A CHEAP AND FAST METHOD OF GRADING OPTICAL MARK RECOGNITION SHEET USING IMAGE PROCESSING

H. M. ABRAR FAHIM PATWARI
Roll: ASH1606037M

DEPARTMENT OF APPLIED MATHEMATICS

NOAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY

2022

A CHEAP AND FAST METHOD OF GRADING OPTICAL MARK RECOGNITION SHEET USING IMAGE PROCESSING

by

H. M. ABRAR FAHIM PATWARI
Roll: ASH1606037M

Project submitted in partial fulfilment of the requirements
for the degree of
Bachelor of Science

March 2022

ACKNOWLEDGEMENT

I want to thank Almighty Allah for giving me the chance to work on this project and finish it with the help of these amazing people of Applied Mathematics Department.

I would like to express my sincere thanks to **Mrs. Salma Jahan** for her continuous support. This project wouldn't be possible without her invaluable guidance and immense patience. Throughout working on this wonderful project, I have learned a lot from her immense knowledge and experience. She consistently allowed this project to be our work and steered us in the right direction whenever needed.

I would like to thank all of my teachers at **Noakhali Science and Technology University**. In the past four years, their teaching and guidance have helped us grow our knowledge and understanding. And I must extend my deep appreciation to all my group members, without their support and coordination we would not have been able to complete this project.

Finally, I am very thankful to my parents for their outstanding support and belief in me through my studies.

Thank you

H. M. Abrar Fahim Patwari

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

OMR	Optical Mark Recognition
IBM	International Business Machines
AI	Artificial Intelligence
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
OCR	Optical Character Recognition
ICR	Intelligent Character Recognition
NCS	National Computer System
SIFT	Scale Invariant Feature Transform
SURF	Speeded Up Robust Features
MMCA	Modify Multi-Connect Architecture
JPEG	Joint Photographic experts Group
IDE	Integrated Development Environment
OpenCV	Open Source Computer Vision Library
CUDA	Compute Unified Device Architecture
OpenCL	Open Computing Language
GUI	Graphical User Interface
STL	Standard Template Library

ABSTRACT

This method is cheaper and faster in grading an OMR sheet than an expensive OMR machine. The use of an OMR machine is not practical for personal or small-scale use. So we approached to solve the problem with the help of computer vision. This method is powered by the modern computer vision library ***OpenCV***. We found the edge of the answer sheet with ***Canny Edged Detector*** and wrapped the perspective with the ***transformation matrix***. After that, we were finally able to detect the ***threshold*** and match the answer sheet with the actual answer. Finally, we relay the grade to the OMR sheet. Our proposed method can grade the answer sheet in real-time. And a GUI is made to make the task more organized. Since the data derived from the paper is more flexible.

Keywords: MCQ, OMR Grading, Computer Vision, Image Processing, Python OpenCV, Numpy, GUI,

CHAPTER 1

INTRODUCTION

1.1 OMR and OMR sheet

1.1.1 OMR

Optical Mark Recognition (also called Optical Mark Reading or OMR) is the process of reading information that people mark on surveys, tests and other paper documents. It is convenient to apply a process that is used to read data from OMR sheets that are specialized forms with a bubble pattern on them. Once the data is retrieved from the forms they get stored in the computer system as easy to access files. The data that is to be interpreted from the OMR sheets are mostly in the form of markings such as ticks, dots, darkened circles or dashes. These marks are identified when the OMR sheet passes through the scanner of OMR sheet reader software that extracts the data and stores it into separate files. (Technologies)

1.1.2 OMR Sheet

OMR sheet is a specially created sheet of paper that can go through the OMR sheet reader software and can be used as a source to extract data. The OMR sheet comprises multiple-choice questions that are specially placed in an order that can be detected by the scanner because of its unique alignment. The numbers of columns given on a form are equal to the number of sensors available in the scanner. (Technologies)

1.1.3 History of OMR

Optical Mark Recognition (OMR) is the scanning of paper to detect the presence or absence of a mark in a predetermined position. (Bratianu & Dinca, 2010) Optical mark recognition has evolved from several other technologies. In the early 19th century and 20th century, patents were given for mechanics that would aid the blind. (Bradford, 1974)

OMR is now used as an input device for data entry. Two early forms of OMR are paper tape and punch cards which use actual holes punched into the medium instead of pencil filled circles on the medium. Paper tape was used as early 1857 as an input device for the telegraph. (McKeown, 2018) Punch cards were created in 1890 and were used as an input devices for computers. The use of punch cards declined greatly in the early 1970s with the introduction of personal computers. (Hompel et al., 2019) With modern OMR, where the presence of a pencil filled in the bubble is recognized, the recognition is done via an optical scanner.

The first mark sense scanner was the IBM 805 Test Scoring Machine; this read marks by sensing the electrical conductivity of graphite pencil lead using pairs of wire brushes that scanned the pages. In the 1930s, Richard Warren at IBM experimented with optical mark sense system for test scoring as documented in US patents 2,150 ,256(filled in 1932, granted in1939) and 2,010,653(filled in 1933, granted in 1935). The first successful optical mark-sense scanner was developed by Everett Franklin Lindquist as documented in US Patent 3,050,248(filled in 1955, granted in 1962). Lindquist had developed numerous standardized educational tests and needed a better test scoring machine than the then standard IBM 805. The rights to Lindquist's patents were held by the measurement Research Centre until 1968, when the University of Lowa sold the operation to Westinghouse corporation.

During the same period, IBM also developed a successful optical mark-sense test scoring machine, as documented in US patent 2,944,734(filled in 1957, granted in 1960). IBM commercialized this as the IBM 1230 optical mark scoring reader in 1962. This is a variety of related machines that allowed IBM to migrate a wide variety of applications developed for its mark sense machines to the new optical technology. These applications included a variety of inventory management and trouble reporting forms, most of which had the dimensions of a standard card.

While the other players in the educational testing areas focused on selling scanning Services, Scantron corporation, founded in 1972 (Marr, 2019), had a different model; it would distribute inexpensive scanners to schools and make a profit from selling the test forms. As a result, many people came to think of all mark-sense forms (whether optically sensed or not) of Scantron forms.

In 1983, Westinghouse Learning Corporation was acquired by National Computer System (NCS). In 2000, NCS was acquired by Pearson Education, where the OMR technology formed the core of Pearson's Data management group. In February 2008, M&F worldwide purchased the Data Management group from Pearson; the group is now part of the Scantron brand. (Demush, 2019)

1.1.4 How OMR works

Optical Mark Reader is an OMR scanner that reads the mark of pencils made on predefined positions on the OMR sheets. The best OMR software available in the market can be paired with any normal scanner and have inbuilt sensors that are efficient in detecting dark pencil marked areas present on the OMR form. There are approx. 48 sensors that work together to make the scanning of OMR sheets possible. Once the scanning is complete the information is converted to a computer data file. (Technologies)

1.1.5 Application of OMR

Consumer Surveys: Companies often conduct various types of surveys regarding their products or services. Since large volume of data is collected in this process, OMR technology is used to make it easy and fast.

Test sheets: In recent times many competitive as well as government exams have started to integrate OMR technology. They have recognized its vast functionalities and are using its various features to their benefit.

Election Poll Results: In elections counting the number of votes separately is a tedious task. By using some of the best OMR software available in the market the process becomes relatively fast and accurate as well as curtails the chances of cheating and rigging the vote counting.

Keeping the count of Inventory : In big warehouses, industries or factories, keeping the count of inventories is a mandatory aspect and with the help of Optical Mark Recognition it becomes very easy.

Lotteries: In Lotteries it is used to count the lucky lottery numbers. (Maniar et al., 2021)

1.2 Computer Vision

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

Computer vision works much the same as human vision, except humans have a head start. The human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image.

Computer vision trains machines to perform these functions, but it has to do it in much less time with cameras, data and algorithms rather than retinas, optic nerves and visual cortex. Because a system trained to inspect products or watch a production asset can analyze thousands of products or processes a minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities.

Computer vision is used in industries ranging from energy and utilities to manufacturing and automotive – and the market is continuing to grow. It is expected to reach USD 48.6 billion by 2022. (Marr, 2019)

1.2.1 The way Computer Vision Works

Computer vision needs lots of data. It runs analyses of data over and over until it discerns distinctions and ultimately recognizes images. For example, to train a computer to recognize automobile tires, it needs to be fed vast quantities of tire images and tire-related items to learn the differences and recognize a tire, especially one with no defects.

Two essential technologies are used to accomplish this: a type of machine learning called deep learning and a convolutional neural network (CNN).

Machine learning uses algorithmic models that enable a computer to teach itself about the context of visual data. If enough data is fed through the model, the computer will “look” at the data and teach itself to tell one image from another. Algorithms enable the machine to learn by itself, rather than someone programming it to recognize an image.

A CNN helps a machine learning or deep learning model “look” by breaking images down into pixels that are given tags or labels. It uses the labels to perform convolutions (a mathematical operation on two functions to produce a third function) and makes predictions about what it is “seeing.” The neural network runs convolutions and checks the accuracy of its predictions in a series of iterations until the predictions start to come true. It is then recognizing or seeing images in a way similar to humans.

Much like a human making out an image at a distance, a CNN first discerns hard edges and simple shapes, then fills in information as it runs iterations of its predictions. A CNN is used to understand single images. A recurrent neural network (RNN) is used in a similar way for video applications to help computers understand how pictures in a series of frames are related to one another.

1.2.2 The History of Computer Vision

Scientists and engineers have been trying to develop ways for machines to see and understand visual data for about 60 years. Experimentation began in 1959 when neurophysiologists showed a cat an array of images, attempting to correlate a response in its brain. They discovered that it responded first to hard edges or lines, and scientifically, this meant that image processing starts with simple shapes like straight edges. (Demush, 2019)

At about the same time, the first computer image scanning technology was developed, enabling computers to digitize and acquire images. Another milestone was reached in 1963 when computers were able to transform two-dimensional images into three-dimensional forms. In the 1960s, AI emerged as an academic field of study, and it also marked the beginning of the AI quest to solve the human vision problem.

1974 saw the introduction of optical character recognition (OCR) technology, which could recognize text printed in any font or typeface. (Wikipedia) Similarly, intelligent character recognition (ICR) could decipher hand-written text using neural networks. (Wikipedia) Since then, OCR and ICR have found their way into document and invoice processing, vehicle plate recognition, mobile payments, machine translation and other common applications.

In 1982, neuroscientist David Marr established that vision works hierarchically and introduced algorithms for machines to detect edges, corners, curves and similar basic shapes. Concurrently, computer scientist Kunihiko Fukushima developed a network of cells that could recognize patterns. The network, called the Neocognitron, included convolutional layers in a neural network.

By 2000, the focus of the study was on object recognition, and by 2001, the first real-time face recognition applications appeared. Standardization of how visual data sets are tagged and annotated emerged through the 2000s. In 2010, the ImageNet data set became available. It contained millions of tagged images across a thousand object classes and provides a foundation for CNNs and deep learning models used today. In 2012, a team from the University of Toronto entered a CNN into an image recognition contest. The model, called AlexNet, significantly reduced the error rate for image recognition. After this breakthrough, error rates have fallen to just a few percent. (Demush, 2019)

1.2.3 Computer Vision Applications

There is a lot of research being done in the computer vision field, but it’s not just research. Real-world applications demonstrate how important computer vision is to endeavours in business, entertainment, transportation, healthcare and everyday life. A key driver for the growth of these applications is the flood of visual information flowing from

smartphones, security systems, traffic cameras and other visually instrumented devices. This data could play a major role in operations across industries, but today goes unused. The information creates a testbed to train computer vision applications and a launchpad for them to become part of a range of human activities:

1. IBM used computer vision to create My Moments for the 2018 Masters golf tournament. IBM Watson watched hundreds of hours of Masters footage and could identify the sights (and sounds) of significant shots. It curated these key moments and delivered them to fans as personalized highlight reels.

2. Google Translate lets users point a smartphone camera at a sign in another language and almost immediately obtain a translation of the sign in their preferred language.[11]

3. The development of self-driving vehicles relies on computer vision to make sense of the visual input from a car's cameras and other sensors. It's essential to identify other cars, traffic signs, lane markers, pedestrians, bicycles and all of the other visual information encountered on the road.

4. IBM is applying computer vision technology with partners like Verizon to bring AI to the edge, and to help automotive manufacturers identify quality defects before a vehicle leaves the factory

1.3 Image Processing

1.3.1 Introduction

By definition, image processing is a method to convert an image into digital form and perform some operations on it, to get an enhanced image or to extract some useful

information from it. it is a type of signal dispensation in which the input is an image and the output is an image or characteristics associated with that image. (Ramamoorthy et al., 2018)

1.3.2 Image Processing Operations and Algorithms

Image processing involves several techniques and algorithms. The most representative image processing operations are: Changing Colorspaces, Geometric Transformations of Images, Image Thresholding, Smoothing Images, Morphological Transformations, Image Gradients, Canny Edge Detection, Image Pyramids, Image Transforms in OpenCV.

The image processing algorithms are commonly used for complete image capture can be categorized into **a)** Low-level techniques, such as colour enhancement and noise removal, **b)** Medium-level techniques, such as compression and binarization, and **c)** higher-level techniques involving segmentation, detection, and recognition algorithms extract semantic information from the captured data.

1.3.3 Types of Image Processing Algorithms

Some of the conventional image processing algorithms are as follows:

Contrast Enhancement algorithm: The colour enhancement algorithm is further subdivided into.

Histogram equalization algorithm: Using the histogram to improve image contrast

Adaptive histogram equalization algorithm: It is the histogram equalization that adapts to local changes in contrast

Connected-component labelling algorithm: It is about finding and labelling disjoint regions

Dithering and half-toning algorithm: Dithering and half-toning includes: Error diffusion algorithm, Floyd–Steinberg dithering algorithm, Ordered dithering algorithm, Riemersma dithering algorithm

Elser difference-map algorithm: It is a search algorithm used for general constraint satisfaction problems. It was used initially for X-Ray diffraction microscopy.

Feature detection algorithm: Feature detection consists of *Marr–Hildreth algorithm* which is an early edge detection algorithm and the *Canny edge detector algorithm*. A canny edge detector is used for detecting a wide range of edges in images. Such as, Generalized Hough transform algorithm, Hough transforms algorithm, SIFT (Scale-invariant feature transform) algorithm: SIFT is an algorithm to identify and define local features in images. SURF (Speeded Up Robust Features) algorithm, SURF is a robust local feature detector.

Seam carving algorithm: Seam carving algorithm is a content-aware image resizing algorithm

Segmentation algorithm: This particular algorithm parts a digital image into two or more regions. GrowCut algorithm is an interactive segmentation algorithm

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

A different methods of OMR grading has been used. And there was some attempts to solve this task with the help of image processing as well. We shall look into some of these algorithms from a different researchers. Also, we shall discuss the scope of their method in this chapter.

2.2 Research work on related methods

Yasira Fathima, Prarthana Bhat, Vikshitha Amin , Mrs.Sucheta G proposed a paper that uses the image processing technique, they develop a system based on a PC-type microcomputer which is connected to an image scanner (Gowda & Amin, 2018). Initially, the true answer keys are stored in the database. The scanned sheets are sent to the system which is pre-processed using noise reduction and skew detection techniques, which is then compared with the true answer keys for the generation of results. The results and analysis proved that the system was 99.96% accurate.

Zeki KÜÇÜKKARA, Abdullah Erdal TÜMER proposed paper and they uses Python software language with OpenCV, Imutils and ZBar libraries (Tümer & Küçükkara, 2018). When the answer sheet is loaded in the application, incorrect answers are marked as red while the correct ones as green and with the calculation of the correct/incorrect answers and blanks, the result is printed on the optical form image The recognition success percentage of the system was found as 99.76.

Russul Hussein, Hasan, Emad Abdul Kareem proposed a paper on Optical Mark Reader Based on Modify Multi-Connect Architecture MMCA using image processing (Hasan & Abdul Kareem, 2021). This paper proposes a system that works on two phases which is the training phase and the identification phase. This method was able to recognize multiple options.The system is 99.96 % accurate and is gaining importance in the field of education. Here the extraction of data is performed based on two axes and the in the checking of answers the number of black pixels is evaluated and results are generated.

Krisana Chinnasarn Yuttapong Rangsanseri and Punya Thitimajshima proposed a paper on text/Graphics Images removal of salt and pepper noise (Chinnasarn et al., 1998). This paper proposed a system that will perform a comparison between the real answer sheet and scanned sheet of the candidate. If the scanned sheet matches the original sheet further processing takes place or else not. For the answer evaluating process the number of black pixels will be found and results will be sent to the candidate in the form of SMS.

Sumitra. B. Gaikwad proposed a paper on OMR Sheet Scanning using image processing (Gaikwad, 2015). This paper proposes a method in which the answer sheet of the candidate will be sent for scanning and the scanned image will be taken to be the in-put. Using image processing the total number of a correct answer, wrong answers and not answered questions will be separately displayed. Totaling of scores is also done. This paper avoids the use of costly hardware and saves time.

Astha Gupta and Sandhya Avasthi proposed a paper on a low cost method to the OMR process for surveys and research using image processing (Gupta & Avasthi, 2016).This paper analyzes the new technique which overcomes the limitations of the OMR process. It scans the hard copies of paper that are designed and marked in a specific

templates and saves the scanned copy in JPEG format. This includes 4 basic steps namely template designing, image capturing and performing transformation and scaling. The solution has been designed making use ofopen-source tools and technology to keep its platform-independent.

Bhoj Bahadur Karki and Nirman Giri proposed a paper and *this paper tries to* compare different mechanisms with each other and will try to implement one of the best techniques among them to verify the result (Karki, 2018).In this paper, a comparison of different algorithm is made in order to evaluate their performance. After evaluation, this paper focuses on building an application using one of the algorithm which will be more efficient. Normally, accuracy will be our major concerned while choosing algorithm rather than performance. They have proposed to use windowing technique as it gives the most accurate result. According to the test the result obtained by the program was nearly 100%.

CHAPTER 3

METHODOLOGY

3.1 Introduction

We used python programming language and some of its libraries to create the GUI and project. Three algorithms was most dominant in this task. Image canny algorithm (Canny, 1986), Contour Finding (Nitzberg et al., 1993), and Perspective transform (Zhang, 2014) in Open CV. We briefly discussed these in this chapter.

3.2 Canny Edge Detector

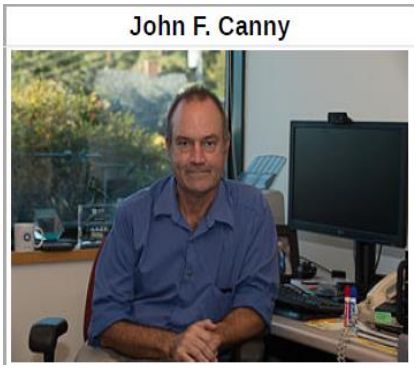

Figure 1: Jhon F. Canny

An edge detection algorithm that is used to detect a wide range of edges in a given image is the Canny Edge Detector. The Canny Edge Detector is the most widely used in computer vision. Like another traditional edge detector, it is not just masking or hovering on the input image matrix. Instead, it is detailed and has the step to follow. It was developed by John F. Canny in 1986 who is an Australian computer scientist. He published a book named "A computational approach to edge detection" under IEEE transaction on pattern analysis and machine intelligence to explain why the technique works.

In python programming language we use the Canny(image, threshold1, threshold2, edges=..., apertureSize=..., L2gradient=...) function to find edges in the input image and marks them in the output map edges using the Canny algorithm. The smallest value between threshold1 and threshold2 is used for edge linking. The largest value is used to find initial segments of strong edges.

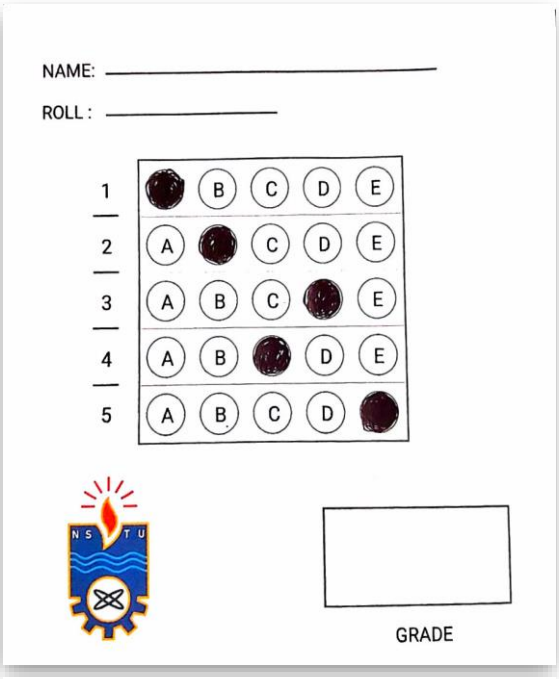

Figure 2: Original Image

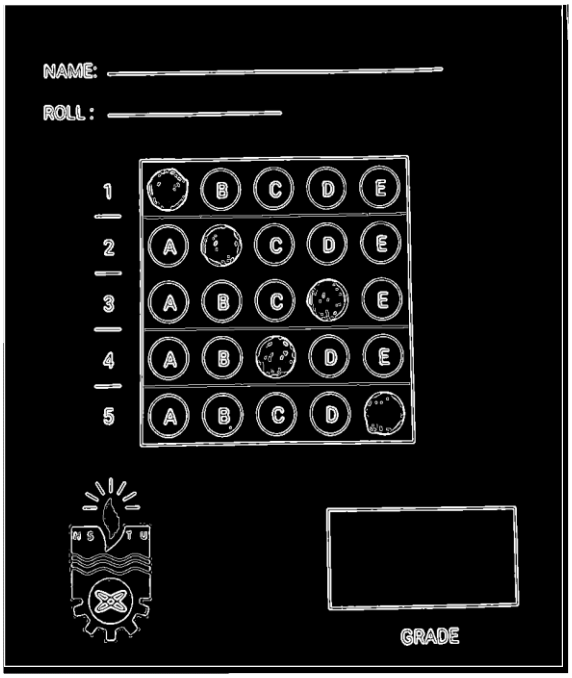

Figure 3: Canny

3.2.1 Edge Detection Criteria

The general edge detection criteria include:

1. The detection criterion expresses the fact that important edges should not be missed and there should be no spurious responses.
2. The localization criterion says that the distance between the actual and located position of the edge should be minimal.
3. The one response criterion minimizes multiple responses to a single edge. When there are two responses to a single edge, one of them should be considered false. The third criterion solves the problem of an edge corrupted by noise and works against non-smooth edge operators.

3.2.2 Process

The five different steps of the Canny Edge Detection algorithm are:

1. Apply the Grayscale effect to convert the RGB image to black and white or grey monochrome image to reduce the image data to simplify the next computation.
2. Apply a Gaussian filter to filter out the noise in the input image.
3. Find the intensity gradient of the image.
4. Apply non-maximal suppression to get rid of spurious response to edge detection.
5. Apply double threshold to determine potential edges.

6. Track edge by hysteresis. Finalize the strong edges by suppressing all other edges that are weak and not connected to the strong edges.

3.2.3 Conversion to the Grayscale

We have to convert the RGB given image to a gray scale image to reduce the 3D image matrix to a 2D image matrix to simplify the next step computation. The formula for converting the RGB image to a gray scale image is

Y = 0.299R + 0.587G + 0.114B.

Here Y is denoted as a nonlinear luma component and R, G, B are denoted red, green, blue colour respectively. In python programming language we use cvtColor(src, code, dst=..., dstCn=...) function to convert the RGB image to a gray scale image.

3.2.4 Gaussian Filter

Since the noise in the image affects all the edge detection result, we have to remove the noise to prevent false detection caused by it. Gaussian filter is an operator which is used to remove the noise in the input image. This noise removed image shall enable further processing to be smooth and flawless. The equation for a Gaussian filter kernel of size (2k+1)×(2k+1) is given by:

$$H_{ij} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{\left(i-(k+1)\right)^2+\left(j-(k+1)\right)^2}{2\sigma^2}\right); 1 \le i,j \le (2k+1)$$

Here we use a 5×5 Gaussian Filter to create the adjacent image, with σ = 1. (The asterisk denote convolution operation.)

$$\boldsymbol{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \boldsymbol{A}$$

It is necessary to know that the size of the Gaussian kernel will affect the performance of the detector. When the size of the Gaussian kernel increases, the detector sensitivity to noise decreases. But at the same time, the localization error to detect the edge will slightly increase with the increase of the Gaussian filter kernel size. A 5×5 is a good size for most cases, but this will also vary depending on specific situations.

Again, if we increase σ the number of detected edges reduce. In python programming language we use GaussianBlur(image,ksize, sigma) function to apply the Gaussian filter on the Grayscale image.

3.2.5 Intensity Gradient Calculation

Each edge in an image may point in a variety of directions. So, the Canny algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image. There are three types of edge detection operators such as Roberts, Prewitt or Sobel to find the intensity of Gradient. But in most of the case,s we use the Sobel operator as an edge detection operator. If we define A as the 2D Gaussian image. Then, the Sobel operator returns a value for the first derivative in the horizontal direction (G_x) and the vertical direction (G_y). Then, the resulting gradient approximation can be calculated with

$$G = \sqrt{G_x^2 + G_y^2}$$

And the formula for finding the edge direction is

$$\theta = tan^{-1}\left(\frac{G_y}{G_x}\right)$$

The G will be compared against the threshold and with which, one can understand the taken point is an edge or not. The edge detection angle is rounded to one of four angles representing vertical, horizontal and two diagonals (0˚, 45˚, 90˚ and 135˚). An edge direction falling in each colour region will be set to a specific angle value, for instance, θ in [0˚,22.5˚] or [157.5˚, 180˚] maps to 0˚.

3.2.6 Non-Maximal Suppression

Non-Maximal Suppression or minimum cut off suppression of gradient magnitudes is an edge thinning technique. It can be shown that convolving an image with asymmetric 2D Gaussian and then differentiating in the direction of the gradient (perpendicular to the edge direction) forms a simple and effective directional operator.

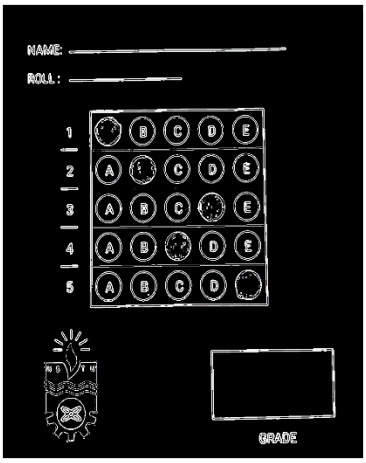

Figure 4: Non Maximal Suppression

Suppose G is a 2D Gaussian and assume we wish to convolve the image with an operator G_n which is the first derivative of G in some direction **n**.

$$G_n = \frac{\partial G}{\partial \boldsymbol{n}} = \boldsymbol{n}\nabla G \quad \ldots\ldots\ldots\ldots (1)$$

We would like to n to be perpendicular to the edge: this direction is not known in advance, but a robust estimate of it based on the smoothed gradient direction is available. If f is the image, the normal to the edge n is estimated as

$$\boldsymbol{n} = \frac{\nabla(G * f)}{|\nabla(G * f|} \quad \ldots\ldots\ldots\ldots\ldots \quad (2)$$

The edge location is then at the local maximum of the image f convolved with the operator G_n in the direction **n**

$$\frac{\partial}{\partial \boldsymbol{n}} G_n * f = 0 \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

This equation (3) illustrates how to find the local maximum in the direction perpendicular to the edge; this operation is often referred to as non-maximal suppression.

3.2.7 Double Threshold

As we see from the previous result, Non-maximum suppression has not provided us with excellent results. There is still some noise. The image even raises a threat in mind that some of the edges shown may not be so and some edges could be missed in the process. To find a flawless result, it is essential to filter out the edge pixels with a weak gradient value and preserve the edge pixels with a high gradient value. For this, we need to go with double thresholding and in this process, we have to set two thresholds. One, a high and another a low.

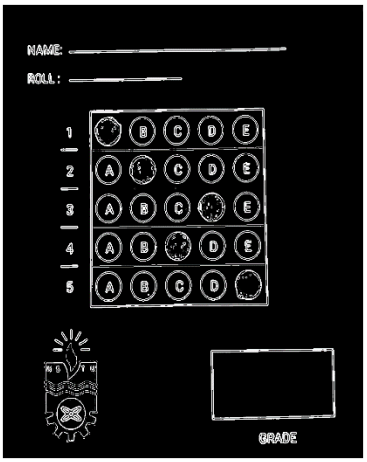

Figure 5: Double Threshold

Simply assume a high threshold value as 0.35. Any pixel with a value above o.35 is to be seen as a stronger edge. Another threshold, the lower one can be 0.2. In that case, any pixel below this value is not an edge at all and hence set them all to 0. But if an edge pixel's gradient value is smaller than the low threshold value i.e. 0.2, it will be suppressed. The two threshold values are empirically determined and their definition will depend on the content of a given input image.

3.2.8 Walkthrough of the algorithm

This section will show the progression of an image through each of the five steps.

Figure 6: Walkthrough of Canny

3.3 Contour Detection

3.3.1 Introduction

Contour can be simply defined as a closed curve that joins all the continuous points along the boundary having same color and intensity. Contour detection is a mostly

used computer vision process for shape analysis, object detection and recognition of an image.

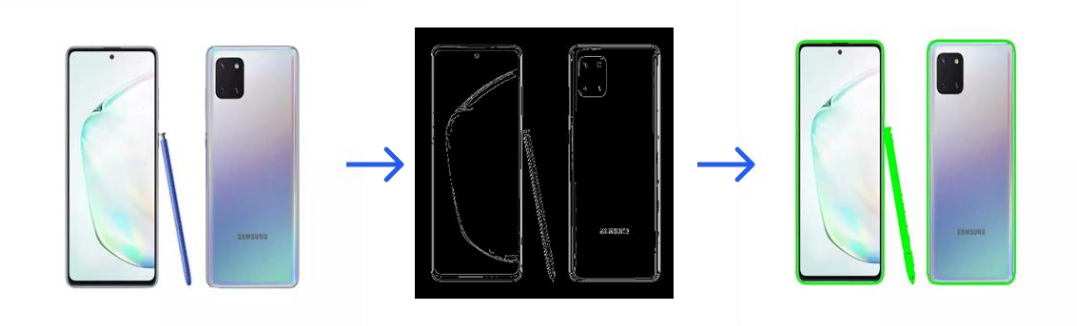

Figure 7: Contour Detection

The same colour or intensity that covers an area helps to identify the object that we want and the curve that encloses this area is the contour that represents the shape of the object. So, it is assumed to us that the contour detection process is as like an edge detection process. But in the case of contour detection, edges detected must form a closed path.

In OpenCV, we find contours of the white objects from a black background. So, in that case, the object that we want to detect should be white and the background should be black.

In the python programming language, we use cv2.findContours() function to detect the object that we want and also use cv2.drawContours() to draw contour outlines or filled contours.

3.3.2 Detecting Contours in an Image

For detecting contours in an image, OpenCV provides a handy function named find contours() that retrieves contours of each object in a binary image using the algorithm @cite Suzuki85 and also return information about the image topology.

3.3.3 Function syntax:

findContours(image, mode, method, contours=..., hierarchy=..., offset=...)

Input Arguments	image	Source, an 8-bit single-channel image. Non-zero pixels are treated as 1's. Zero pixels remain 0's, so the image is treated as binary. You can use #compare, #inRange, #threshold, #adaptiveThreshold, #Canny, and others to create a binary image out of a grayscale or colour one. If mode equals #RETR_CCOMP or #RETR_FLOODFILL, the input can also be a 32-bit integer image of labels (CV_32SC1)
	Mode	Contour retrieval modes such as RETR_EXTERNAL, RETR_LIST, RETR_CCOMP, RETR_TREE and RETR_FLOODFILL
	method	Contour approximation method
	offset	Optional offset by which every contour point is shifted. This is useful if the contours are extracted from the image ROI and then they should be analyzed in the whole image context
Output Arguments	contours	Detected contours. Each contour is stored as a vector of points.
	hierarchy	Optional output vector that provides information about the image topology. It has as many elements as the number of contours.

Table 1 Contour Parameters

3.3.4 Object location and their relationship in a binary image

For object recognition and shape analysis OpenCV provides find contours() function that can detect the contours of each object and can retrieve information about how the contours are linked to each other in a binary image. In an image, object may have a different locations. Some objects may be nested in other objects. To define the objects' location and their relationship in an image we use RetrievalModes as input arguments and

get information as an array Hierarchy which is an output argument of findContours() function.

3.3.5 Hierarchy

As we know some objects may be nested in other objects, we call outer one as parent and inner one a child. This way, we define location and relationship among each contour in a binary image and we can assume the connectivity among contours by showing whether a contour is a child of some other contour or it is a parent. Hence, the way we represent the relationship among contours in an image is called Hierarchy.

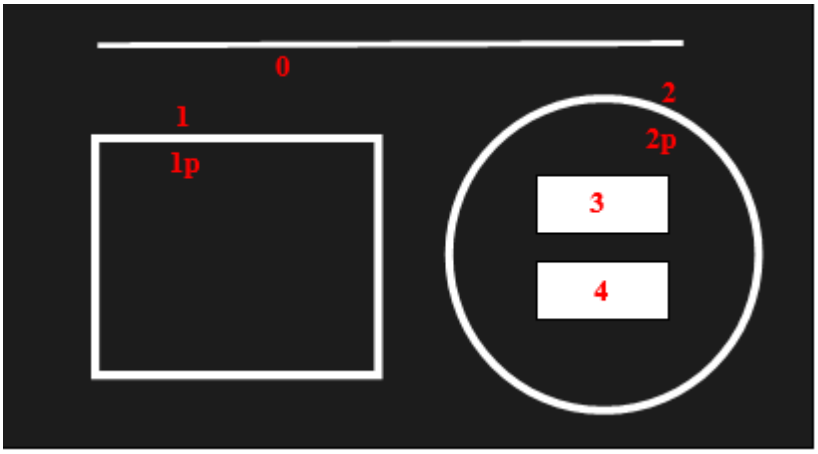

Figure 8: Hierarchy

In this image, there are shapes which are numbered from 0-5.

3.3.6 Contour Approximation Modes

CHAIN_APPROX_NONE	retrieves all the contour points.
CHAIN_APPROX_SIMPLE	compresses horizontal, vertical, and diagonal segments and leaves only their endpoints. For example, it retrieves four points for rectangular contour.

CHAIN_APPROX_TC89_L1	applies one of the flavors of the Teh-Chin chain approximation algorithm
CHAIN_APPROX_TC89_KCOS	applies one of the flavors of the Teh-Chin chain approximation algorithm

Table 2 Contour Approximation Modes

3.4 Perspective Transform

Perspective transformation (Zhang, 2014) is a tool of OpenCV that is used to get a birds-eye view of a specific region of a given image for better insights. For example, we can consider an image of paper on a table. To select only the paper and to get a birds-eye view of that paper we need to change the perspective of that image by perspective transformation. (Khatri et al., 2022)

For perspective transformation, OpenCV provides two functions cv2.getPerspectiveTransform and cv2.warpPerspective. In cv2.getPerspectiveTransform function, we have to pass the corner points of a specific region of a given image and also pass the points inside which we want to display our specific region. After passing two sets of points, it will return a transformation matrix. Using this transformation matrix as an input argument, cv2.warpPerspective will warp the image.

Syntax: cv2.getPerspectiveTransform(src, dst) and cv2.warpPerspective(src, M, dsize)

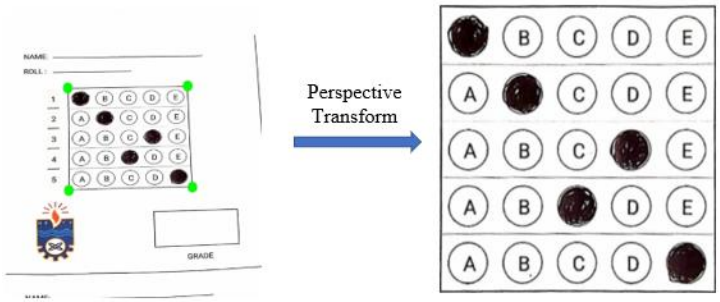

Figure 9: Perspective Transform

Here the four corner points of the biggest rectangle of the original image are [[169, 189], [497, 184], [172, 738], [510, 430]] And the four corner points of our required region inside which we want to display are [[0, 700], [700, 0], [0, 700], [700, 700]]

Based on the above two sets of points cv2.getPerspectiveTransform() will return a transformation matrix which is

$$\begin{bmatrix} 2.14912e+00 & -2.58931e-02 & -3.58308e+02 \\ 4.49897e-02 & 2.95133e+00 & -5.65404e+02 \\ -3.25149e-05 & 1.26973e-04 & 1.00000e+00 \end{bmatrix}$$

We multiply the location coordinate of each source pixel with a transformation matrix to compute the destination location coordinate of each pixel. There are two steps to obtain the destination location coordinate of each pixel.

Step-1: At the first step, we multiply each pixel x, y coordinate values with the transformation matrix. Here we consider the pixel at [497, 184] point.

$$\begin{bmatrix} 2.14912e+00 & -2.58931e-02 & -3.58308e+02 \\ 4.49897e-02 & 2.95133e+00 & -5.65404e+02 \\ -3.25149e-05 & 1.26973e-04 & 1.00000e+00 \end{bmatrix} \begin{bmatrix} 497 \\ 184 \\ 1 \end{bmatrix} = \begin{bmatrix} 705.0422 \\ 0.00 \\ 1.0072 \end{bmatrix}$$

Step-2: We divide the first two rows by third row to obtain (x, y) as given like this [705.0422/1.0072, 0.00/1.0072] = [700, 0]. The source to destination pixel location would be like this. Source[497, 184] = Destination[700, 0]

These two steps are applied to all source pixels. The function that does the work is cv2.warpPerspective.

CHAPTER 4

MAIN WORK

4.1 Software and Tools

4.1.1 Pycharm

PyCharm is one of the most famous Integrated Development Environment (IDE) for Python, developed by a Czech organization called JetBrains. The integrated environment comes with the platform for Code Analysis, Graphical Debugger, Unit Tester and Version Control Systems, etc. It also supports web development with the Django framework.

4.1.2 Python

Python (Labs, 2022) is an open-source (free) programming language that is used in web programming, data science, artificial intelligence, and many scientific applications. Learning Python allows the programmer to focus on solving problems, rather than focusing on syntax. Its relative size and simplified syntax give it an edge over languages like Java and C++, yet the abundance of libraries gives it the power needed to accomplish great things.

4.1.3 Python Libraries

A Python library is a reusable chunk of code that you may want to include in your programs/ projects. Compared to languages like C++ or C, Python libraries do not pertain to any specific context in Python. Here, a ‘library’ loosely describes a collection of core modules. Essentially, then, a library is a collection of modules. A package is a library that can be installed using a package manager like ruby gems or npm

4.1.4 NumPy

NumPy (Karimi, 2021) is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms basic linear algebra, basic statistical operations, random simulation and much more.

4.1.5 OpenCV

OpenCV (Jain, 2021) (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high-resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people in the user communities and an estimated number of downloads exceeding 18 million [3]. The library is used extensively in companies, research groups and by governmental bodies.

4.2 Algorithm

The work is followed by the following steps of work. We shall discuss them shortly.

Step 1:
Converting an image into its grey form

Step 2:
Applying Gaussian Blur in that image

Step 3:
Image Canny Process. By using the Canny Algorithm. The strongest edge is found from this step (Canny, 1986). We covered this in the methodology chapter.

Step 4:
Finding Contours (Lines that surround and define the edges of a subject, giving it shape and volume.) In this step non-zero pixel is treated as 1. Whilst the Zero pixel remains as 0. In that case, our source is regarded as binary.

It finds the contours in a binary image. (Sampaio & Kroger, 2016) It is a useful tool for shape analysis and object detection and object recognition. Here, all contour is stored as a vector of points. (Nitzberg et al., 1993)

Step 5:
Drawing the contours. In this step, we take the image form and apply the contour vector we get from the previous step. And draw the contours.

Step 6:
Find the rectangular Contours. This is crucial because we shall look for the answer in a rectangle. This step is likely to give us a lot of rectangles. But not all of them is necessary for our objective.

Step 7:
Finding Biggest Rectangle and marking its corner point. The rectangle contains the choices marked. And mark its corner points for further manipulation

Step 8:
Find the Second Biggest rectangle and also mark its corner points. This will be the rectangle where we shall relay the output of our data to.

Step 9:
Reorder the points so we can wrap it next.

Step 10:
Take the corner point of the two biggest contours found in steps 7 and 8. And then reordering them in step 9.

Step 11:
Create a destination image. Where we shall have the image wrapped. It is measured from the actual size of the OMR ratio

Step 12:

Get perspective transform Matrix. This is taken from the corner points taken from step 10 and the destination created in step 11. So basically, this calculation of a perspective transform is carried out from four pairs of the corresponding points.

This step returns us a matrix.

Step 13:

Wrapping the perspective from the source using the Matrix we got from step 12. This step is going to apply a perspective transformation to the image. It uses the specified matrix.

Step 14:

Turn it to grey for further manipulation.

Step 15:

Finding the threshold. This step applies fixed-level thresholding to a multiple-channel array. And this noise cancelling method is carried out to filter out pixels with too small or too large values.

This returns a binary image to our greyscale image taken from step 14.

Step 16:

Slicing the boxes from Number of Question and Choices we predetermined.

Step 17:

Count non-zero-pixel value from the Sliced Boxes we get from step 16

Step 18:

Find the Marked Answer from the rows. For simplicity, we select the box that has the maximum number of pixel values.

If a minimum of 30% of the pixel value is not received. Then the question is treated as not answered.

Step 19:

Comparing the answer to see if the marked option matches with the answer. If they match then the answer is right.

Step 20:

Show actual answer Inverted on the Sheet with inverted Transformation Matrix. We got the transformation matrix from step 12. This step requires the inverse of that matrix.

Step 21:

Save the result into a usable format such as .csv format.

Step 22:

Go through step 1 to step 21 for n number of papers.

Step 23:

Analyze the data from the final CSV output.

By Applying the algorithmic procedure in step 23 we get flexible data. Since the data is in CSV format, we can use this data to generate a graph and see how the examinee did on the exam. And generate-graph to back the initial assumption.

4.3 Flow Chart

The flow chart will contain the complete algorithmic steps in its body. It’s a simple representation of what happens under the hood of the proposed method. We get a brief idea about the total method from the algorithm. But the flow chart contain this information gives a more robust and on point understanding of the method.

It's basically a visual representation. We initialize and end the flow chart with ellipse. And the input and output are represented by the rectangle. And decision-making point is denoted with a diamond-shaped rectangle.

The flow chart of the proposed method is bellowed.

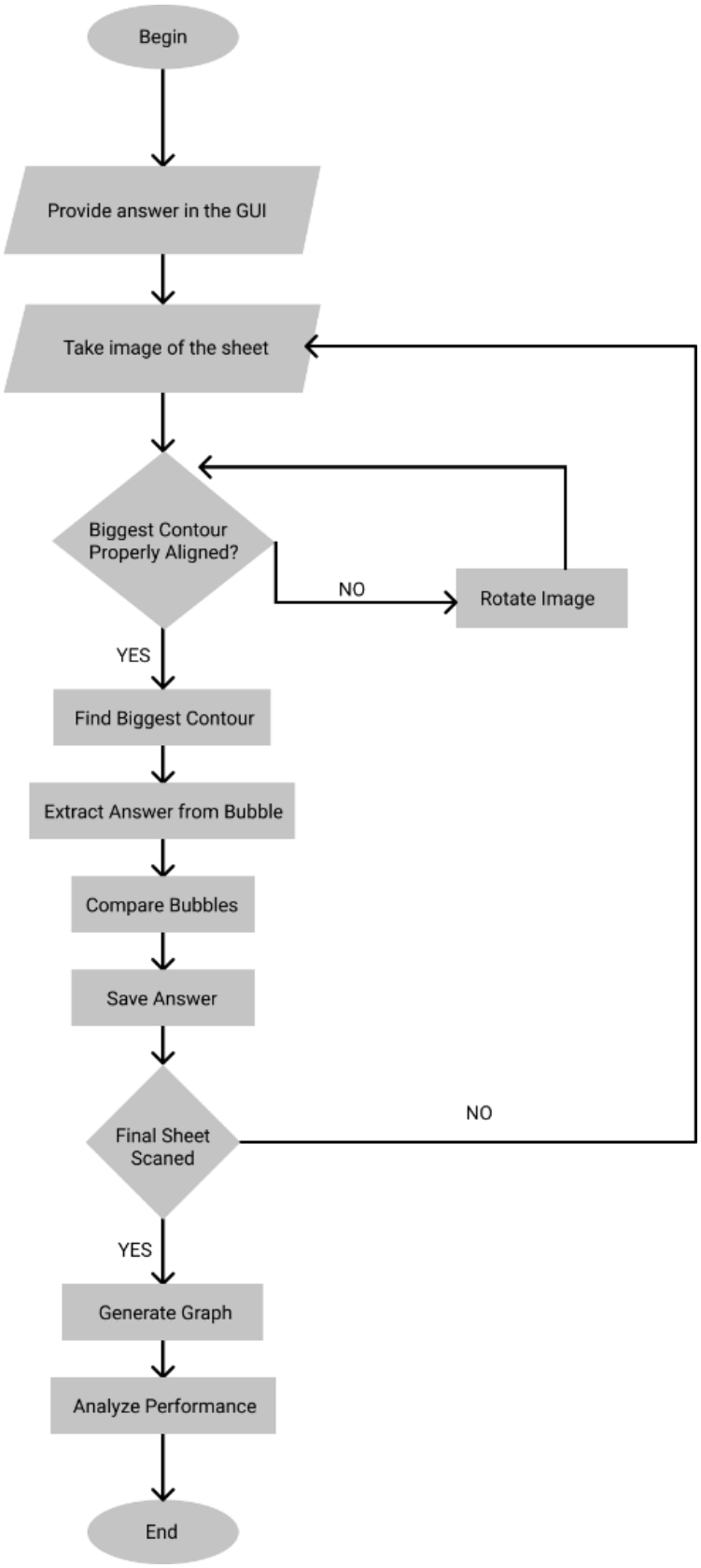

Figure 10: Flow Chart

4.4 Pipeline

First we declare the answer in the GUI and continue to grade the OMR sheet

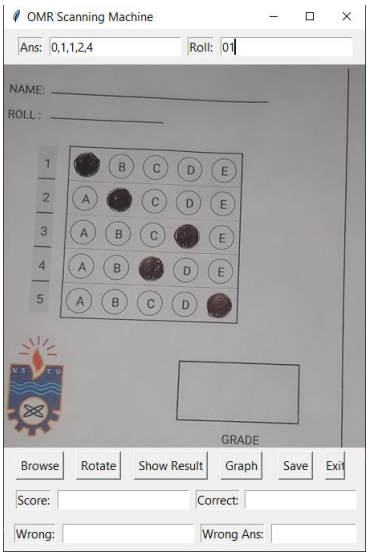

Figure 11: GUI

The total project pipeline will go like this. From left to right and below.

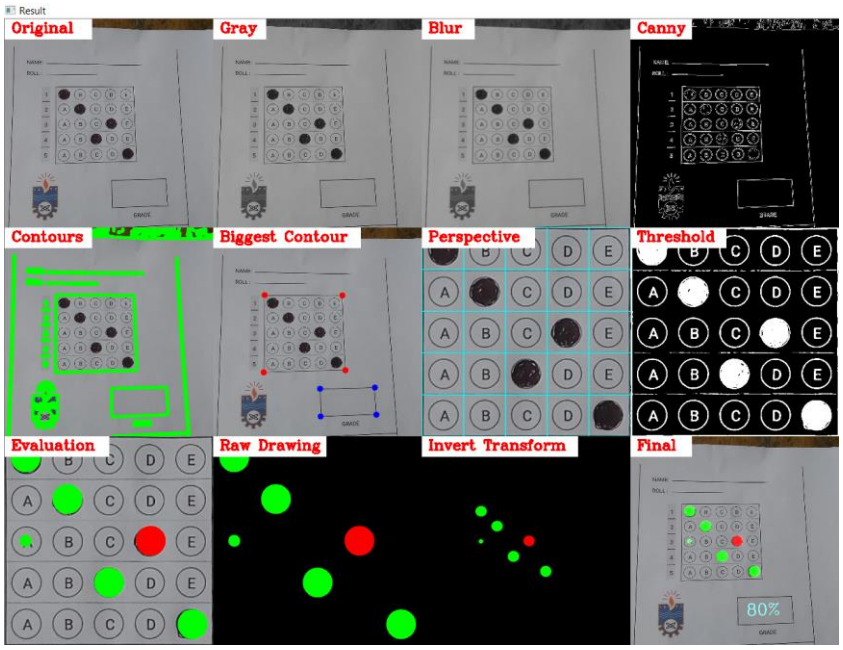

Figure 12: Pipeline

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Introduction

We have scanned 100 papers and got 100% accurate data from the paper. Though this is to be noted that our result greatly depends on the quality of the image we took. We used a decent 8-megapixel mobile camera. And the image was taken under bright light. The impact of low light was not tested in the method. Our proposed method was impacted by a shaky environment. 4 pictures were needed to be recaptured. Due to its blurred outcome. So, we used a phone holder to tackle this problem.

5.2 Result

5.2.1 Step by Step Result

The raw image was like this

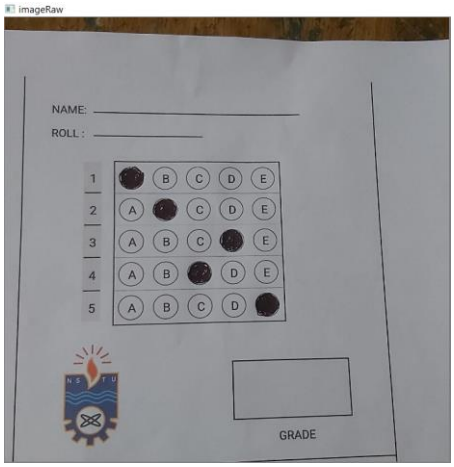

Figure 13: Raw Image

After making the **image grey** and then Applying **Image Blur** we got the following

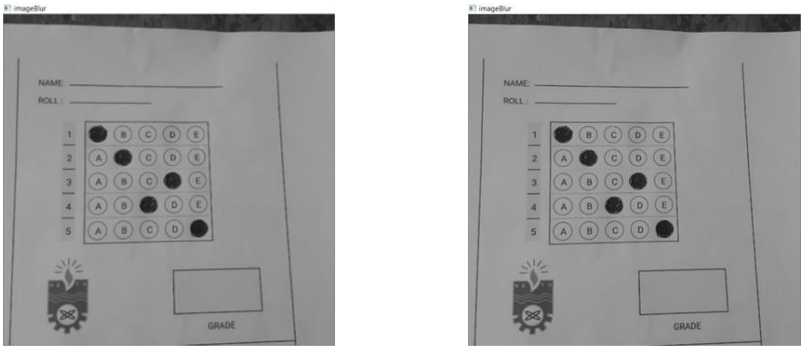

Image Grey Image Blur

Figure 14: Image Grey and Image Blue

The result from the canny edge detector is following

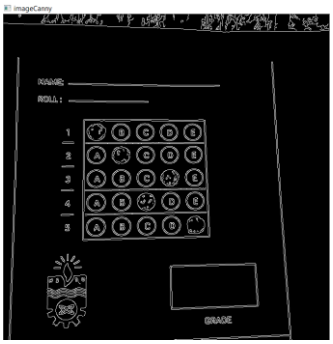

Figure 15: Image Canny

From there we got the contours and identified the two biggest contours.

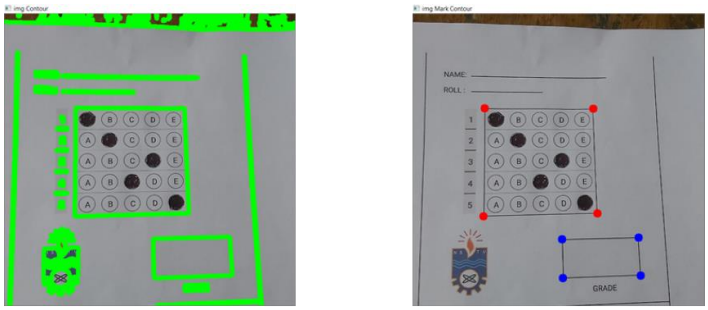

Image Contour Biggest Contour

Figure 16: Image contour and Biggest Contours

When we used threshold value = 130, it gave significantly good output.

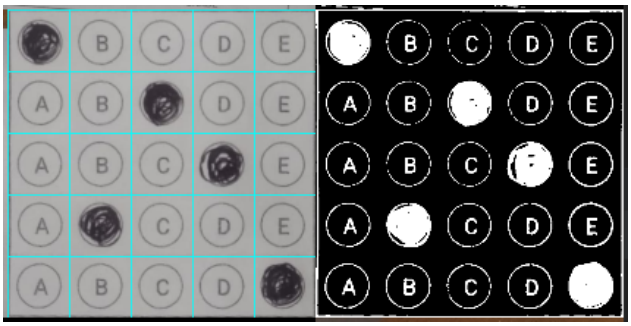

Figure 17: Threshold

The bubble evaluation was like this.

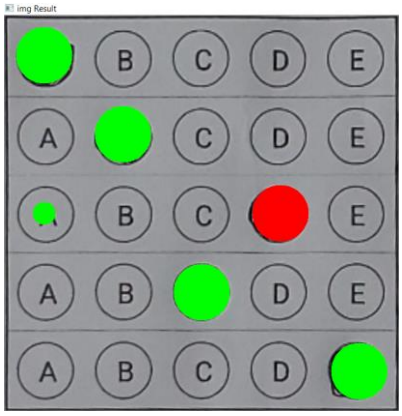

Figure 18: Bubble Evaluation

Finally relaying the evaluation and grade on our final image gave a result like this

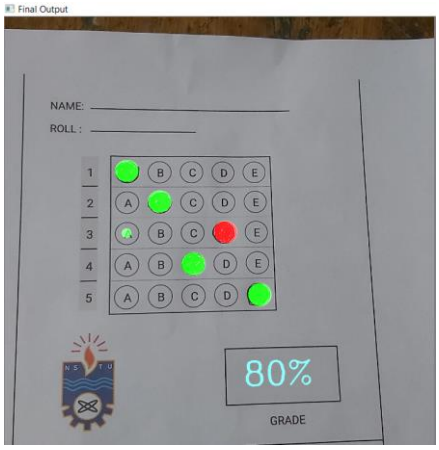

Figure 19: Final Output

5.2.2 Result Analysis via graph

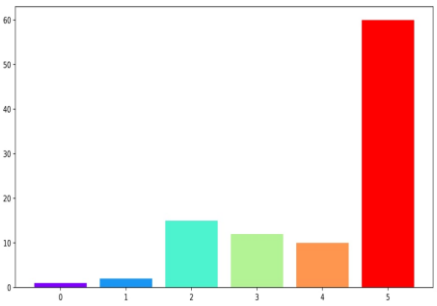

Figure 20: Bar plot of the result

The figure bar plot and pie chart and ascending graph represent that,

60 students got all 5 answers correct
10 students got 4 answers correct
12 students got 3 answers correct
15 students got 2 answers correct
2 students got 1 answer correct
1 student got no answer correct

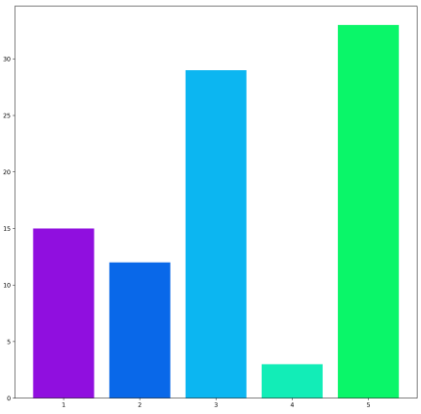

Figure 21: Comprehension Analysis

The bar plot above describes that

1 number question answer was mistaken by 15
2 number question answer was mistaken by 12
3 number question answer was mistaken by 29
4 number question answer was mistaken by 3
5 number question answer was mistaken by 33

So from this data examiner can conclude that the topic involving questions 3 and 4 are mistaken by the student most. Maybe the topics need extra care.

5.2.3 Edge Cases

We considered some interesting edge cases. These edge cases are provided below.

Double Marked: We set up a minimal threshold for the box. If it reaches 30% of the total pixel count. The bubble is considered selected. And if there are multiple bubbles selected then the answer is considered wrong.

Figure 22: Double Filled

10%-30% Marked: If the bubble is filled up until 30% then, the bubble is considered not marked.

Figure 23: Partial Filled

Bubble Edge Marked: Some examinee tends to start from the edge of the bubble. And stop at second thought. Since the edge is only 10-20% of the total pixel, the bubble is considered not marked.

Figure 24: Edge Filled

50% Marked: If the examinee fills the bubble half. It is considered as the marked bubble. And two marked bubbles in the same question is going to give a wrong answer.

Figure 25: Half Filled

Random Mark: If the examinee makes some light mark on any bubble, as long as it doesn't reach 30% the mark is ignored.

Figure 26: Random Lines

5.3 Discussion

The proposed method is likely to have some drawbacks. Having a good image is essential for image processing. Our output is as good as our input. A good image has some parameters in this case. Such as clarity, pixel density, and colour.

If the image has a sharp edge it is easily detected. Where a blurry image while going through Gaussian blur, will output a non-detectable image. This is also true for another method of grading the OMR sheet. But these methods are more expensive in terms of device. And procedure.

Pixel density and good colour gradient can be obtained with an average smartphone camera. Since in recent year smartphone has become more available. Ensuring a quality image is easier and cheaper.

Zeki KÜÇÜKKARA, Abdullah Erdal TÜMER (Tümer & Küçükkara, 2018) proposed paper had a major drawback. It was carried out with specialized paper and QR codes. And printing QR code requires a quality printer. And students can damage the QR code unintentionally. And even after the proper care of the paper, the answer sheet was scanned with a scanner. And that brings up the cost. But our proposed method tackles these problems. We used a simple Graphical User Interface (GUI) to handle the answer. And from our scanned result, the method didn't require a scanner. An average smartphone camera did the trick (ensuring a clear image). And our proposed method can grade the answer sheet under low light as well.

Bhoj Bahadur Karki and Nirman Giri proposed a paper (Karki, 2018) and their paper compared different mechanisms to scan the OMR document. And they provided the following table:

Paper	Main Algorithm	Accuracy
A low-cost OMR Solution for Educational Application	flag point searching	98%
Robust and Low-Cost Optical Mark Recognition for Automated Data Entry	backward difference method	100% (for horizontal search), 85.72% (for cross mark search),
A generalized approach to Optical Mark Recognition	Circle generating algorithm	99.20%
Cost-Effective Optical Mark Reader	Windowing technique	100%
Automatic OMR Answer Sheet Checker	Correlation Algorithm	100%
Implementation of OMR technology with the help of the ordinary scanner	Windowing Technique	100%
Scanner Based Optical Mark Reader	Region Separation	100%

Table 3: Comparison between different model

Our Image processing method can provide 100% accurate results. It is a low cost and portable solution. On that basis, we can claim our method to be more effective than the above method.

This is to be noted our method is the best fit for low to the mid-impact scenario. This will boil down to the question of scalability. For thousand plus scan and high stake cases we recommend using an OMR machine.

CHAPTER 6

CONCLUSION

The work we did can come to use by many people. Especially if someone preparing for a test. The biggest aspect of this method is the feedback loop. By analyzing the result and graph, anyone can easily identify the weakness in the respective topic. Now given the rise of cloud computing, it is not even necessary to run this code on a pc. A mere picture can do the trick. The computation can be done on the server-side. By uploading the image on the cloud server user can get the data back.

We implemented the method for the 5/5 grid answer sheet. In future, we hope to scale it into a more mainstream format. And the rise of cloud computing and the cheap cost of it is going to enable us to host this computation in the cloud. And user can grade their MCQ sheet by uploading it on the server.

This method is a new system for OMR with the help of image processing and computer vision. Of all 100 MCQ paper scanned we were able to extract the data. And got 100% accuracy in doing so. Our proposed method can also be developed and implemented on a smartphone device. MCQ based exams are increasing over time. And this method is helpful for both the examiner and the examinee. It needs to be noted that MCQ tests can be taken without the need for any paper at all. But this traditional method is mostly used for competitive exams. Bangladesh University Admission tests are taken through this method. A student while taking preparation for such an exam takes several tests. But grading it manually and then analyzing the data is tedious and time-consuming. But having an OMR scanner or any scanner is not feasible for a lot of the students. Plus the proposed method extracts data from the OMR sheet in a more flexible manner. So the

student can analyze the data and figure out where he lacks preparation. The proposed method would help a lot of institutions to use this as their daily needs. This will most likely save a lot of their time and money.

REFERENCES

Bradford, W. (1974). Principles of optical scanning systems.

Bratianu, C., & Dinca, V. M. (2010). Knowledge Economy Dimensions. *Revista De Management Comparat International/Review of International Comparative Management*, *11*, 210-221.

Canny, J. (1986). A Computational Approach To Edge Detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *PAMI-8*, 679 - 698. https://doi.org/10.1109/TPAMI.1986.4767851

Chinnasarn, K., Rangsanseri, Y., & Thitimajshima, P. (1998, 12). *Removing salt-and-pepper noise in text/graphics images*

Demush, R. (2019). A Brief History of Computer Vision (and Convolutional Neural Networks). https://hackernoon.com/a-brief-history-of-computer-vision-and-convolutional-neural-networks-8fe8aacc79f3

Gaikwad, M. S. B. (2015). Image Processing Based OMR Sheet Scanning. *4*(3). https://www.pdffiller.com/jsfiller-desk10/?projectId=621e2ce764141b369b068eb4&lp=true

Gowda, S., & Amin, Y. (2018). An Image Processing Oriented Optical Mark Reader. *5*, 258.

Gupta, A., & Avasthi, S. (2016). Image based low cost method to the OMR process for surveys and research. 2395-3470.

Hasan, R., & Abdul Kareem, E. (2021). An Image Processing Oriented Optical Mark Reader Based on Modify Multi-Connect Architecture MMCA.

Hompel, M., Masoudinejad, M., Bousbiba, O., & Roidl, M. (2019). *Automatic Identification Technology*. https://doi.org/10.1007/978-3-319-92447-2_30

Jain, R. (2021). *Getting Started with OpenCV*. https://doi.org/10.1007/978-1-4842-7274-9_8

Karimi, Z. (2021). Numpy Package.

Karki, B. (2018). Comparative study on Optical Mark Recognition, published on Nascoit. https://doi.org/10.13140/RG.2.2.16646.11849

Khatri, N., Dasgupta, A., Shen, Y., Zhong, X., & Shih, F. (2022). Perspective Transformation Layer.

Labs, N. (2022). Python Programming.

Maniar, S., Parmani, J., Bodke, M., & Saxena, K. (2021, 07). *Generation and grading of arduous MCQs using NLP and OMR detection using OpenCV*

Marr, B. (2019). *7 Amazing Examples Of Computer And Machine Vision In Practice*. https://www.forbes.com/sites/bernardmarr/2019/04/08/7-amazing-examples-of-computer-and-machine-vision-in-practice/#3dbb3f751018

McKeown, C. (2018). *Input Devices*. https://doi.org/10.1201/9780429055935-6

Nitzberg, M., Mumford, D., & Shiota, T. (1993). *Finding contours and junctions*. https://doi.org/10.1007/3-540-56484-5_3

Ramamoorthy, S., Rajaram, S., & Abbas, S. A. (2018). *Big Data in Medical Image Processing*. https://doi.org/10.1201/b22456

Sampaio, M., & Kroger, P. (2016). Contour Algorithms Review. *Brazilian Journal of Music and Mathematics*, *1*, 72-85.

Technologies, P. *Image Processing Algorithms*. https://www.pre-scient.com/knowledge-center/image-processing/image-processing-algorithms.

Tümer, A., & Küçükkara, Z. (2018). An Image Processing Oriented Optical Mark Recognition and Evaluation System. *International Journal of Applied Mathematics, Electronics and Computers*, *6*, 59-64. https://doi.org/10.18100/ijamec.2018447788

Wikipedia. Intelligent character recognition. https://en.wikipedia.org/wiki/Intelligent_character_recognition

Wikipedia. Optical character recognition. https://en.wikipedia.org/wiki/Optical_character_recognition
Zhang, Z. (2014). *Perspective Transformation*. https://doi.org/10.1007/978-0-387-31439-6_116