An OMR Checker For Mobile-based Images

A B. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Udayraj Deshmukh
(150101021)

under the guidance of

Prof. Pradip K. Das

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**An OMR Checker For Mobile-based Images**" is a bonafide work of **Udayraj Deshmukh** (**Roll No. 150101021**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Prof. Pradip K. Das**

Professor,

Department of Computer Science & Engineering,

Indian Institute of Technology Guwahati, Assam.

April, 2019

Contents

List of Figures

Chapter 1

Abstract

Optical Mark Recognition (OMR) is one of the most widely used automated data collection mechanism. Information to be inputted is marked using a pen or pencil in predefined fields on a printed paper. It is used at large scale in multiple choice examinations, surveys, lotteries, postal codes, etc. Traditionally a dedicated OMR scanning machine is used to detect the marked fields, which is quite expensive as well as requires skilled technician to configure it for correct functioning. This project aims to eliminate the need of a scanner, and speed up the OMR checking process by allowing for multiple devices to scan OMR sheets simultaneously in an organized manner. The system developed is aimed to be low cost, agile, robust and easy to use. Various image processing techniques namely normalization, adaptive thresholding, template matching and perspective transform are used. The implementation is done using openCV in Python.

Chapter 2

Introduction

OMR technology finds its major application in evaluating objective type examination papers and surveys. Many traditional OMR devices work with a dedicated scanner device that shines a beam of light onto the paper. The contrasting reflectivity at predetermined positions on a page is then used to detect the marked areas. The first mark sensing scanner was the IBM 805 Test Scoring machine; it read marks by sensing the electrical conductivity of graphite pencil lead using pairs of wire brushes that scanned the page. Early technologies developed for this task use dedicated Infra-Red OMR Machines, which although being very expensive are used mainly because of their high processing speed and accuracy. Such Infra-Red machines have high specifications - they work only for special thickness and color of OMR sheets. Also sheets with a crush or fold are rejected and have to be evaluated manually.

Further improvements in OMR technology were observed to aim mainly at reducing cost of the process. Mainly by making it feasible to use traditional scanner, and dividing the task into two steps - i) Scanning the OMR sheets, and ii) Identifying marked bubbles on the sheets. The detection of bubbles was done using mainly Image processing and computer vision techniques instead of IR sensors and expensive dedicated scanner.

The speed of the overall process can be considerably improved whilst maintaining ac-

curacy. Since with increasing resolutions of mobile cameras, it is possible to obtain images with comparable quality to that of scanners. This project proposes a mobile camera based system. In today's world smartphones with decent quality cameras are very widely used. So there will be no extra cost associated with using this smartphone-based solution. And by using a collaborated system, the speed of OMR checking will be directly proportional to number of mobile devices used. Above discussion motivates the problem statement of this project.

2.1 Problem Statement

An OMR checking system that can handle images taken from mobile devices at reasonably varying perspective angles and distances, with robustness against varying print quality, ambient light conditions, etc. The aim being to eliminate the need of expensive scanners and still improve speed of the process by allowing for taking images parallely from many devices.

Chapter 3

Review of Prior Works

3.1 Recent works

Recent works on optical mark recognition using image processing techniques include software along with specified scanner for specific design of Document or Form for OMR purpose. Hussman S. proposed a low cost and high speed system using Field Programmable Gate Array (FPGA), but it had constraints on the input of the forms [SH03].

An approach[Abb09] by A. A. Abbas reads both the ROI image of base paper with correctly marked answers and input paper images using the scanner, then both images are binarised and inverted. Next, the small objects are eliminated and the test paper image is rotated to align with the base paper image. Subsequently, two pre-processed binary images are multiplied and only the correct answers will appear in the resulted image.

[KA13] aims at replacing the heavy and expensive present day dedicated OMR machines by an ordinary scanner as a solution. It uses a histogram based technique to obtain an adaptive threshold per OMR sheet. Also they used square boxes as markers for correcting minute rotation in the scans.

In the approach taken by [GK13], An OMR sheet with a bounding box and corner points is used. The corner points from the scanned image are found. Then the bubbles in the sheet are checked whether they are marked by counting the number of black pixels

inside the bubbles.

In [S.15], The image is converted to binary image and then an ROI algorithm extracts the region for each question. The coordinates of marked bubbles are determined from the cropped region and used for deciding which option is marked. This process is followed for each question going downwards until end the of paper.

However, there were some drawbacks of above systems. If an expensive, high speed scanner is not available. Using ordinary scanner slows down the speed drastically for large scale checking. Using multiple mobile devices can speed it up again without additional costs.

With that fact in view, some existing works have enabled checking OMR sheets from a mobile device[PSGR15]. But there are constraints on the OMR layout in them, such as having a thick bounding box surrounding all the bubbles. Also the following possibilities are not addressed:

1. Low contrast in print : The per-page fixed threshold based process for bubble detection fails on bad quality OMR sheets often found in xeroxed sheets. This is due to low contrast between marked bubbles and their colored background.

2. Non linear distortion : If the sheet is not completely flat during scanning, there is an uneven skewing of positions of the bubbles causing wrong detections.

3.2 Proposed contribution

- Robustness on images of xeroxed sheets (low print contrast).
- Flexibility on OMR design e.g. no bounding box, independence from bubble shape and size, etc.
- Use of circular corner markers which would be invariant of rotations.
- Handling simple cases of non linear distortions

Chapter 4

Methodology

The system will take image of an OMR sheet as input, process it to read the marked bubbles and generate a score sheet as an output.

4.1 Input Description

Input Images:

There are mainly two types of OMR sheets in consideration -

1. Standard sheets: 120 gsm thick, A4, color printed.

2. Xeroxed sheets: 80 gsm thick, A4, black and white.

OMR Layout:

The layout of the OMR is shown in Fig 4.2. Note that the proposed algorithm can also handle OMR images directly from a scanner as well.

Fig. 4.1 Sample input images taken from a mobile camera

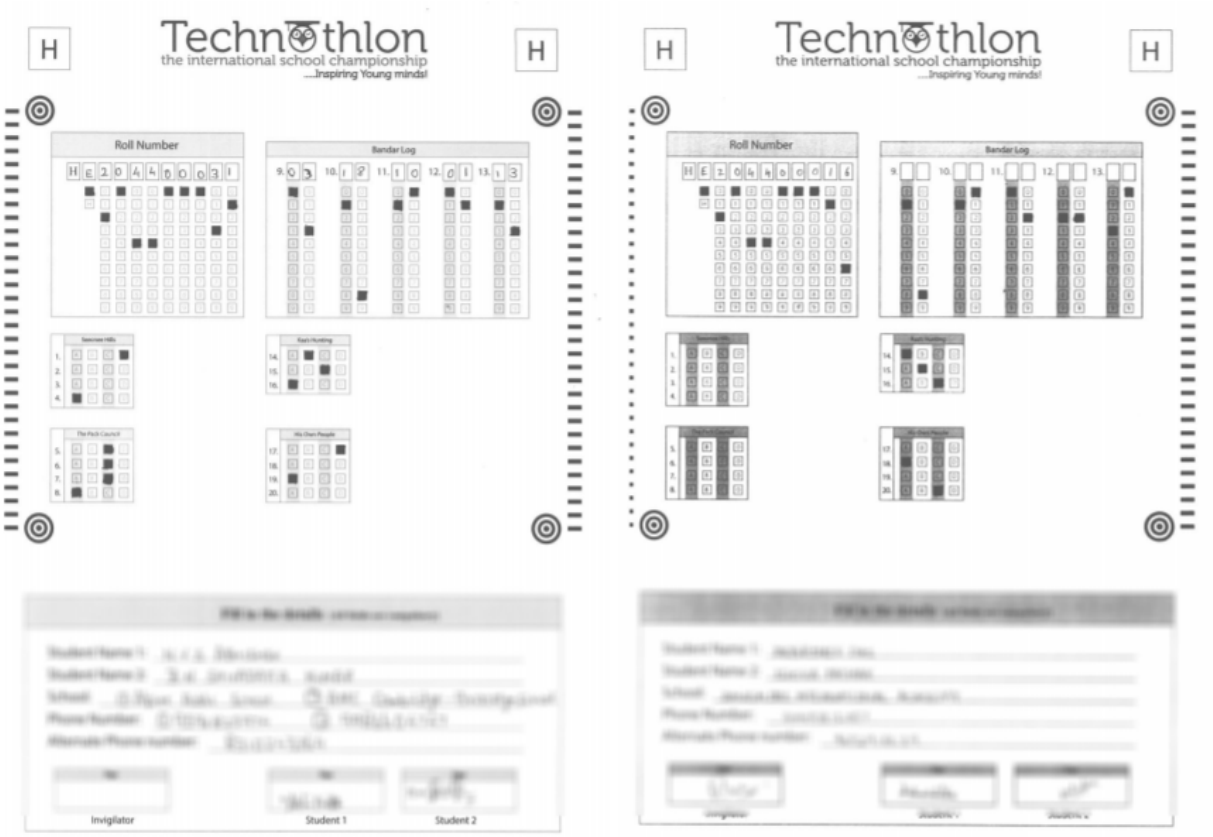

Fig. 4.2 Sample OMR scans: Left- Printed OMR, Right - Xeroxed OMR

4.2 Common techniques used

4.2.1 Normalization and its effects

A min-max normalization is applied to the image to cover the full range of intensity values i.e. (0-255). It is given by the formula:

$$I_{norm} = \frac{(I - I_{\min}) * 255}{I_{\max} - I_{\min}}$$

where I_{norm} is the normalized image, I is the input image, and $I_{\min}$ and $I_{\max}$ are the minimum and maximum intensity values in the image.

This increases the contrast between dark and light areas in the image, making it easier for obtaining a threshold (Fig 4.3).

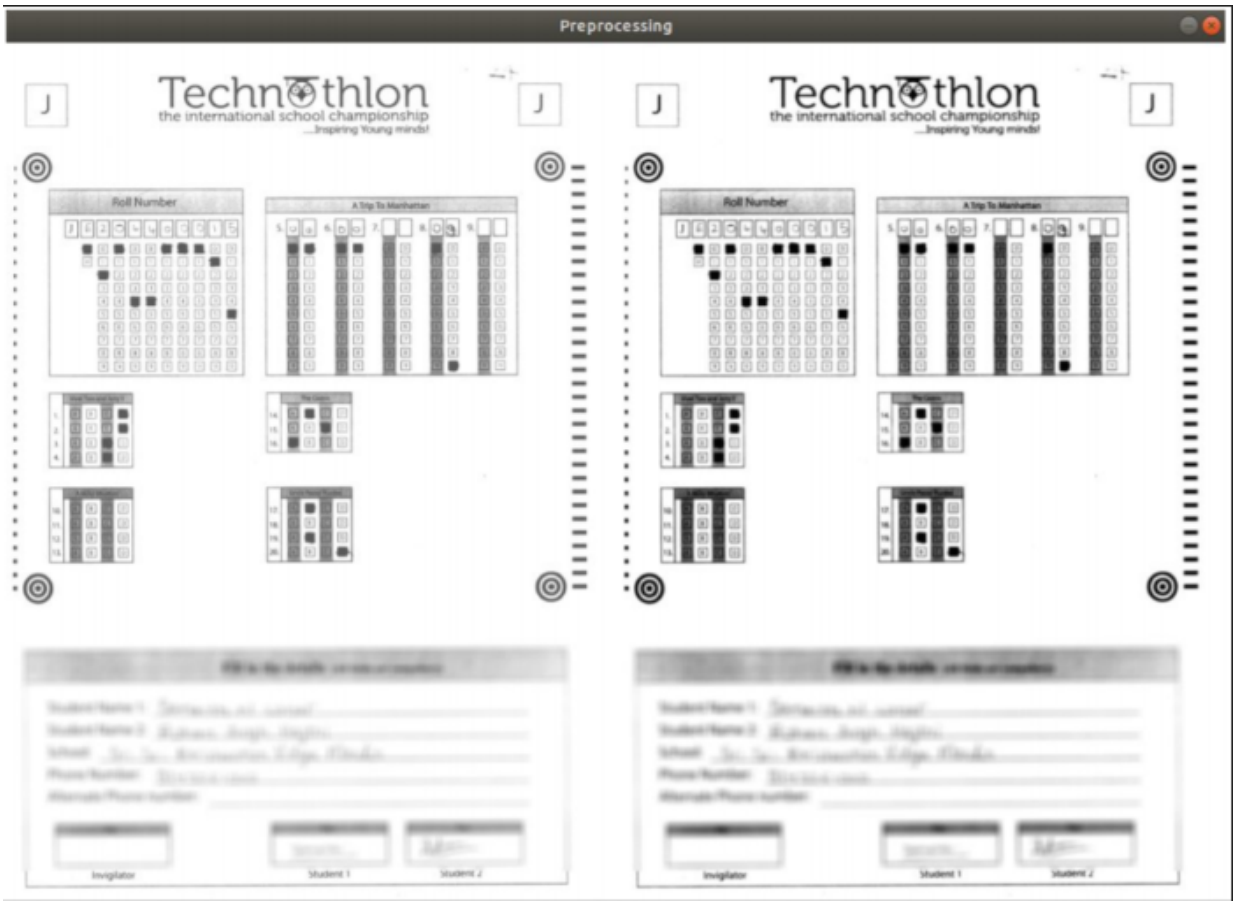

Fig. 4.3 Effect of normalization

4.2.2 Template matching

In template matching, the template image simply slides (one pixel at a time) over the input image (as in 2D convolution), and calculates a normalized cross correlation [Lewis95] at each of its positions. The point of maxima in this cross correlation output is the position in input image that is most similar to template image. Marker occurrences in the image have to preserve the orientation of the original marker i.e. rotated marker would not be detected in the image. It is upto the user which kind of marker to use. We have used the following marker image (Fig 4.4).

Fig. 4.4 Marker circle

By using a template with circular symmetry, rotated images will not affect the detection. We are using concentric circles to make it more distinguishable from rest of the OMR sheet.

Also, using four markers for relative positioning was found to be more effective than using a thick bounding box as there was no problem of randomness in detecting outer or inner border of the bounding box.

4.2.3 Perspective Warping

Using the obtained four corner points, we obtain a transformation matrix M corresponding to the perspective transform. This matrix is applied on the image to obtain a warped image. This transformation can translate, rotate and scale the image such that the four given points of the input image will become the corner points of the output image. Thus we obtain a region of interest (ROI) image from the four points.

4.3 Main Algorithm

An overview of this system is shown in the following block diagram (Fig 4.5): The sys-

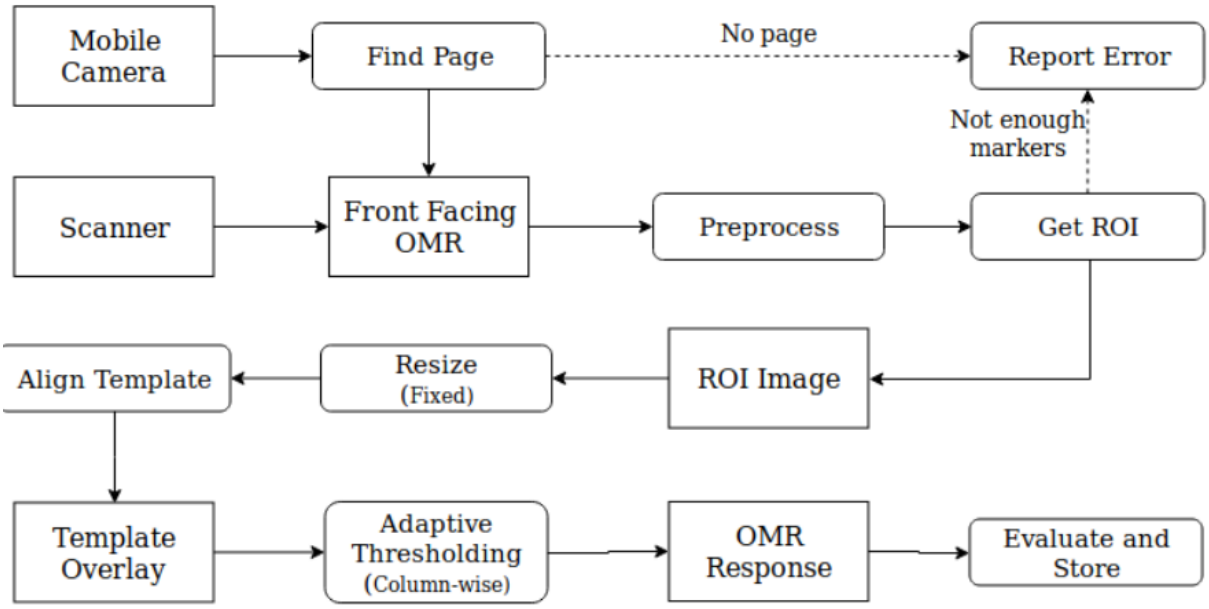

Fig. 4.5 Block diagram of OMR system

tem proposed has two components - mobile application, workstation application. Mobile application is only intended for verifying presence of OMR with four markers. It does not need to evaluate the OMR. So it only runs until get ROI stage in above block diagram. After verification, the OMR image is saved in a user-defined directory on the mobile phone with incremental numbers as filenames. These images can then be uploaded in bulk to the workstation directory using any file transfer method or file synchronization applications such as Google Drive. Working of the application is shown in Fig 4.6.

On workstation side the markers are again identified and rest of the algorithm follows on the image. The reason for designing the system this way is because of criticality of images to be collected at the centralized workstation. It is required for two reasons - manual correction for ambiguous OMRs, and for review requests from student to see if OMR was scanned correctly. Distributing the process of taking images over multiple devices is sufficient to speed up the entire process significantly. In fact, the application is created only

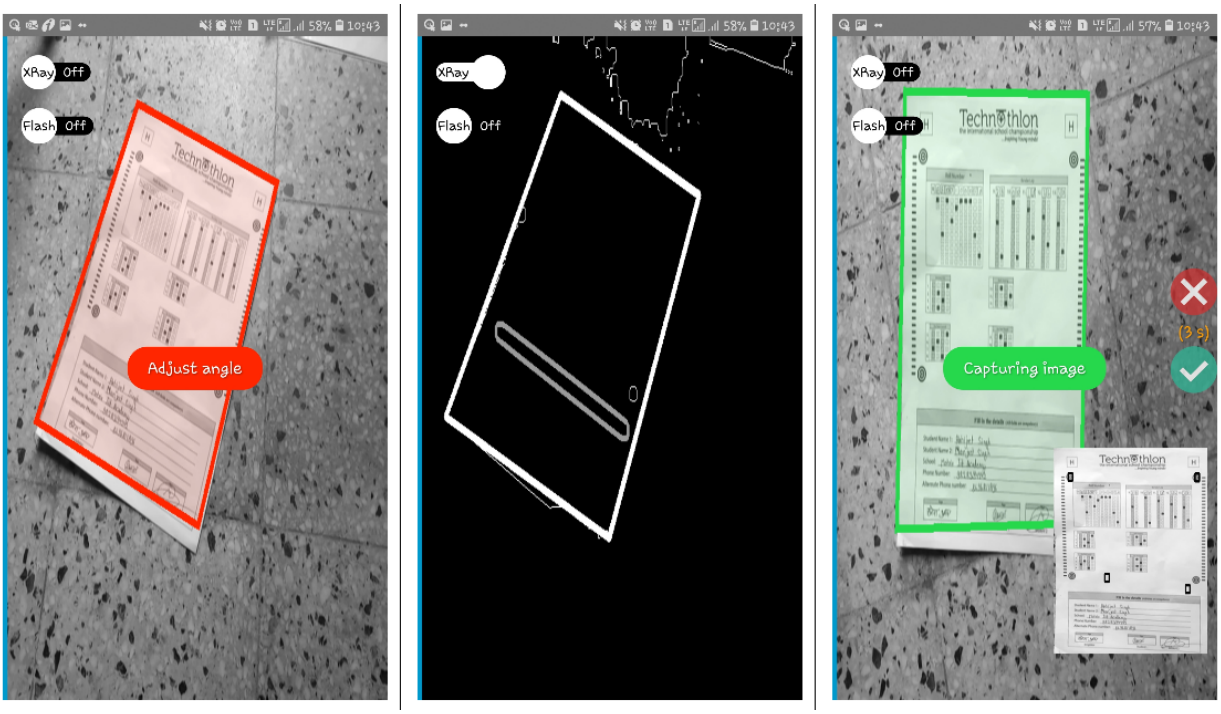

Fig. 4.6 Various screens of the mobile application.

for reducing manual work. One may as well manually put their images into the correct directory at workstation to evaluate them.

4.3.1 Find Page

This method has following block diagram (Fig 4.7): To extract corner points of the OMR

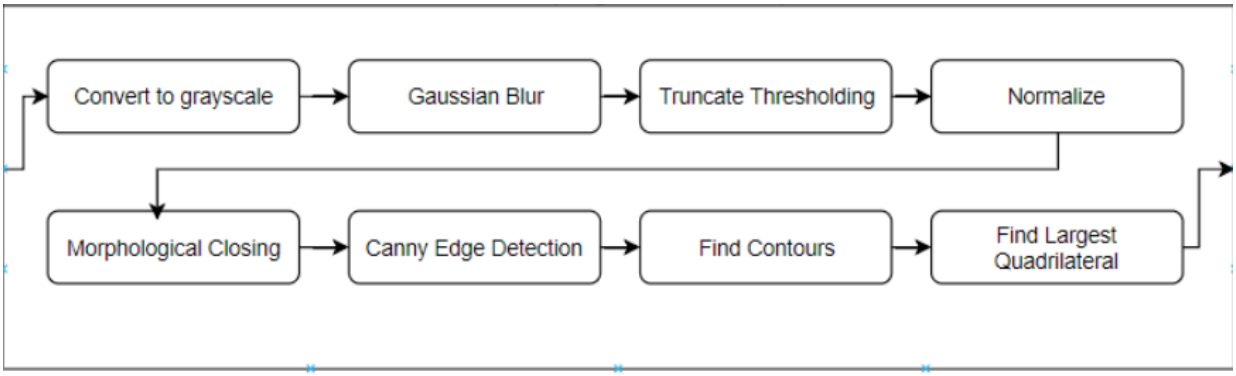

Fig. 4.7 Block diagram of the Find Page method

sheet, we follow these steps sequentially:

Algorithm 1 Find Page

1. Convert the image to gray scale image I_{gray}.
2. Apply gaussian blur with kernel size K_{blur} on I_{gray} to get I_{blur}.
3. Apply truncated thresholding with $T_{truncPage}$ on I_{blur} to get $I_{truncPage}$. Here every pixel value above $T_{truncPage}$ is replaced by 255. The page now becomes brighter and more uniform.
4. Apply min-max normalization on $I_{truncPage}$ to get $I_{normPage}$.
5. Apply morphological closing with kernel size K_{close} on $I_{normPage}$ to get I_{closed}. It is useful in closing small holes inside the page(See Fig 4.8).
6. Apply canny edge detection with parameter: C_{low} and C_{high} on I_{closed} to get I_{canny}.
7. Find the contours in I_{canny}.
8. To find the largest quadrilateral
   (a) First each of the contours is converted to a polygon using ramerdouglaspeucker polygon approximation algorithm[[HDKP73]]
   (b) Then out of the polygon which have 4 vertices, one with the largest area is chosen.
   (c) Warp perspective transform is applied on I_{gray} using these four vertices to get I_{front}

Parameter Values: $K_{blur} = 3$, $T_{truncPage} = 200$, $K_{close} = 10$, $C_{low} = 55$, $C_{high} = 185$

An example of above method is shown for reference:

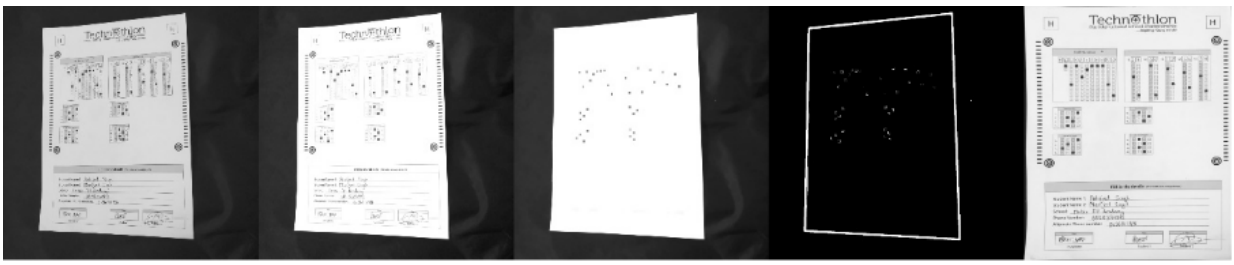

Fig. 4.8 Outputs at steps 1, 4, 5, 7, 8 respectively

4.3.2 Preprocessing

1. Convert input I_{front} to grayscale I_{gray} if it is not in grayscale.
2. Resize I_{gray} to a fixed size of W_1 x H_1 to get $I_{resized}$.
3. Apply gaussian blur with kernel size K_{blur} on $I_{resized}$ to get I_{blur}.
4. Normalize $I_{resized}$ to get I_{norm}.

Parameter Values: W_1 x H_1 = 666 X 820. These resize dimensions were tested sufficient for rest of the steps.

4.3.3 Get Region of Interest(ROI)

If the image and marker template are blurred, we may find multiple very nearby points having the highest correlation which is not desired. Here we exploit the fact that we have to obtain four distant matches - the corner circles on the OMR sheet. Thus we first divide the image into quadrants to save computations as well as eliminate the possibility of detecting very nearby points as matches.

1. Divide I_{norm} into four quadrants.

2. In each of the quadrants apply template matching with marker image I_{marker} as the template.

3. If the maxima for any of the four quadrants is less than a threshold T_{marker}, return with error message: Not enough markers

4. Apply warp transform on I_{norm} using the four maxima locations to get I_{ROI}.

Parameter values: $T_{marker} = 0.3$

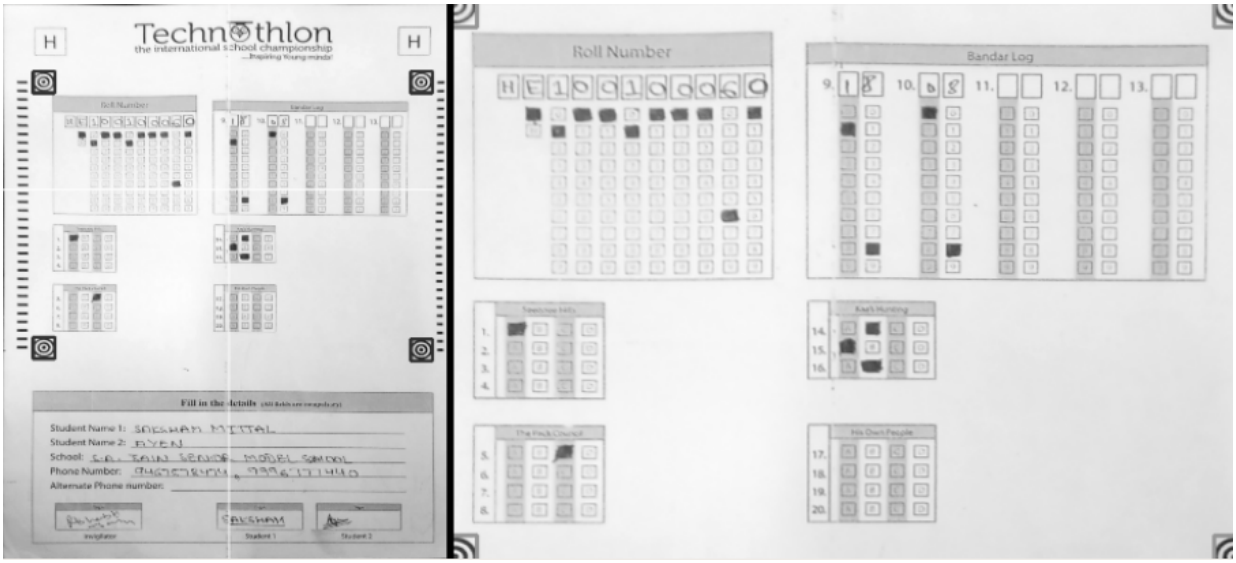

Fig. 4.9 Left: Template matching - Bounding boxes drawn at the points of maxima in each quadrant. Right: Warp perspective applied on I_{norm} to get I_{ROI}

4.3.4 Resize

Resize I_{ROI} to a fixed size of W_2xH_2 to get I_{resROI}. Parameter Values: W_2xH_2 = 1500 x 1846 (Obtained from template layout file)

Note: Since we are taking $W_2 > W_1$ and $H_2 > H_1$, there will be an upsampling of the image, hence no need of applying gaussian blur again.

4.3.5 Align Template

The problem: A case where alignment is shifted due to perspective effect is shown in Fig 4.10. There are three possible reasons of wrong alignment of the template overlay -

1. Photos taken from too wide angle or high distance cause template to shift due to perspective effect.

2. The paper is not perfectly flat when capturing.

3. Deviation in detecting marker location due to blur effect.

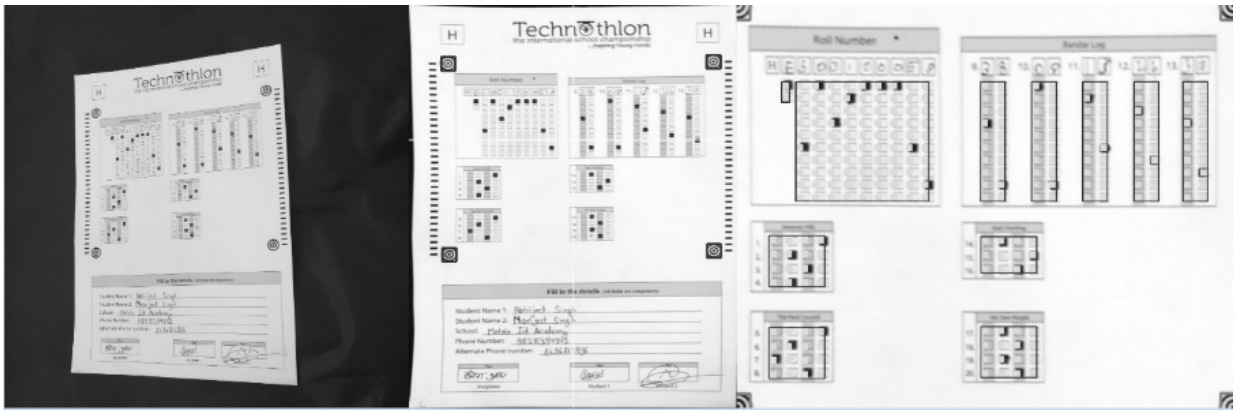

Fig. 4.10 Perspective Effect: Columns on the left are misaligned

The template layout file from input has information about blocks of bubbles that form a rectangle.

To solve this problem, we first extract the columns in the image. Then try to shift individual blocks from the template to align with these columns. We define edge for such rectangular block as vertical column of width Tedge and height the same as the block. left edge would be the edge at leftmost end of the block, similarly right edge would be the edge the rightmost end. We define shift for the block which corresponds to the alignment correction value. The alignment algorithm is given in Algorithm 2.

Parameter Values: C_{lim} = 5.0, C_{tile} = 8 x 8, G_{low} = 0.7, $T_{truncAlign}$ = 220 $K_{morphOpen}$ = 3 x 10, $N_{morphOpen}$ = 3, $K_{morphErode}$ = 5 x 5, $N_{morphErode}$ = 2, $T_{binaryMorph}$ = 60, T_{edge} =

Algorithm 2 Align Template

1. Apply CLAHE with clip limit C_{lim} and tile size C_{tile} on I_{resROI} to get I_{clahe}. It increases contrast in the image locally, so the columns become darker.

2. Apply Gamma correction on I_{clahe} with gamma $G_{low} < 1$ to get I_{gamma}. It further darken the columns

3. Apply truncated thresholding with threshold $T_{truncAlign}$ and max value also as $T_{truncAlign}$ (instead of default 255) to get $I_{truncAlign}$

4. Normalize $I_{truncAlign}$ to get $I_{normAlign}$

5. Apply morphological opening on $I_{normAlign}$ with a vertical kernel of size $K_{morphOpen}$ and iterations $N_{morphOpen}$ to get $I_{morphOpen}$

6. Repeat step 3 and 4 on $I_{morphOpen}$ and invert the image to get $I_{morphInv}$.

7. Apply binary threshold on $I_{morphInv}$ with threshold $T_{binaryMorph}$ to get $I_{binaryMorph}$.

8. Apply morphological erosion on $I_{binaryMorph}$ with a kernel of size $K_{morphErode}$ and iterations $N_{morphErode}$ to get $I_{morphErode}$.

9. Now, on $I_{morphErode}$ run the following alignment algorithm -

```
for block B in template:
shift = 0
for step = 0 to N_maxSteps:
    L = mean value of left edge of B
    R = mean value of right edge of B
    if((L > 100 and R > 100) or
     (L <= 100 and R <= 100)):
        break
    else if(L > 100):
        shift = shift - 1
    else:
        shift = shift + 1
B.shift = shift
```

5, $N_{maxSteps} = 15$ One important parameter here is $K_{morphOpen}$ which is an asymmetrical kernel. It helps connecting vertically close elements in the image such as the bubbles in a column.

Template layout after alignment correction (Fig 4.11):

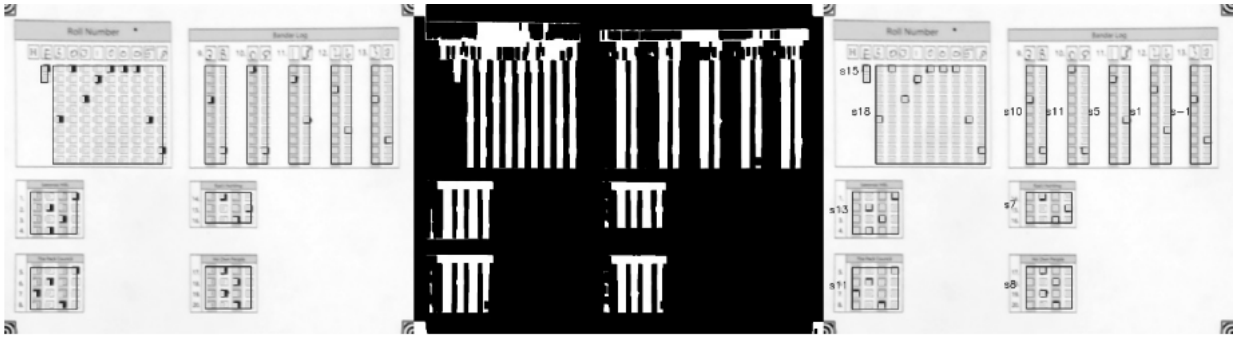

Fig. 4.11 Alignment correction

4.3.6 Adaptive Thresholding

Now, the mean intensities at each of the aligned bubble locations are calculated and stored in an array in increasing order. Their histogram is plotted to observe a pattern -

1. A common case of Color Printed OMR, 120 gsm (Fig 4.12):

   Very large gap between white and black areas. Negligible gray area points.

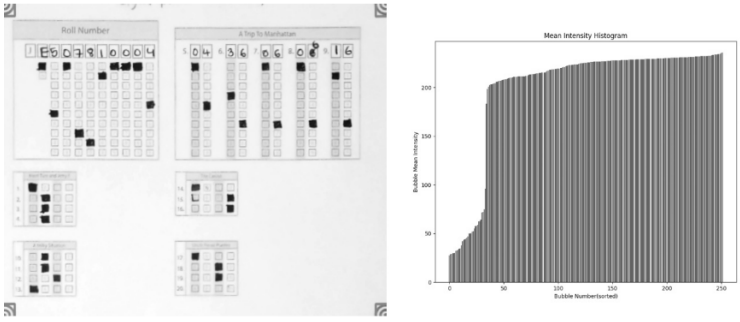

Fig. 4.12 Histogram for Color Printed OMR, 120 gsm

2. A Common case of Xeroxed OMR, 80 gsm (Fig 4.13):

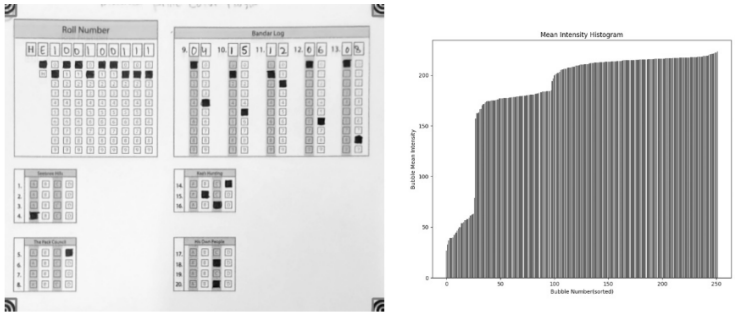

Fig. 4.13 Histogram for Xeroxed OMR, 80 gsm

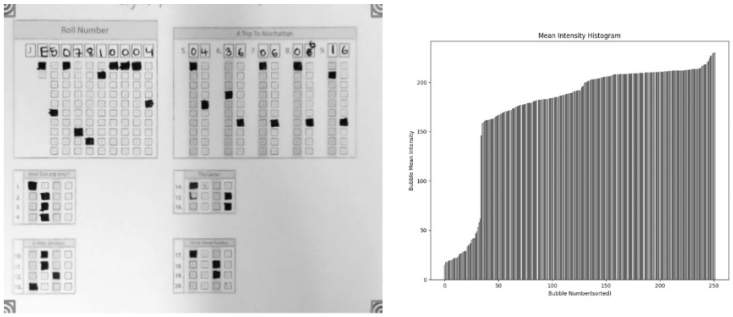

Fig. 4.14 Histogram for Shadowed Xeroxed OMR, 80 gsm

3. Effect of narrow shadow on a Xeroxed OMR, 80 gsm (Fig 4.14):

4. Effect of wide shadow: Xeroxed OMR, 80 gsm (Fig 4.15):

   Gap between black and gray parts is reduced. And a smoother gradient is observed between gray and white area

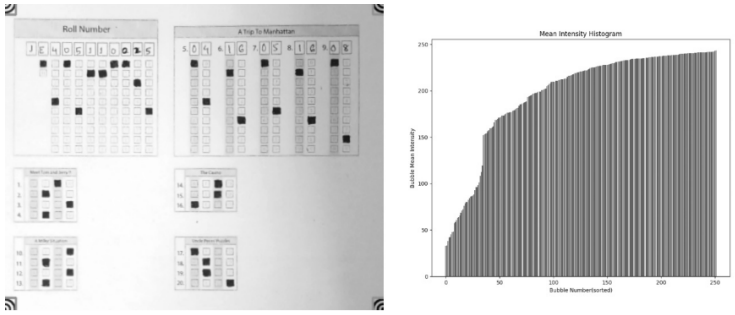

Fig. 4.15 Histogram for Wide-shadowed Xeroxed OMR, 80 gsm

5. Large number of bubbles darkened: Color Printed OMR, 120 gsm (Fig 4.16):

From above examples, it was observed that there is always a large gap between black

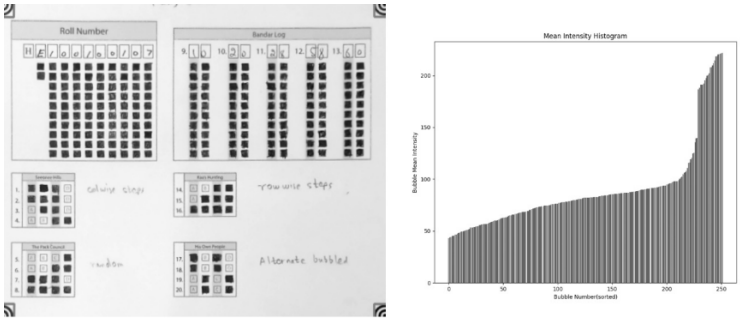

Fig. 4.16 Histogram for large number of marked bubbles

and gray areas, while the gap between gray and white areas may not be present. The value at mid point of the largest jump between black and gray area is taken as threshold T_{global}. A similar approach was used in [Atal13], but the algorithm used was limited only to common cases mentioned above, where the boundary between black and white area was large and less varying with respect to min and max values in the histogram. As observed in cases 3-5, the point of boundary between these areas varies significantly depending on lighting conditions. To handle such cases, we present a different approach to finding T_{global} as shown in Algorithm 3:

Using algorithm 3, we are get appropriate adaptive threshold for cases 1-4. Now, there still remains abundant cases where only T_{global} does not suffice, especially when the effect of bad quality print and shadow is combined.

In Fig 4.18, The gray area is reaching mean intensity as low as 50, which is usually the value for black areas.

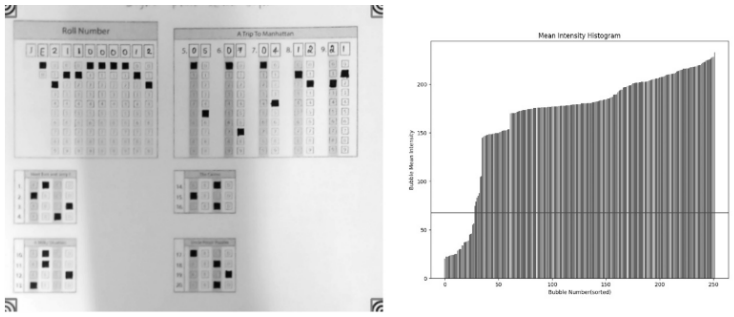

Fig. 4.17 T_{global} : horizontal line in the histogram.

Algorithm 3 getGlobalThreshold (I_{means})

```
Sort I_means in increasing order
max_1 = J_min
thr_1 = 255
for i = 1 to length( I_means ) - 1:
    jump = I_means[i+1] - I_means[i-1]
    if(jump > max_1):
                max_1 = jump
                thr_1 =  I_means[i-1] + jump/2
max_2 = J_min
thr_2 = 255
for i = 1 to length( I_means ) - 1:
    jump = I_means[i+1] - I_means[i-1]
    thr = I_means[i-1] + jump/2
    if(jump > max_2 and abs(thr) > J_delta):
                max_2 = jump
                thr_2 = thr
T_global = min(thr_1, thr_2)
Parameter values:
    J_min = 20, J_delta = 40
```

Fig. 4.18 T_{global} is not low enough to detect the bubbles on the bottom left

The shadow affects overall distribution of intensities, but if we look at column-wise distribution instead, there is a clear and consistent boundary between black and gray/white area (Fig 4.19) So we can apply a similar approach to find a threshold T_{local} for each column. Here we additionally need to verify if any of the bubble is marked i.e. if there is any outlier present which can be easily found. If there is none, we can use T_{global} for discriminating the bubbles (Algorithm 4).

Algorithm 4 getLocalThreshold (C_{means}, T_{global})

```
Sort C_means in increasing order
if ( no outliers in C_means ):
        T_local = T_global
else:
    max_1 = J_min
    T_local = 255
    for i = 1 to length(C_means) - 1:
        jump = C_means[i+1] - C_means[i-1]
        if(jump > max_1):
                max_1 = jump
                T_local =  C_means[i-1] + jump/2

Parameter values:
    J_min = 20
```

Column-wise histograms for the bubbles on bottom left for the erroneous case are shown in Fig 4.19.

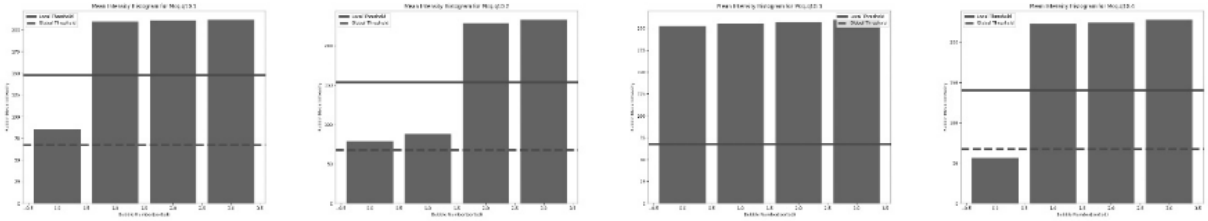

Fig. 4.19 Column-wise histograms of an MCQ type question (T_{global} denoted by Dashed line, T_{local} by Solid line).

Applying local threshold method solves the issue as seen in Fig 4.20.

So the overall adaptive thresholding algorithm will be as follows:

Fig. 4.20 Reading response Using column-wise threshold: T_{local}

1. I_{means} = Array of mean intensity values at each bubble position
2. Sort I_{means}
3. T_{global} = getGlobalThreshold(I_{means}).
4. For each column C in Template layout:
   (a) T_{local} = getLocalThreshold(C_{means}, T_{global})
   (b) Use T_{local} as threshold for detecting bubbles.

In generating OMR Response, the bubble is considered as marked if its mean intensity is below T_{local}, otherwise unmarked. If multiple marked bubbles are found, their corresponding values are concatenated and forwarded. This concatenation is necessary for multi-digit integer type questions. This is followed for each question and the response is aggregated into the OMR Response.

4.3.7 Evaluate and Store

The obtained response is evaluated against an answer key to obtain the final score. The score along with other details such as roll number are appended to a CSV file(Fig 4.21). For the answer key, the user can customize the marking scheme which also includes support for negative marking and section-wise marking schemes.The marking scheme along with answer key is stored in an easily readable format in a json file.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	roll	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18	q19	q20	score
2	JE413720006	C	C		C		2	22							B	B	D					4
3	JE413720007			D					3	6			D	B	D	B	B			C	C	2
4	JE413720003	B		D	C						C	B	C	C	B		B		C	C		11
5	JE413720005	A	B	B	C						B	D	A	C	C	A	B	B	D	A	B	14
6	JE413720009	A	C	D	C		8	39	20	8	12	B	A	D	C	D	C	B	D	D			2
7	JE413720002			C	C									B	B	C	B	B					6
8	JE413720010	C	D	D	C		8	39	2	9	14	B	B	B	B	C	C		D	D			4
9	JE413720001			D	C		2	96	3	18	8			D	B	C	A	B			A		1
10	JE413720017	A	D	C	B		12	11		18		D		A	B	D	A	B		C			-4
11	JE413720014	C	A	B	C		8	39	2	9	12	B	A	D	A	D	C	D	D	C			3
12	JE413720012	D	D	D	C		2	49	2	8	8	D	B	D	C	B	A	C	B	D	D	B	4

Fig. 4.21 CSV output file containing registered responses

Chapter 5

Results

The algorithm was implemented in OpenCV 4.0.0 and Python 3.6.7. The system was tested on a workstation having 8GB RAM, Intel i7 5th Gen processor. The resolution of phone camera used was 4.7MP. The robustness of the system is considerable at this configuration and does not increase drastically on using higher resolutions.

The testing was done on OMR sheets put under various conditions as shown in figures below - Various Bubbling (Fig 5.1), Rotation (Fig 5.2), Perspective (Fig 5.3), Shadow (Fig 5.4), Slight Temperament with marker (Fig 5.5), Folding from sides: (Fig 5.6), Folding from middle: (Fig 5.7).

Above corner cases were all handled correctly by the proposed algorithm.

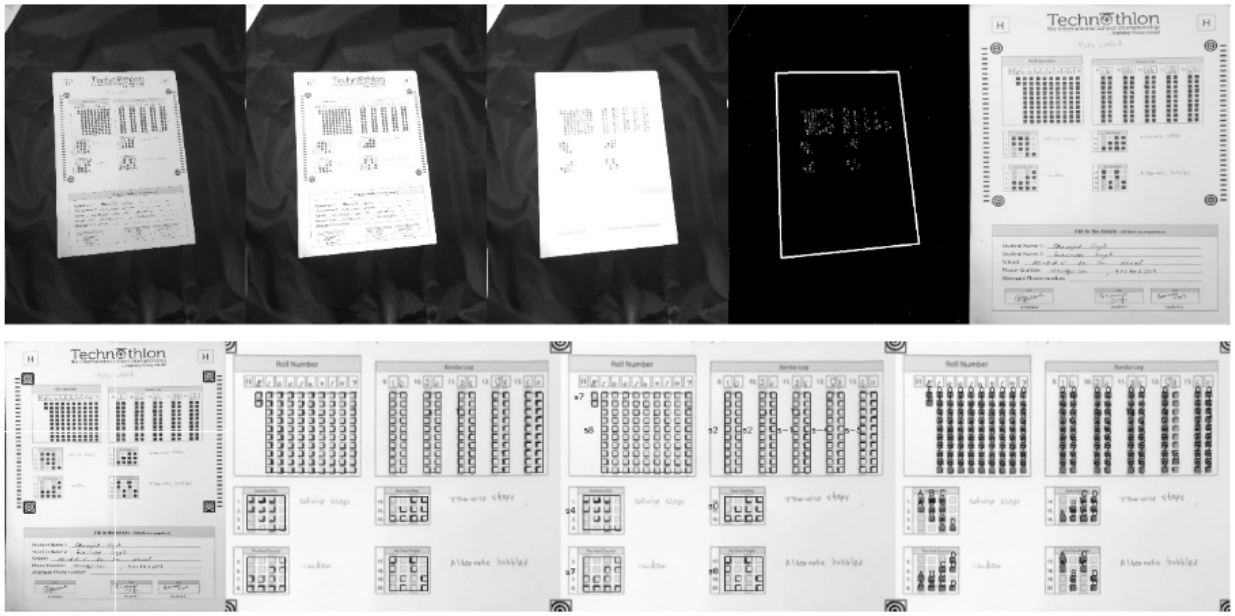

Fig. 5.1 Various Bubbling

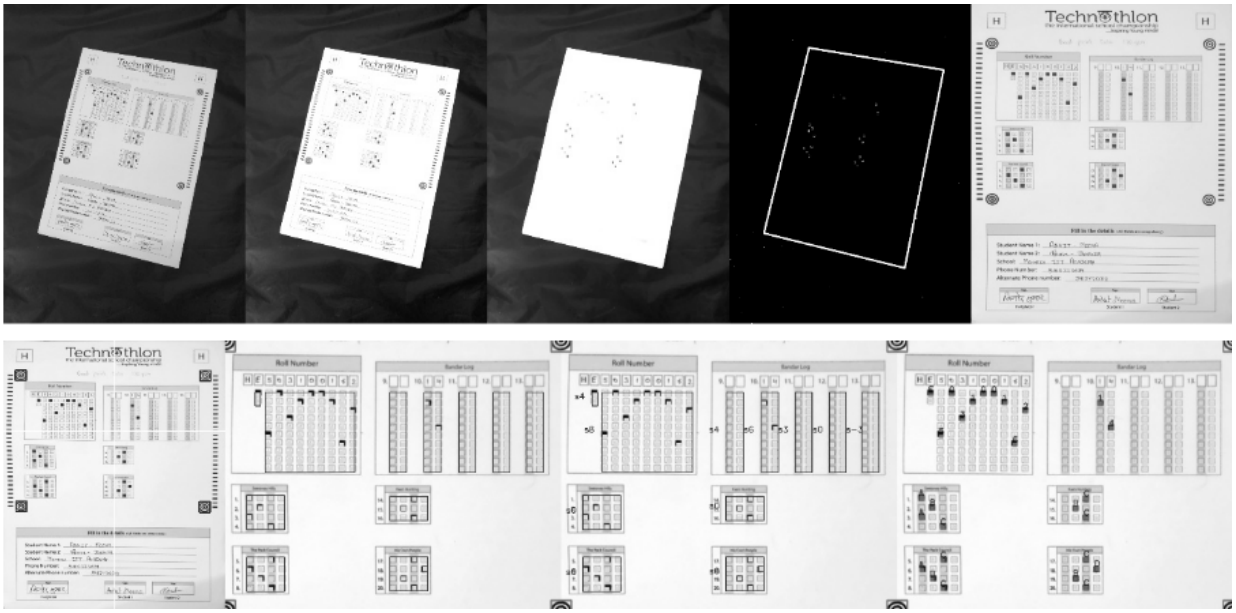

Fig. 5.2 Rotation

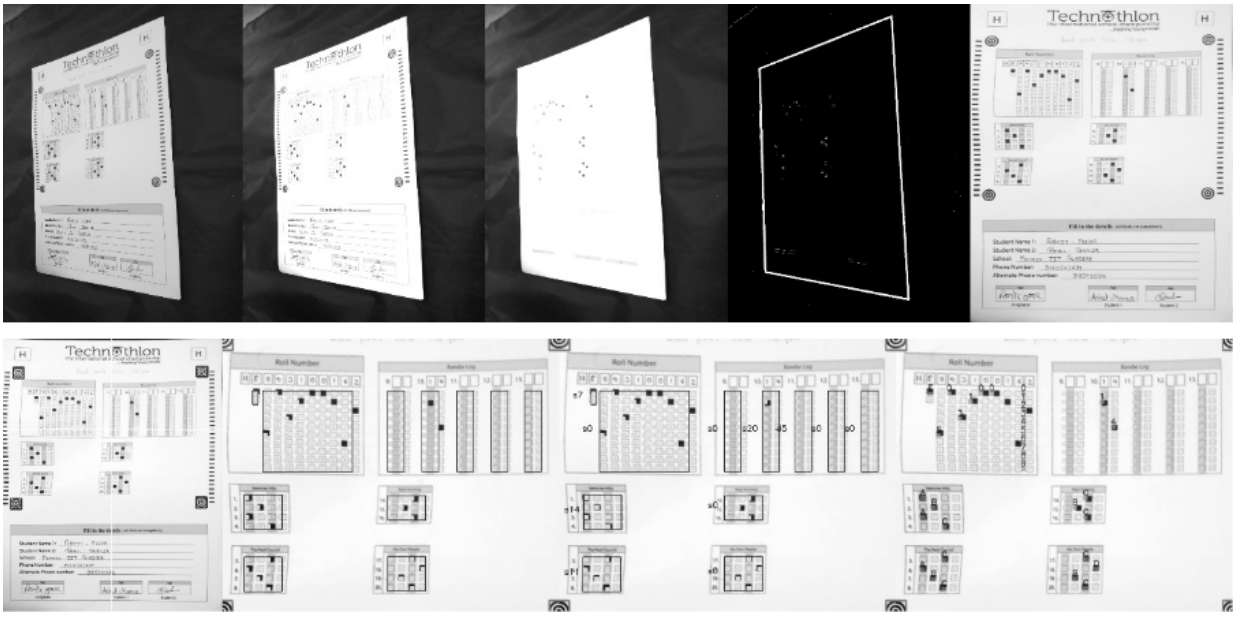

Fig. 5.3 Perspective

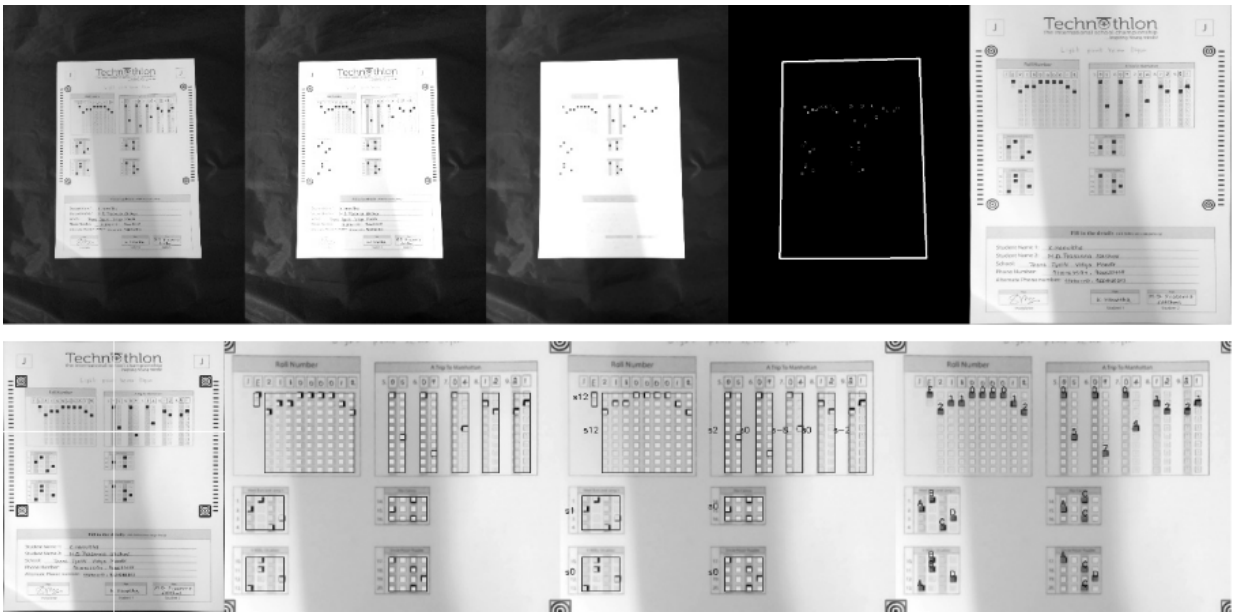

Fig. 5.4 Shadow

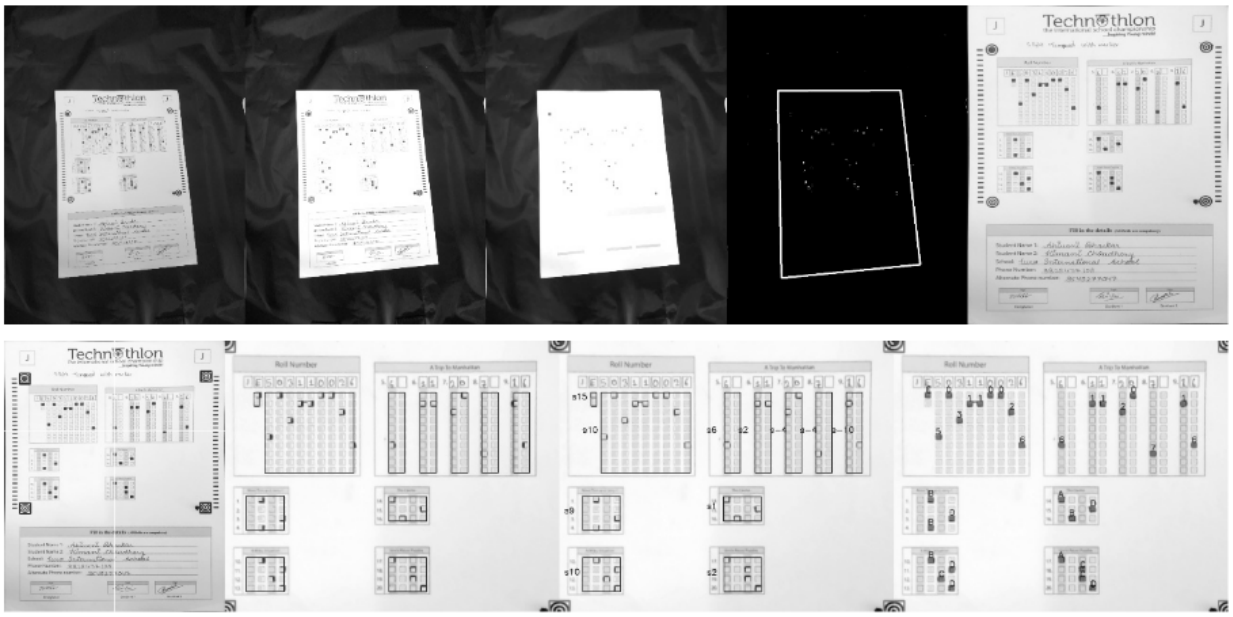

Fig. 5.5 Temperament

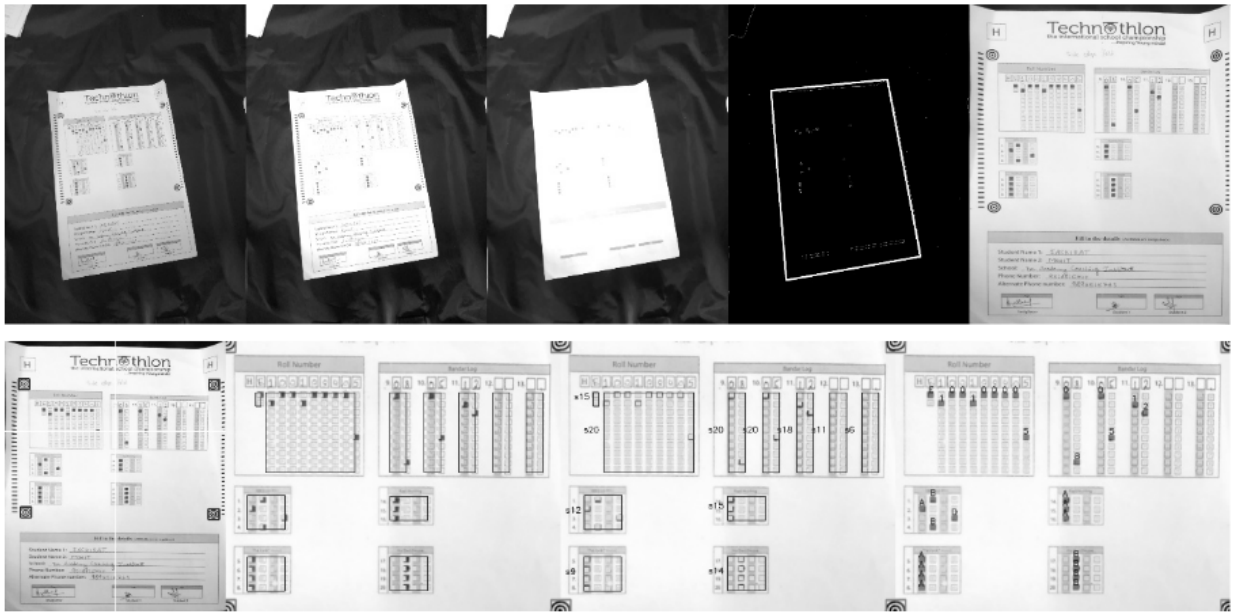

Fig. 5.6 Side Folding

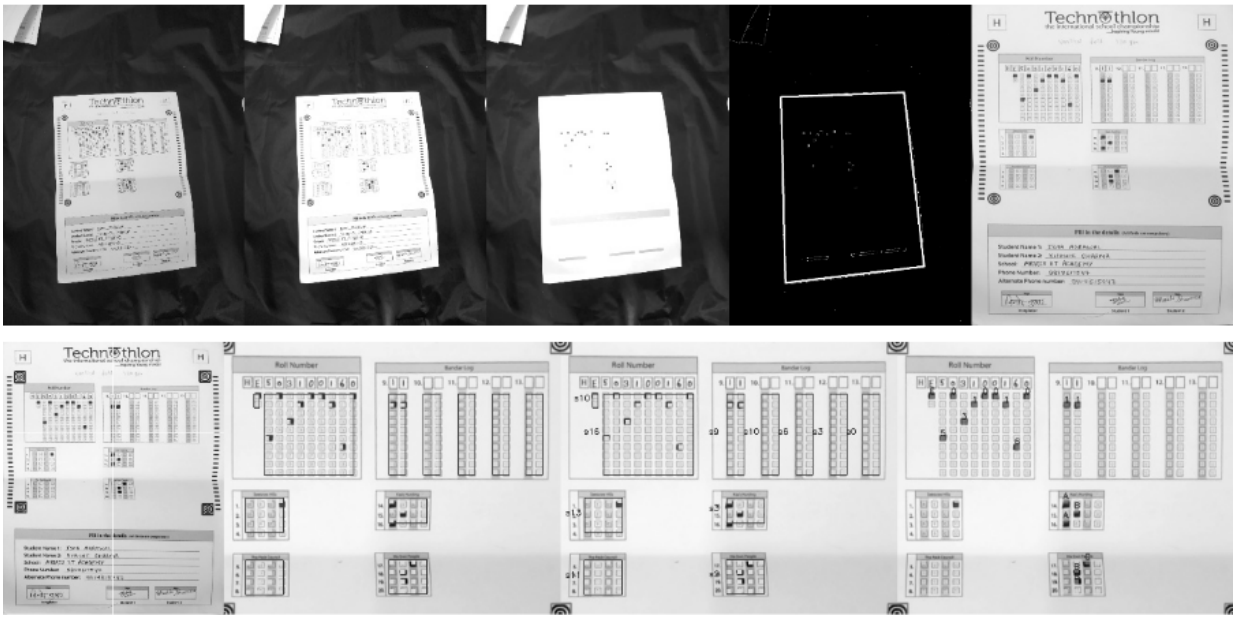

Fig. 5.7 Middle Folding

Chapter 6

Conclusion and Future Work

We have proposed a cost-effective and robust software to check OMR sheets captured from mobile cameras with high accuracy and speed. This system can readily replace current day OMR scanners to achieve the same objective. There is no limit on number of mobile cameras that can be used simultaneously in this system, thus with no added cost we can scale up the system limitlessly. The system also puts no constraints on quality of paper or color of printing. Experimental results clearly depict the robustness and correctness of the proposed system.

In future we aim to further speed up the workstation application using threading, GPU usage, etc. Also the mobile application has a great scope of improvement in speeding up scanning process. We are glad to have open sourced the application codes completely at https://github.com/Udayraj123/OMRChecker/ and welcome any kind of contribution to it.

References

[Abb09] A. A. Abbas. An automatic system to grade multiple choice question paper based exams. *Journal of Al-Anbar University for Pure Science*, 3, 2009.

[GK13] Hemant Ram Rana G. Krishna. Implementation of omr technology with the help of ordinary scanner. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3:714–719, 2013.

[HDKP73] DAVID H DOUGLAS and THOMAS K PEUCKER. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10:112–122, 10 1973.

[KA13] A. Arora K. Atal. Cost effective optical mark reader. *International Journal of Computer Science & Artificial Intelligence*, 24, 2013.

[PSGR15] R. Patel, S. Sanghavi, D. Gupta, and M. S. Raval. Checkit - a low cost mobile omr system. In *TENCON 2015 - 2015 IEEE Region 10 Conference*, pages 1–5, Nov 2015.

[S.15] Gaikwad S. Image processing based omr sheet scanning. *International Journal of Advanced in Electronics and Communication Engineering*, 4, 2015.

[SH03] C.Fung M. Albrecht Stephen Hussmann, Leona Chan. Low cost and high speed optical mark reader based on intelligent line camera. *Proceedings of the SPIE AeroSense 2003,optical pattern recognition*, 5106:200–08, 2003.