.TERM_CRITERIA_MAX_ITER, 100, 1.0)
    K = 17
```

```
ret,label,center=cv2.kmeans(z,K,None,criteria,100,cv2.KMEANS_RANDOM_CE
NTERS)
    center = np.uint8(center)
    res = center[label.flatten()]
    res2 = res.reshape(g.shape)
    max = np.max(res2)
    m = res2 < max
    im[m] = 0
    fname = 'segment/'+str(no)+'.jpg'
    cv2.imshow('original',im1)
    cv2.imwrite(fname,im)
    no=no+1
    cv2.imshow('blur',blur)
    cv2.imshow('segment',im)
    cv2.imshow('kmeans',res2)
    cv2.waitKey(1)
print('Running time: ')
rtime = time.time() - start
print(rtime)
cv2.waitKey()

cv2.destroyAllWindows()
```

2. Fuzzy C Means.

```
import numpy as np
import skfuzzy as fuzz
import cv2
import re
import glob
import time as time
numbers = re.compile(r'(\d+)')
def numericalSort(value):
    parts = numbers.split(value)
    parts[1::2] = map(int, parts[1::2])
    return parts
path = "../kmeans/images/*.*"
no=0
for file in sorted(glob.glob(path), key=numericalSort):
    start = time.time()
    def change_color_fuzzycmeans(cluster_membership, clusters):
        img = []
        for pix in cluster_membership.T:
            img.append(clusters[np.argmax(pix)])
        return img
    print(file)
    image=cv2.imread(file)
    rgb_img=cv2.resize(image, (350,234))
    image=cv2.resize(image, (350,234))
    kernel = np.ones((21,21),np.uint8)
    b,g,r=cv2.split(image)
    rgb_img=g
    rgb_img = cv2.dilate(rgb_img, kernel, iterations=2)
    rgb_img = cv2.GaussianBlur(rgb_img,(45,45),14)
    rgb_img = rgb_img.reshape((rgb_img.shape[0] * rgb_img.shape[1],1))
    img = np.reshape(rgb_img, (234,350)).astype(np.uint8)
    shape = np.shape(img)
```

```
    cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(rgb_img.T, 17, 2, error=0.05,
maxiter=100, init=None)
    new_img = change_color_fuzzycmeans(u,cntr)
    fuzzy_img = np.reshape(new_img,shape).astype(np.uint8)
    cv2.imshow('original',image)
    cv2.imshow('g',g)
    m = np.max(fuzzy_img)
    mask = fuzzy_img<m
    image[mask]=0
    cv2.imshow('fcm',fuzzy_img)
    cv2.waitKey(1)
    cv2.imshow('segment',image)
    fname = 'segment/seg'+str(no)+'.jpg'
    no=no+1
    cv2.imwrite(fname,image)
    cv2.waitKey(1)
print('Running time: ')
rtime = time.time() - start
print(rtime)
cv2.destroyAllWindows()
```

3. Hierarchical Agglomerative.

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import glob
import time as time
from sklearn.feature_extraction.image import grid_to_graph
from sklearn.cluster import AgglomerativeClustering
num=0
no=1
sum=0
for file in glob.glob('C:/Users/Hirak/Desktop/Summer Project/HRF Dataset/.'):
    orig_img = cv2.imread(file)
    rorig_img=cv2.resize(orig_img,(350,233))
    cv2.imshow('sdasdad',rorig_img)
    gray_img=cv2.cvtColor(orig_img,cv2.COLOR_BGR2GRAY)
    rescaled_img=cv2.resize(gray_img,(350,233))
    #smoothened_img=cv2.GaussianBlur(rescaled_img,(23,23),38)
    kernel = np.ones((39,39),np.uint8)#33,33
    dilation_img=cv2.dilate(rescaled_img,kernel,iterations = 3)
    dilation_img=cv2.erode(dilation_img,kernel,iterations = 2)
    dilation_img=cv2.GaussianBlur(dilation_img,(3,3),1)
    X = np.reshape(dilation_img, (-1, 1))
    # Define the structure A of the data. Pixels connected to their neighbors.
    connectivity = grid_to_graph(*dilation_img.shape)
    # Compute clustering
    print("Compute structured hierarchical clustering...")
    st = time.time()
    n_clusters = 15  # number of regions
    ward = AgglomerativeClustering(n_clusters=n_clusters,
linkage='ward',connectivity=connectivity)
    ward.fit(X)
    label = np.reshape(ward.labels_, dilation_img.shape)
    t=time.time()-st
```

```
    print("Elapsed time: ", t)
    print("Number of pixels: ", label.size)
    print("Number of clusters: ", np.unique(label).size)
    sum=sum+t
    # Plot the results on an image
    plt.figure(figsize=(5, 5))
    plt.imshow(dilation_img, cmap=plt.cm.gray)
    for l in range(n_clusters):
        plt.contour(label == l,colors=[plt.cm.nipy_spectral(l / float(n_clusters)), ])
    plt.show()
    m=np.max(dilation_img)
    mask=dilation_img<m
    rorig_img[mask]=0
    #plt.imshow(rorig_img, cmap= plt.cm.gray)
    cv2.imshow('asdsa',rorig_img)
    #cv2.imwrite('C:/Users/Hirak/Desktop/01_h_mask.jpg',rorig_img)
    if num==1:
        cv2.waitKey(15000)
    num=num-1
    fname='C:/Users/Hirak/Desktop/Summer
Project/Agglo_Segmented/img'+str(no)+'.jpg'
    cv2.imwrite(fname,rorig_img)
    no=no+1
    cv2.waitKey(1)
print('Total time')
print(sum)
cv2.waitKey()
cv2.destroyAllWindows()
```

4. DBSCAN

```
import cv2
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn import metrics
import pandas as pd
import glob
import re
numbers = re.compile(r'(\d+)')
def numericalSort(value):
    parts = numbers.split(value)
    parts[1::2] = map(int, parts[1::2])
    return parts
path = 'C:/Users/DELL PC/python/opencv/kmeans/images/*.*'
no=0
df = pd.read_csv("csv/size.csv", index_col =0)
for file in sorted(glob.glob(path),key =numericalSort):
    img=cv2.imread(file)
    img1=cv2.imread(file)
    img2 = cv2.imread(file)
    n = 0
    while(n<4):
        img = cv2.pyrDown(img)
        img1 = cv2.pyrDown(img1)
        if n<3:
            img2 = cv2.pyrDown(img2)
        n = n+1
    cv2.imshow('Original',img)
    kernel = np.ones((3,3),np.uint8)
    z=np.float32(img.reshape(-1,3))
    db = DBSCAN(eps=1.1, min_samples=7, metric = 'euclidean',algorithm
='auto').fit(z[:,:2])
    i = np.uint8(db.labels_.reshape(img.shape[:2]))
    m = np.max(i)
```

```
    mask = i < m
    img1[mask] = 0
    img1 = cv2.erode(img1, kernel, iterations = 1)
    img1 = cv2.pyrUp(img1)
    img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
    ret,thresh = cv2.threshold(img1,127,255,0)
    M = cv2.moments(thresh)
    if M["m00"] !=0:
        cX = int(M["m10"] / M["m00"])
        cY = int(M["m01"] / M["m00"])
        if cX < img1.shape[1]/2:
            cX = cX-9
        else:
            cX = cX+9
        cv2.circle(img2, (cX, cY), 25, (0,255,0), 2)
    else:
        cX = 0
        cY = 0
    fname = 'segmentdb/img'+str(no)+'.jpg'
    cv2.imshow('optic_disc',img2)
    cv2.imwrite(fname,img2)
    no=no+1
    cv2.imshow('i',img1)
    diameter = 0
    df1 = pd.DataFrame({"Diameter":
[diameter*8],"X_Center":[cX*8],"Y_Center":[cY*8]})
    df = df.append(df1, ignore_index = True)
    print("Center:", end=' ')
    print(cX*8, end=' ')
    print(cY*8)
    print("Diameter:", end=' ')
    print(diameter*8)
cv2.waitKey()
export_csv = df.to_csv (r'csv/size.csv')
cv2.destroyAllWindows()
```