MCQ's Evaluation using Python OCR: An Algorithmic Implementation and Design Approach

Noreen Malik, Muhammad Adnan Kaim Khani, Dr, Asad Ali Shaikh, Allah Bachayo Brohi, Abida Luhrani

Department of Computer Science, Ilma University, Karachi.

E-mail: noreenmalik154@gmail.com, m.adnan@ilmauniversity.edu.pk, dr.asad@ilmauniversity.edu.pk, allahbachayo@ilmauniversity.edu.pk, abidaluhrani@gmail.com.

Corresponding Author: Noreen Malik, noreenmalik154@gmail.com,

Received: 19-04-2023; **Accepted:** 11-06-2023; **Published:** 01-07-2023

Abstract: Today, Multiple Choice Questions (MCQs) are being widely used as an effective way to assess or to grade high school and university students.MCQs has become a fast and reliable method for national entrance examination over the world, particularly in Pakistan because it can assess students with the broad range of knowledge coverage within a limited time. However, when the use of MCQs is more demanded, a manual grading solution seems harder. Although technology for automatic grading of multiple-choice questions has existed for several decades, it is not yet as widely available or affordable as it should be. The main reasons preventing this adoption are in the cost and the complexity of the setup procedures. Hence, in this research, we propose a cheaper alternative to grade multiple choice questions, an OCR based solution built in python, which will use desktop scanners for image acquisition. OCR Algorithm takes advantage of Python programming language for its cross-platform, open source, and community-based environment. In this system desktop scanners will be used, as they are cheaper and widely available. In this algorithm, OpenCV (Open-Source Computer Vision Library), an open-source computer vision and machine learning software library will be utilized, to OCR scanned images of answer sheets.

Keywords: Multiple Choice Questions (MCQs), Desktop Scanners, OCR Algorithm and OCR scanned images.

1. Introduction

This chapter provides a brief introduction of automated processes to generate multiple choices question results, Optical Character Recognition (OCR) technology and tools to help in automating the process. Today, automated processes are used in many areas of our life, which once required manual work. In this modern era, conducting exams and analyzing results remains complicated and time-consuming process. On average, 95% of teachers grade their students' tests manually [11]. Unfortunately, the process of conducting exams and analyzing results consumes too much time, teachers lose the opportunity to address students' comprehension gaps, quickly. Traditionally, conducting these types of exams required the purchase of an expensive and potentially complicated dedicated scanning system and a large investment in time on the part of the teachers giving and grading the exams. Additionally, the costs of these dedicated scanning systems were not and still being not limited to the initial purchase. The costs often include the purchase of preprinted standardized forms as opposed to, on demand printing of bubble sheets and require special software to run the scanner and produce the reports. There is a recurring cost of maintenance and the cost of support these systems need from it. Training can also be time-consuming and difficult, since certain software packages can be complicated. These solutions may also have limitations in their test assessment capabilities and/or their ability to provide educators with other essential forms. Fortunately, the liabilities associated with bubble sheet testing and multiple-choice testing are being addressed. Starting with the composition of the tests themselves, but also including some software and hardware solutions that now add efficiency and cost savings to what used to be a difficult process. From an assessment standpoint, test designers are now producing more sophisticated multiple-choice tests. These

well-designed multiple-choice tests can require considerable thought, even notes and calculations, before students can choose the proper bubble.

1.2 General Process of Generated Output from MCQ's

The recent development of desktop scanners, and the advancement in technology the scanner now has the capabilities of the earlier larger Optical Mark Reader (OMR) machines [34], but at a much lower price for smaller volume. The users have now made it possible for educationalists to utilize computer testing techniques. Therefore, the purpose of proposed research is to design and implement a fast and accurate python-based OCR software, which utilizes desktop scanners and OCR technology for grading multiple-choice questions accurately while being cost efficient.The general/ natural process of conducted exams and getting output/generated automated results of exams as follows:

1. Answer Sheets are designed and delivered to the students.
2. Multiple Choice Question paper delivered to the students.
3. After the exam, every answer sheet is scanned and saved to a folder.
4. Finally, all scanned sheets are given as an input to the automated system to generate results and finally results are collected in csv format.

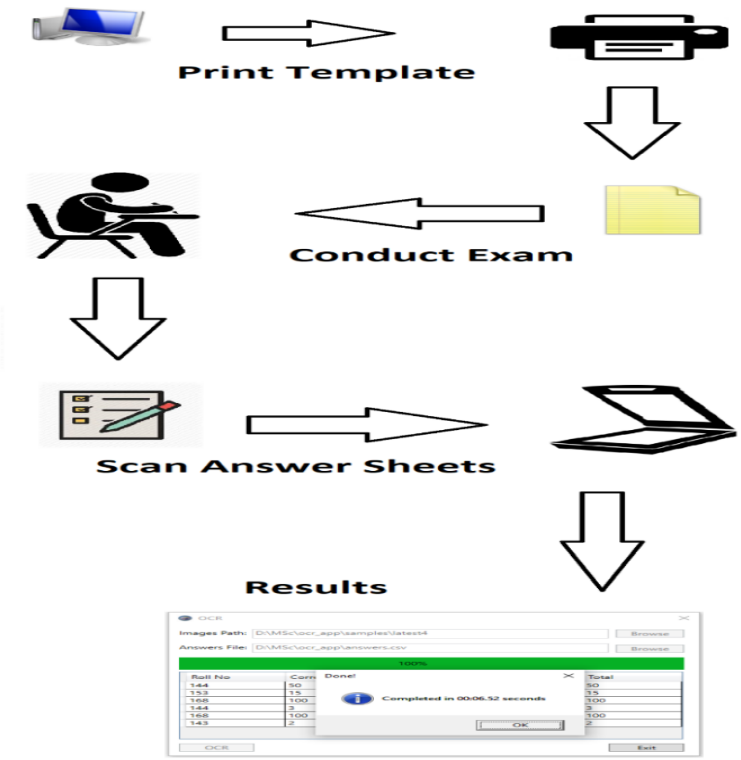

Figure 1. High-level architecture of proposed system

1.3 Tools and Technologies

It is necessary to survey the tools that can perform very well for the scanning and generating the output efficiently. There are vast variety of tools and technologies available for image processing, answer sheet design and IDEs each with its own abilities and drawbacks. For this, the tools and technologies are carefully considered to achieve the goal.

1.4 Python Programming Language

Python (Python, 2017) was first released in 1991[28]. It is a high-level programming language which is widely used for general-purpose programming. Python is an interpreted language; Python is designed on the philosophy which emphasizes code readability. Python uses whitespace indentation to delimit code blocks rather than curly brackets or keywords. Python has a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. The decision to use python for this research was made because it has great support for OpenCV, only other option in this regard is C++ since, python is higher level language than C++ which makes it easier to code also, python provides C and C++ based libraries like numpy and scipy which drastically improves the performance of the applications built in python. Python version 3.6.6 is used in the development of this algorithm.

2. Related Work

2.1 Related Work OMR based Solution

A framework for grading multiple choice questions is presented by[22], the framework uses a combination of Octave scripts and spreadsheet software to perform Optical Mark reading (OMR) and automatic scoring. The framework was able to deliver high scoring accuracy and significantly increases the productivity of teachers. It also improves the accessibility to automatic scoring of MCQs.The Octave OMR scripts, while currently in non-optimized form, can process each answer sheet with 120 items in 1 minute and 8 seconds. Not everyone will find the current implementation of the framework easy to use as it was not included in the initial priorities of the work. It assumes that the user is well-versed with formula-based processing with MS Excel or compatible software and running scripts on Octave. However, many aspects of the framework can be readily enhanced to improve ease of use and processing speed. [5]successfully replaced OMR machine at the Brazil's Federal University of ABC with their OMR system implementation in python. The application simplified the process of generation and correction of MCQs (multiple-choice questions). [2] proposed a Hough Transform based system to grade multiple choice questions with a claim of 100% accuracy which perfect for grading multiple choice questions. Despite having excellent accuracy, the system lacks in performance with an average time of 12 seconds, a maximum time of 29.269 seconds per answer sheet and a total of 49.345 minutes for 247 answer sheets. Since OMR machines are used to grade large number of answer sheets with high speed, the proposed system is not feasible in terms of performance where large quantity of answer sheets are to be graded. [34] developed an Arduino based Optical Mark Reader (OMR) with a combination of other devices like- MUXs, switches, LM324ICs and an Infrared based reflective sensor. The system has an accuracy of 100%. The drawback of this system is that Arduino and its different components are not readily available in institutions and setup and usability is complex. An application for OMR is presented[13]. The application is developed in C# and OpenCV is used for image processing. The drawback of this application is the use of C# Programming Language which can get slower in image processing unlike Python's numpy arrays which are developed in C and C++. Authors did not include metrics for speed of the application or accuracy of the application.

An improvement of an existing OMR system is proposed[35]. the proposed improvement over an existing OMR system is an additional OCR process, which gives students an option to annotate wrong marked answer and write a corresponding letter which the OCR process will read and determine it to be correct answer. The improvement is great which gives students the ability to correct a mistakenly marked answer however, the main benefit OMR is its performance over OCR. Therefore, introducing OCR functionality in OMR system greatly reduces performance of the whole system.[15]presented a novel approach in Optical Mark Recognition by using a low-cost Field Programmable Gate Array (FPGA), the presented system gave an accuracy of 100% with a minimum processing speed of 45,000 pages/h. Though, the system achieved staggering processing speed of 45,000 pages/h, to design a paper feeder for such a system is a very difficult task, also common paper feeders may pose a problem for this system because of the misalignment of the pages which authors mentioned in their research. [6] developed an OMR (optical mark reader) based system to recognize filled answers from specially designed answer sheet for grading multiple-choice exams. The development environment of the system consisted of an image scanner connected to a PC-type microcomputer. The system had two modes of operation: learning mode and operation mode. In the learning mode, the system would recognize all significant horizontal and vertical lines in a blank sheet image then construct a model of the answer sheet and save it for later use of OMR. After that, a rectangular area is constructed by recognizing all possible crosslines in the answer sheet. The operation mode operates by feeding each answer sheet into the system to identify the matching horizontal lines from every model, a projection of histogram is generated from horizontal and vertical space of the area. Then, to recognize the filled answers, the number of black pixels from each answer is

counted. In the next step, a decision is made by taking the difference from input and its corresponding model of the answer sheet. Finally, a database of students, scores and list of subjects is created. The system displayed its effectiveness in the experimental results of many styles of answer sheets.

2.2 Related Work on Image Processing and Camera based Solution

[5]developed an OMR based android application named "MCTest - Multiple Choice Test" using Java programming language for automatic grading of multiple-choice tests. The application captured live video feed from the camera and each individual frame was preprocessed using the OpenCV library. They used Mathematical Morphology technique for image segmentation, in order to circumvent situations where images were captured under non-ideal conditions. The multiple-choice test sheets were corrected instantly in real conditions and provided statistical tools to analyze the results. The authors claimed trials performed in dozens of tests have shown that MCTest presents an accuracy rate of up to a 100%, even when considering user-related errors, such as poorly filled tests. The use of camera instead of a scanner makes it prone to problems relating to bad lighting and poor-quality cameras reduces the recognition rate and makes it difficult to implement such a system.

[12] built an automatic grading system named Eyegrade, which uses webcam to capture exams. The system performs both optical mark recognition as well as optical character recognition of handwritten student identification numbers, which avoids the use of bubbles in the answer sheet. [25] presented a novel approach inspired by nature's way of distributing species known as Biogeography Based Optimization Biogeography, the algorithm first converts the image into a binary image than finds concentrated regions by collecting concentration of pixels of different regions in a document to detect marks. No real implementation was done, no results were discussed regarding accuracy or performance of the given algorithm. [33] built a system named JECT-OMR in Python and used Gamera Framework for image processing. The system was designed for two types of answer sheets, one which has squares that can be crossed to mark it as an answer and the other answer sheet with circles that are filled to mark them as an answer. The system had an accuracy of 99% in recognizing squares, authors did not mention the accuracy of the answer sheet with circles. Since, OpenCV is de facto standard for image processing which is fast and efficient library, authors used Gamera Framework for this purpose and did not mention performance of the system.

[22] proposed a camera based system for grading multiple-choice tests. The system used an automatic paper feeder and a camera for image acquisition. Three types of cameras were used to test reliability and efficiency of the system. To detect and correct tilt of the answer sheet, Hough transform was used, then images were scaled to a given size. To detect tick mark corresponding to the answer for each question a mask was wrapped over the answer area. The authors claimed better accuracy, reliability and elapsed time than conventional optical mark recognition (OMR) systems with an accuracy of 99.7%. The speed metrics of the system were never discussed also, programing language in the implementation of this system was not mentioned.

An algorithm for preprocessing scanned documents is proposed [14]. As a preprocessing step algorithm, it works to improve the accuracy of text extraction from document images. The algorithm is used to accurately extract text from digital images, which are poorly scanned, has many distortions, and are poorly scanned text-photo images or natural images. The results showed that the algorithm is 2 to 6.8 percent more efficient on accurate text extraction. [30] presented their study on the comparison of techniques used in non-natural documents image denoising. They concluded that wavelet-transform-based techniques were effective in denoising the images. These techniques are effective for their essential properties of multiscale nature, multiresolution, and sparsity. Saengtongsrikamon et al. (2009) performed their research work to replace OMR with an application which they built in Java and claimed to be accurate with error rate of about 1 in 1000 or 0.1 percent. Their error rate is within acceptable limit which make it feasible to substitute OMR machines, but they did not specify the performance of the system, which is a critical issue in replacing OMR machines. Since, they did not use open source image processing libraries like OpenCV and did not use languages such as python instead used Java programming language makes it difficult for other researchers to improve the application to make it a suitable candidate in replacing OMR machines.

Bienieckiet al.(2007) argued in their research, that image-preprocessing phase is extremely important for OCR than actual recognition phase. For experimental evaluation, they used FineReader 7.0 for OCR. In their experiments, first they scanned a Polish book than introduced various image noises and image deformations in scanned images using Paint Shop Pro 9 software. Finally, they concluded that appropriate preprocessing algorithms should be used in preprocessing phase before performing actual recognition. In their experiments, OCR was more sensitive to geometric deformations than noise and low resolutions. [2] proposed a new and comprehensive solution for grading of multiple choice questions by developing an automatic grading system based on Circular Hough Transform and Casual Median Filter. Although, the system was tested on more than 2500 exam papers with 3 case studies performed, only performance of the system was analyzed, and accuracy was not discussed except in case study 3 author claimed an accuracy of 100%. The system was developed in MATLAB with case studies performed in MATLAB environment

and no general-purpose language was used like Python to make an application to be deployed and tested in real condition environment.

[7]aimed at reducing the cost of OMR machines, implemented a Template Matching algorithm with an accuracy of 100%, to grade multiple choice questions. Authors used X marks instead of circles or bubbles in answer sheets and template matching is used to detect X marks in selected answers where a match is detected by repeatedly shifting the template up, down, left and right until a satisfactory match is found. Since, templates are repeatedly shifted in each direction with differing number of pixels to get a satisfactory match, it can deistically reduce the speed of this algorithm, which authors failed to include in their research. Usage of Python and OpenCV for hand segmentation is presented by [20] the study performed using Kinect depth sensor along with Pykinect and OpenCV to get accurate hand segmentation.

2.3 Related Work on Simple Scanner based Solution

Nerkar (2015) built a software in C# programming language with AFORGE .Net for image processing, the software had the ability to work with any kind of desktop scanner, they had used HP scan jet 2410 scanner, the cheapest scanner available at the time. Though, authors claimed an accuracy of 100% which is comparable to an OMR machine, they did not mention the performance of the system which may be a hindrance in adoption of the software in institutions. Since, they did not use platform independent languages like python, their software can only be used on windows based systems[9] proposed a novel approach based on Digital Camera rather than desktop scanners for image acquisition and for Optical Mark Recognition built their source code in MATLAB. They designed a template of answer sheet with 24 questions and 5 squares for each possible answer, the size of the squares in answer sheet was quite big and filling it was cumbersome. A novel method of image registration is proposed[17], the method employs a combination of adaptive polar transform (APT) and an innovative projection transform along with a matching mechanism to register images with more accuracy and higher efficiently than traditional log-polar transform (LPT)[10] evaluated the drawbacks of current OMR technique. They argued that current OMR techniques are unable to accurately recognize thin papers so; they presented a new image based Optical Mark Recognition technique that was able to OMR lowprinting precision questionnaire documents and thin papers of 70 grams or less. The authors claimed 98% success rate and about 200 million questionnaire sheets were processed with their technique.

2.4 Research Gap

From the review of literature, it is notable that most of the work on OCR based solution is conducted on either OMR or Camera based solutions. The less work has been conducted on the simple scanner. Our research objective is to improve the efficiency of the simple scanner-based solution for OCR based MCQ's exams.

3. Methodology

This chapter provides methodology of our proposed approach, the complete process and the casestudy is discussed in this chapter.

3.1 Overview

An answer sheet is required in the first step for the design and implementation of this algorithm, the template of answer sheet for multiple choice questions is designed in Microsoft Word, which has four tables of 25 questions each with a total of 100 questions and one table for roll number. This template is used to conduct multiple choice questions exam, later these answer sheets are scanned and saved in a folder, which is then used as input to this application to perform OCR. After performing OCR, this application grades correct answers corresponding to given answers and generates output in a csv file containing roll numbers and their grades or marks. A desktop application is implemented based on this algorithm to grade multiple choice questions. The OCR engine is implemented in Python Programming Language for its readability, maintainability and itshigh performance libraries such as numpy, scipy, OpenCV, etc. The Graphical User Interface for this application is implemented in C# programming language for windows-based systems. The inter-process communication between OCR engine (Python) and GUI (C#) is made using Standard I/O for its low latency and high performance. In the next sections, all processes are deeply described based on the research methodology.

3.2 Template Design

A template is designed in Microsoft Word to conduct exams based on multiple choice questions, this template has 100 questions with four choices of possible answers for each question to mark and a table for the marking roll number. The template is shown in Figure 2.

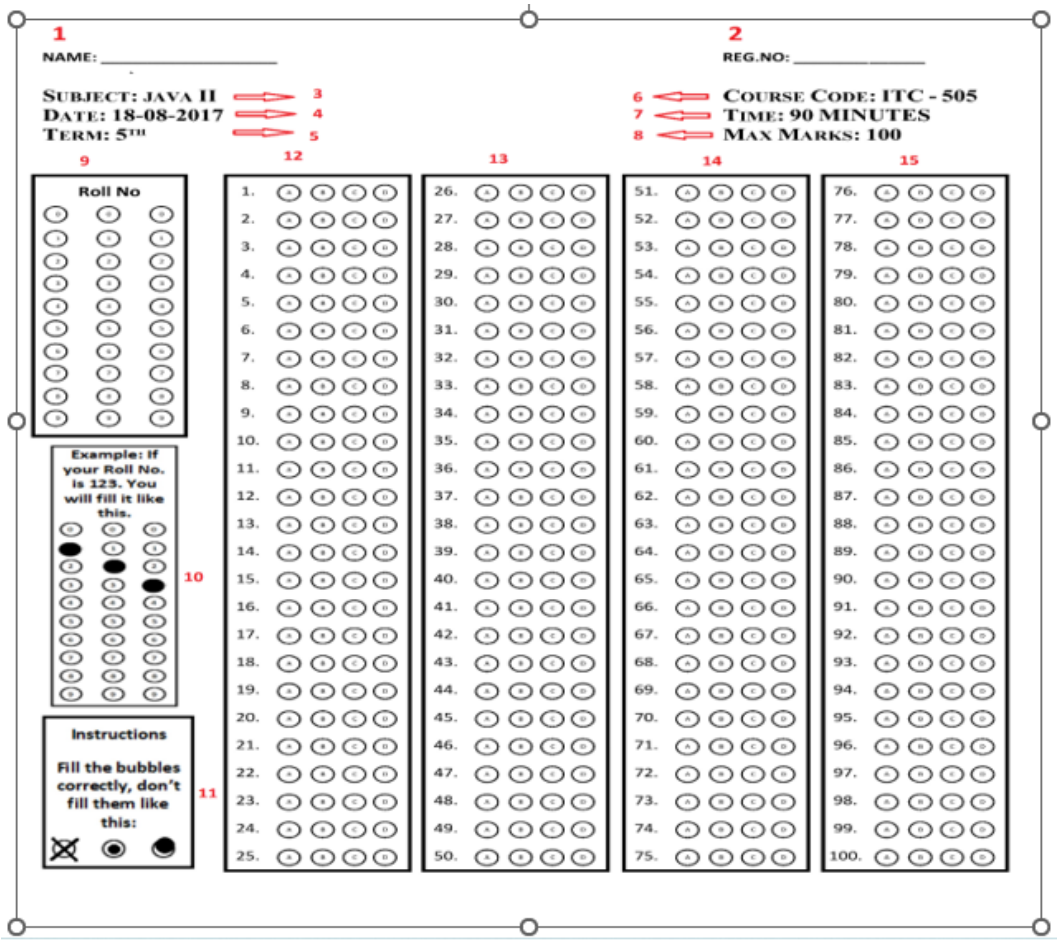

Figure 2. Answer Sheet

The Figure 2. is described below:

1. An area to write Name
2. An area to write Registration Number
3. Subject Name
4. Date of the exam
5. Term or Semester
6. Course Code of current exam
7. Total time for the exam
8. Maximum Marks for the exam
9. A table for marking roll number
10. An example for filling roll number
11. General Instructions
12. Table 1 for markings questions 1 through 25
13. Table 2 for markings questions 26 through 50
14. Table 3 for markings questions 51 through 75
15. Table 4 for markings questions 76 through 100

3.3 Image Acquisition

There are manual scanners and scanners with Automatic Document Feeder, since, this application performs OCR on scanned images of answer sheets, it can OCR regardless of which type of scanner is used.

3.4 Algorithm Design

In the first step images are read from disk since, scanned images are usually colored images, the first preprocessing step is to convert every input image into a grayscale image. Grayscale images are images with different shades of gray, grayscale images have values ranging from 0 to 255 where a value of 0 is darkest level.This is represented as black and a value of 1 is brightest level represented as white. The values between 0 and 255 have different levels of gray. Grayscale images are used to resemble colored images in intensity of pixels. Majority of Computer Vision algorithms works on grayscale images so, this preprocessing step is necessary since, it preserves the details of images and makes it easier and faster for algorithms to implement compared to the colored images.Illustration of a grayscale image. In the second step since, scanned images can be of different dpi, depending on the scanner or options set by the user. Hence the larger and better-quality images tend to be slow to process while small and low-quality images are faster to process, though smaller images have degraded quality, in case of OMR marks can still be detected from scanned images. Utilizing this fact, in the second step with each image is scaled up or down i.e., smaller images are scaled up or zoomed using Bicubic Interpolation and larger images are scaled down or shrunk using Inter Area Interpolation, resolution of 779 x 566. Bicubic interpolation is used for zooming smaller images and Area interpolation for shrinking large size images. In the third step, tables are identified with their borders using Canny Edge Detection algorithm[4]. The Canny Edge Detection algorithm works by first converting the input image into binary using Binarization. The binarization is a technique which is used to convert grayscale images to black and white image, where every pixel with an intensity value less than the threshold is changed to black and the intensity value greater than the threshold is changed to white. The technique of binarization is really important step in image preprocessing and to correctly binarize an image depends upon the correct value of threshold, there are quite a few binarization algorithms, but this algorithm uses Otsu's Binarization Algorithm[24] which determines the threshold of the whole image. Since scanned images are used by this application which don't have shadows, illumination or other artifacts, Otsu's binarization works really great in this situation. Also, the canny edge detection algorithm requires low and high threshold values as input to binaries the image, Otsu's Binarization Algorithm is used to find high threshold value than to get low threshold we simply multiply 0.50. The fourth step is only taken, if tables are not detected in the previous step, there can be many reasons for previous step to not being able to detect tables. It can be that the images are not similar which means that image is not based on the template shown in Figure 3.1 or lines of the tables are not dark enough because of poor quality print or much noise is added by the scanner. This step determines if the image is based on the template shown in Figure 3.1, if it is not based on the template shown in Figure 3.1, an error is raised and the application proceeds to next image. In this step, a copy of the template is used to align and overlay the image on top of base template, the technique which is used to do overlay the image is described[29] and its implementation in python is available at https://github.com/matejak/imreg_dft developed by https://github.com/matejak. After aligning the image, co-ordinates of tables with respect to base template are used. After finding co-ordinates for each table in the previous step, all these tables are cropped and aligned either with perspective transformation or with affine transformation.

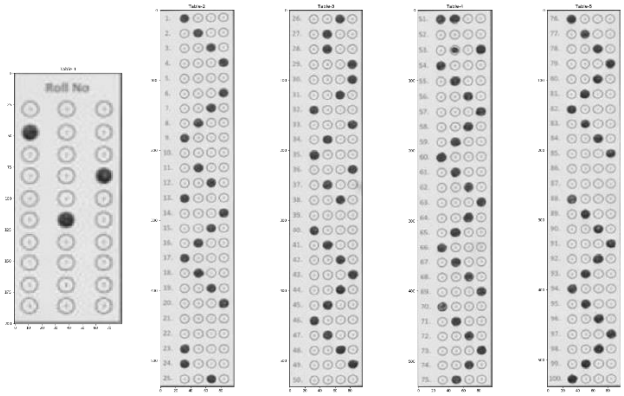

Figure 3. Cropped tables

Then, each table's bubbles are identified using edge detection and by masking the area to calculate the number of pixels to recognize it as marked area. After identifying all marked regions, question numbers are identified by y co-ordinates based on the center of a bubble and their possible marked answers are identified by x co-ordinates based on the center of a bubble, for roll numbers table it's x co-ordinates are divided into 3 columns and roll number is identified by center of a bubble's y co-ordinate. In the final step dilation is applied on each cropped table and previous step is repeated to find bubbles which might have been missed by previous step because of a condition where bubbles are overfilled, this step ensures that all filled bubbles are identified. After that, results which we get with and without dilation are filtered to generate correct result. The activity diagram of all these steps of this algorithm. The source code behind activity diagram is described here: The source code to perform actions. The function takes the path of an image as parameter, then loads the image with OpenCV's imread function after that converts the image into grayscale image and returns the resultant image.

```
def resize_image(image):
    """
        This function scales image up or down to 779x566 resolution.
        Returns:
            Scaled Image.
    """
    desired_height, desired_width = 779, 566
    height, width = image.shape
    height_ratio = height / desired_height
    width_ratio = width / desired_width
    if height_ratio > 1 and width_ratio > 1:
        image = cv2.resize(image, (desired_width, desired_height), interpolation=cv2.INTER_AREA)
    elif height_ratio < 1 and width_ratio < 1:
        image = cv2.resize(image, (desired_width, desired_height), interpolation=cv2.INTER_CUBIC)
    return image
```

Figure 4. Scale Image

The function shown above in Figure3.8, takes an image as input and returns the resized image. The function resizes input image to a constant resolution of 779x566 resolution, if an input image is bigger than the constant resolution, the image is resized with OpenCV's resize function using INTER_AREA interpolation method, on the other hand if the input image is smaller than the constant resolution, the image is resized with OpenCV's resize function using INTER_CUBIC method. The source code to perform action 5 is shown in the activity diagram. The function shown above in Figure3.9, takes an image as input and returns the co-ordinates of each table in the image. To find the co-ordinates of each table in reference image, Canny Edge Detection is used. Since, Canny's Edge Detector takes the low and high threshold values of the image as input, we find these low and high threshold values using Otsu's Binarization Technique in line 59, then execute Canny Edge Detector function available in OpenCV in line 60. OpenCV's Canny function returns an image of only edges in the original image. After getting the edged image, we use OpenCV's find Contours function in line 61 to get all points of the edges. Next, we filter contours and sort them from line 62-63. In line 65 we iterate over each contour to find desired edges with desired height and width then return the resultant points of each table. The source code to perform action 6 is shown in the activity diagram. The function shown above in Figure3.10, takes an image as input then performs DFT (Discrete Fourier Transform) based image registration and returns the corrected image. The base template is loaded in 85 as shown in Figure above.

The functionality after that is modified version of the original implementation of DFT based image registration in python by GitHub user https://github.com/matejak, original implementation is available at https://github.com/matejak/imreg_dft.The source code to perform action 7shown in activity diagram. The function gets tables for questions to recognize the marked answers associated with the questions and compares the given answers with given array of correct answers then returns the total score of all marked questions. This function accepts a Boolean parameter which specifies whether to dilate the image or not, if it is true then the image is dilated first than other actions are performed. In line 188 and line 190, the image is binary inverted. After that outlines of objects in the image are recognized and in the following steps these outlines are filtered to only get those outlines which represent the bubbles. Then each bubble is iterated, and a mask is drawn over an empty image to get the number of pixels in that bubble, if the number of pixels is greater than a constant value it is recognized as a filled bubble.

To get the question number and answer (A, B, C or D) associated with that bubble, OCR_question function is called which in turn returns the recognized question number and associated answer. The functionality of OCR_question function. A For loop iterates for 100 times at line 304, then results are filtered from both arrays to generate the correct result.

3.5 System Requirements

Processor:1 gigahertz (GHz) or faster 32-bit (x86) or 64-bit (x64) processor

Operating System: Windows 7 or later

RAM: ~200 MB per CPU core

Hard Disk Space: ~400 Megabytes

3.6 Case Studies

To get real condition samples, multiple choice questions exam is conducted for students of bachelor's and master's degree program at Information Technology ILMA University.

4. Results and Discussion

In this chapter, we will discuss results of our proposed algorithm, for these three case studies are performed, two of which are used to evaluate the performance of this algorithm i.e. accuracy and speed while the third case study is used to assess the maturity level of students in marking the bubbles correctly. To perform these case studies, an actual dataset of marked answer sheets is required, the process of collecting dataset is described below:

4.1 Dataset

To collect the dataset of marked answer sheets, two separate multiple-choice question exams were conducted for the course of Web Technology with a hundred questions each, the tests were conducted for students of bachelor's and master's degree program. A total of 79 students gave the exam, of which 64 were 2nd year students in BSIT (bachelor's in information technology) and 15 students of MSIT (master's in information technology). After conducting the MCQ exams, all answer sheets were collected and scanned with Canon's LIDE 120 model. Canon's LIDE 120 model is the cheapest scanner available; this is a manual scanner and doesn't come with Automatic Document Feeder (ADF). It's rated scanning speed with 300dpi is 16 seconds. However the actual time it took us to complete a scan with 50dpi and for the scanner to be ready for another scan was about 32 seconds. All answer sheets were scanned with Canon's LIDE 120 scanners as a photograph rather than a document and the total time it took to scan 79 images with 50dpi was about 43 minutes. After scanning all the answer sheets, the images were saved in a folder and used as a dataset for our case studies. The first criteria for evaluating this algorithm are Accuracy and it is discussed below:

4.2 Accuracy:

In context of our research, accuracy can be defined as the ability of the algorithm to correctly recognize all filled bubbles in an answer sheet. To determine accuracy of our algorithm, each answer sheet in dataset is manually checked, and results are compared with the algorithm's output. The dataset contains a total of 79 answer sheets with each answer sheet containing 100 questions with four bubbles each and 30 bubbles for roll numbers, however only 3 bubbles are to be filled for roll number and 1 bubble for each question so, ideally there are supposed to be 8137 bubbles filled for each answer sheet. Since, roll numbers are compulsory to fill and all of them are filled in this dataset but students can leave a question blank or they may fill two answers for an answer so, to get accurate accuracy, we manually count each filled bubble and compare it with the algorithm's output, the results are shown in Table 4.1.

Table 1 Summary of bubbles in answer sheet

Total Filled Bubbles	Recognized Bubbles	Recognition Rate
7883	7882	99.9%

Table 4.1 recognizes the accuracy of our proposed algorithm. It was able to recognize 7882 bubbles out of 7883 and has an accuracy of 99.9%.

4.3 Maturity:

In context of our research, maturity can be defined as, the ability of a student to correctly (doesn't fill partial neither overfill nor double answers for a question) filled bubbles in the answer sheet. The study is performed to determine the maturity level of students in filling bubbles correctly in an answer sheet. This study of maturity conducted on the students of bachelor's in information technology (BSIT) and master's in information technology (MSIT) degree program students and their respective ages range between 20-22 years old for BSIT students and MSIT students are above the age of 25 years.

The maturity attribute is measured using three metrics: these are as follows:
Metric 1: Number of approved and rejected answer sheets of the students.
Metric 2: Number of bubbles selected by the students.
Metric 3: Number of bubbles marked partially by the students.

These are further evaluated based on the actual experimental results conducted for our case study.

Metric 1: Number of approved and rejected answer sheets of the students: The first criteria in determining the maturity of students in filling bubbles is taken by the number of approved and rejected answer sheets by the application, if a student does not mark his/her roll number correctly and leaves a bubble blank for that column than the application is unable recognize the roll number, it is reported as an error so, it will be rejected. Table 4.2 shows the approval and rejection of answer sheets by this application for BSIT and MSIT students.

Table 2 Summary of approved and rejected answer sheets.

	BSIT	MSIT
Total	64	15
Approved	62	15
Rejected	2	0
Approved Percentage	96.8%	100%
Rejected Percentage	3.125%	0%

As shown in table 4.2, the approval rate of MSIT students is 100% and BSIT students' approval percentage is 96.8%. There is 0% rejection for MSIT while 3.125% of BSIT students' answer sheets got rejected. A total of 2 answer sheets out of 64 were rejected for BSIT students and none of the answer sheets were rejected for MSIT students.

Metric 2: Number of bubbles selected by the students: Second criteria of determining maturity of students is considered where a student fills more than one bubble for a question or in a roll number's table. Table 4.3 below shows the number of bubbles selected more than once:

Table 3 Summary of selected bubbles

	BSIT	MSIT
Total Bubbles	6592	1545
Double Filled	19	2
Percentage	0.28%	0.12%

Table 4.3 shows, there are a total of 6592 bubbles for questions and roll numbers where only 1 bubble is supposed to be filled however, 19 or 0.28% of selections for BSIT and 2 or 0.12% of selections for MSIT were marked more than once.

Metric 3: Number of bubbles marked partially by the students: The last criteria in this case study to determine the maturity level of BSIT and MSIT students is whether the bubbles are filled properly, or they are filled partially. Table 4.4 shows the number of bubbles marked partially:

Table 4 Summary of bubbles marked partially

	BSIT	**MSIT**
Total Bubbles	6592	1545
Overfilled	41	38
Partially Filled	98	48
Total	139	86
Percentage	2.1%	5.5%

Table 4.4 shows, there are a total of 6592 filled bubbles in BSIT and 1545 filled bubbles for MSIT. The BSIT students filled 2.1% of bubbles incorrectly and MSIT students filled 5.5% of bubbles incorrectly.

4.4 Performance Analysis

Performance of this algorithm can be defined as the speed at which the algorithm performs OCR on n number of answer sheets. A laptop with following configuration is used to analyze the performance of this algorithm:

OS Name: Microsoft Windows 10 Pro
OS Version: 10.0.15063 15063
CPU: Intel(R) Core(TM) i3-4010U CPU @ 1.70GHz
RAM: 11.922813415527344 GB
Graphics Card: Intel(R) HD Graphics Family

We measure the performance of the algorithm by comparing it with the performance of a commercially available API. Our implementation and commercially available API are both evaluated in terms of speed by calculating the time to OCR a dataset of 79 answer sheets.

Table 5 Performance of proposed algorithm vs commercial API

	Our Proposed Algorithm	**Commercial API**
Total Images	79	79
Completion Time	13.8 seconds	23 seconds

Table 4.5 shows, the total time to perform OCR of 79 answer sheets for our proposed algorithm is 13.8 seconds and total time to perform OCR of 79 answer sheets for commercial API is 23 seconds. Our algorithm is 9.2 seconds faster than commercially available.

We have evaluated our algorithm on three criterions accuracy, maturity and performance, the results of which are discussed here:

The accuracy of our proposed algorithm is 99%, in our dataset of answer sheets there was an exceptional case where a student marked the answer in which he made a hole in the answer sheet that was not detected by our proposed algorithm. Except for this single anomaly all other marks were detected correctly including over and partially filled marks.

We conducted the study to find the maturity level of BSIT and MSIT students in filling the bubbles correctly. We set three criteria of maturity; the results of each criterion are discussed below:

The first criteria in measuring the maturity level of BSIT and MSIT level is the approval and rejection rate of our algorithm as shown in Figure 5

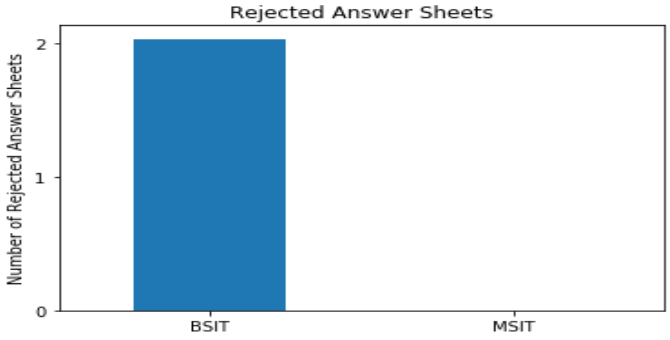

Figure 5. Maturity based on number of rejected answer sheets

Figure 5 shows that the MSIT students displayed more maturity and none of their answer sheets got rejected.

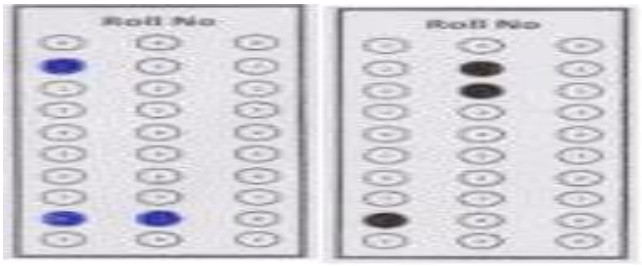

Figure 6. Example of rejected answer sheets

Figure 5 shows the area of roll numbers of the 2 rejected answer sheets. These answer sheets were rejected because the application requires all three columns to be filled and 2 students failed to fill their roll numbers properly so, their answer sheets got rejected. In the second criteria for measuring the maturity level of BSIT and MSIT level, we considered a situation where a student fills more than one bubble for a question or in a roll number's table. The Figure 6 shows that the results of the second criteria of maturity:

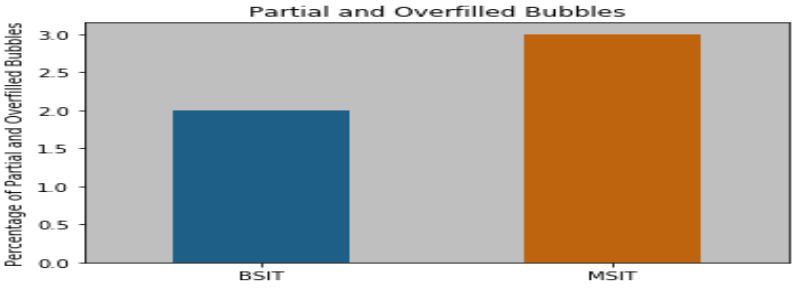

Figure 7. Maturity based on partial and overfill

Figure 7 shows that the number of overfilled and partially filled bubbles for MSIT is more than BSIT.MSIT answer sheets had 2.9% of their bubbles partially filled or overfilled while BSIT answer sheets had 2.1% of their bubbles partially filled or overfilled.

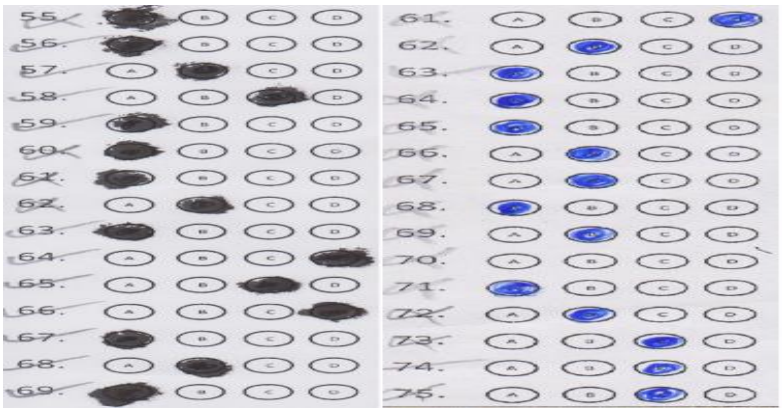

Figure 8. Overfilled Bubbles (Left) and Partially Filled Bubbles (Right)

Figure 8 shows the overfilled and partially filled bubbles. Overfilled bubbles are clearly seen while a few of the partially filled bubbles can be seen in the Figure.

Finally, the score of overall maturity of MSIT and BSIT students is shown in Figure 4.7 below:

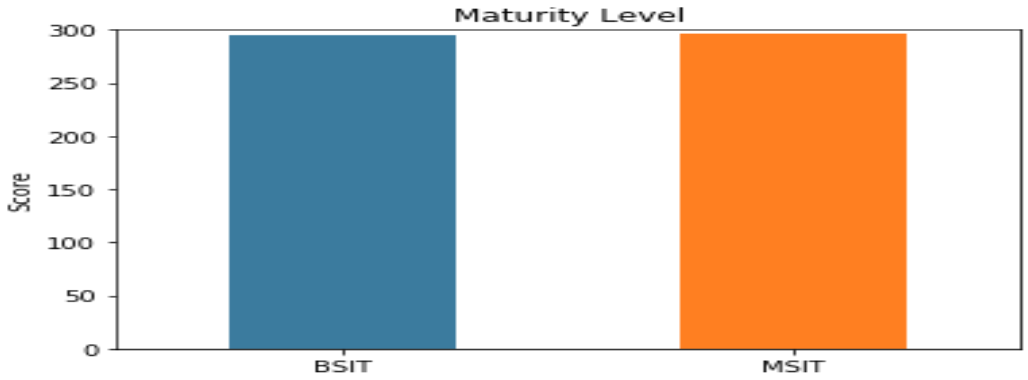

Figure 9. Maturity level of BS and MS IT students

Figure 9 shows that the overall score for both is almost equal. Both BSIT and MSIT students had equal maturity in terms of filling bubbles correctly. BSIT students got 295 score out of 300 and MSIT students got 2.97 score out of 300.

The performance of the proposed algorithm is analyzed by comparative and statistical analysis. The comparative analysis is discussed below:

In comparative analysis, we compared the performance of our proposed algorithm with a commercial API, results of which are shown in Figure below:

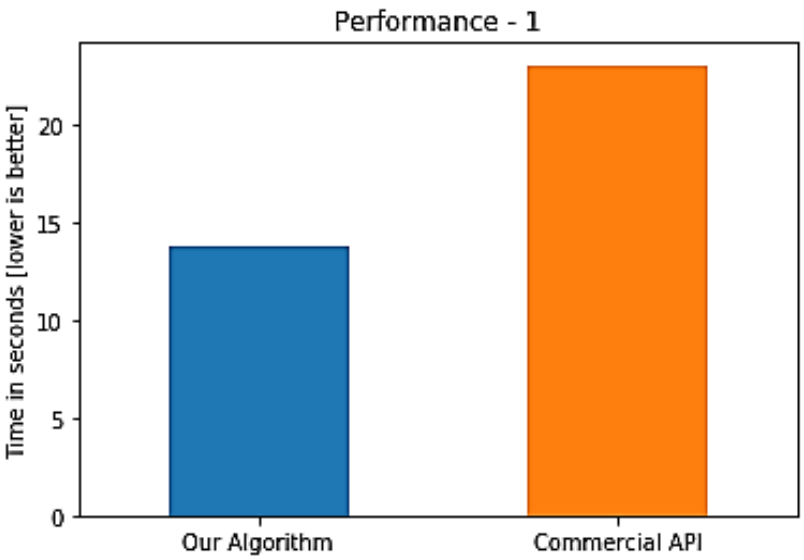

Figure 10. Performance of proposed algorithm vs commercial APIbased on time

The Figure 10 shows that our algorithm's performance is superior to commercial API with a difference of 9.2 seconds. Our algorithm took 13.8 seconds to OCR 79 answer sheets while commercial API took 23 seconds.

5. Conclusion

The research conducted on design and implementation of python-based OCR algorithm to grade Multiple Choice Questions. First, we designed a template of answer sheet in the Microsoft Word than we built an application based our algorithm respective to this answer sheet template. To prove our algorithms efficiency, we conducted multiple choice questions exam of Bachelor's and Master's degree program students at Information Technology Center, Sindh Agriculture University. After the exam, we scanned the answer sheets and stored it in our system, the sheets are considered as dataset for our case studies. Finally, we performed tests on our algorithm for Accuracy, Maturity and Performance attributes. We found that the proposed algorithm is 99.9% accurate and better in performance than a commercially available API. In maturity it was observed that MSIT and BSIT students are almost equal in maturity in filling their answer sheets correctly.

6. Future Work

The template designed for the proposed algorithm requires more professional design. The binaries of the algorithm are only for windows operating system, the binaries may also be created for other platforms so, variety of users may take benefit of the proposed implementation in their institutions. The algorithm requires further testing with more answer sheets and accuracy may be improved up to 100%.

REFERENCES

[1] Bayar, Gokhan. "The use of hough transform to develop an intelligent grading system for the multiple choice exam papers." Karaelmas Fen ve Mühendislik Dergisi 6, no. 1 (2016): 100-104.

[2] Bieniecki, Wojciech, Szymon Grabowski, and Wojciech Rozenberg. "Image preprocessing for improving ocr accuracy". In 2007 international conference on perspective technologies and methods in MEMS design, pp. 75-80. IEEE, 2007.

[3] Canny, John. "A computational approach to edge detection." IEEE Transactions on pattern analysis and machine intelligence 6 (1986): 679-698.

[4] China, Rodrigo Teiske, Francisco de Assis Zampirolli, Rogério Perino de Oliveira Neves, and José Artur Quilici-Gonzalez. "An application for automatic multiple-choice test grading on android." Revista Brasileira de Iniciação Científica 3, no. 2 (2016): 4-25.

[5] Chinnasarn, Krisana, and Yuttapong Rangsanseri. "Image-processing-oriented optical mark reader." In Applications of digital image processing XXII, vol. 3808, pp. 702-708. SPIE, 1999.

[6] Chouvatut, Varin, and Supachaya Prathan. "The flexible and adaptive X-mark detection for the simple answer sheets." In 2014 International Computer Science and Engineering Conference (ICSEC), pp. 433-439. IEEE, 2014.

[7] De Assis Zampirolli, Francisco, Valério Batista, and José Artur Quilici-Gonzalez. "An automatic generator and corrector of multiple choice tests with random answer keys." In 2016 IEEE Frontiers in Education Conference (FIE), pp. 1-8. IEEE, 2016.

[8] De Assis Zampirolli, Francisco, José Artur Quilici Gonzalez, and Rogério Perino de Oliveira Neves. "Automatic correction of multiple-choice tests using digital cameras and image processing." Universidade Federal do ABC (2010).

[9] Deng, Hui, Feng Wang, and Bo Liang. "A low-cost OMR solution for educational applications." In 2008 IEEE international symposium on parallel and distributed processing with applications, pp. 967-970. IEEE, 2008.

[10] De Oliveira, Filipe Tiago. "Optimized Methodology Using Multi-Choice Question Tests on Paper From question authoring to grade publishing." In 2018 3rd International Conference of the Portuguese Society for Engineering Education (CISPEE), pp. 1-6. IEEE, 2018.

[11] Fisteus, Jesus Arias, Abelardo Pardo, and Norberto Fernández García. "Grading multiple choice exams with low-cost and portable computer-vision techniques." Journal of Science Education and Technology 22 (2013): 560-571.

[12] Gaikwad, S. "Image processing based OMR sheet scanning." International Journal of Advanced Research in Electronics and Communication Engineering 4, no. 3 (2015): 519-522.

[13] Harraj, Abdeslam El, and Naoufal Raissouni. "OCR accuracy improvement on document images through a novel pre-processing approach." arXiv preprint arXiv:1509.03456 (2015).

[14] Hussmann, Stephan, and Peter Weiping Deng. "A high-speed optical mark reader hardware implementation at low cost using programmable logic." Real-Time Imaging 11, no. 1 (2005): 19-30.

[15] Kulkarni, Purva Anand. "Classification of Social Media Data for Suicidal Ideation". University of Maryland, Baltimore County, 2017.

[16] Matungka, Rittavee, Yuan F. Zheng, and Robert L. Ewing. "Image registration using adaptive polar transform." IEEE transactions on image processing 18, no. 10 (2009): 2340-2354.

[17] Filho, Alexandre Azevedo, Ethan V. Munson, and Cheng Thao. "Improving version-aware word documents." In Proceedings of the 2017 ACM Symposium on Document Engineering, pp. 129-132. 2017.

[18] Nerkar, B. "Optical Markup Recognition for Exam System." International Journal of Emerging Technology and Advanced engineering 5, no. 2 (2015).

[19] Nguyen, Anh Tu, and Yu Zhu. "Hand detecting and positioning based on depth image of Kinect sensor." International Journal of Information and Electronics Engineering 4, no. 3 (2014): 176.

[20] Sarwar, Abida Luhrani, and Zeeshan Ahmed Humair Nawaz. "Analysis Of Session Initiation Protocol With VoIP In Multimedia Conferencing System." International Journal 10, no. 3 (2021).

[21] Nguyen, Tien Dzung, Quyet Hoang Manh, Phuong Bui Minh, Long Nguyen Thanh, and Thang Manh Hoang. "Efficient and reliable camera based multiple-choice test grading system." In The 2011 International Conference on Advanced Technologies for Communications (ATC 2011), pp. 268-271. IEEE, 2011.

[22] Chung, Bryan WC, and Bryan WC Chung. "Getting Started with Processing and OpenCV." Pro Processing for Images and Computer Vision with OpenCV: Solutions for Media Artists and Creative Coders (2017): 1-37.

[23] Otsu, Nobuyuki. "A threshold selection method from gray-level histograms." IEEE transactions on systems, man, and cybernetics 9, no. 1 (1979): 62-66.

[24] Parul, H. Monga, and Mnupreet Kaur. "A novel optical mark recognition technique based on biogeography based optimization." International Journal of Information Technology and Knowledge Management 5, no. 2 (2012): 331-333.

[25] Sarwar, Abida Luhrani, and Zeeshan Ahmed Humair Nawaz. "Analysis Of Session Initiation Protocol With VoIP In Multimedia Conferencing System." International Journal 10, no. 3 (2021).

[26] Wijesekara, J. P. D., and P. G. T. P. Gunawardhana. "A Review on Mining Software Engineering Data for Software Defect Prediction." Information Technology Research Unit: 9.

[27] Van Rossum, Guido. "Python Programming Language." In USENIX annual technical conference, vol. 41, no. 1, pp. 1-36. 2007.

[28] Reddy, B. Srinivasa, and Biswanath N. Chatterji. "An FFT-based technique for translation, rotation, and scale-invariant image registration." IEEE transactions on image processing 5, no. 8 (1996): 1266-1271

[29] Saba, Tanzila, Amjad Rehman, Abdullah Al-Dhelaan, and Mznah Al-Rodhaan. "Evaluation of current documents image denoising techniques: a comparative study." Applied Artificial Intelligence 28, no. 9 (2014): 879-887.

[30] Khani, Muhammad Adnan Kaim, Abdullah Ayub Khan, Allah Bachayo Brohi, and Zaffar Ahmed Shaikh. "Designing Mobile Learning Smart Education System Architecture for Big Data Management Using Fog Computing Technology." International Journal of Imaging and Sensing Technologies and Applications (IJISTA) 1, no. 1 (2022): 1-23

[31] Saengtongsrikamon, Chatree, Phayung Meesad, and Sunantha Sodsee. "Scanner-based optical mark recognition." Information Technology Journal 5, no. 1 (2009): 69-73.

[32] Spadaccini, Andrea, and Vanni Rizzo. "A multiple-choice test recognition system based on the gamera framework." arXiv preprint arXiv:1105.3834 (2011)

[33] Mollah, Ayatullah Faruk, Nabamita Majumder, Subhadip Basu, and Mita Nasipuri. "Design of an optical character recognition system for camera-based handheld devices." arXiv preprint arXiv:1109.3317 (2011).

[34] Supic, Marko, Karla Brkic, Tomislav Hrkac, Željka Mihajlović, and Zoran Kalafatić. "Automatic recognition of handwritten corrections for multiple-choice exam answer sheets." In 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1136-1141. IEEE, 2014

[35] Rask, Jonas Kjær, Frederik Palludan Madsen, Nick Battle, Hugo Daniel Macedo, and Peter Gorm Larsen. "Visual studio code vdm support." In Proceedings of the 18th International Overture Workshop, pp. 35-49. 2021.