Optical Mark Recognition for Computer Answer Sheet

Yonas Adiel Wiguna[1]

[1] School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40135, Indonesia
E-mail: yonazadielwiguna@gmail.com

Abstract. Nowadays, we use computer-based answer sheet for exam with many participants that can be scored by computer automatically, fast, and objective. However, the devices is still hard and expensive to get. With image processing technology, We aim to get the same result even though the photo of computer answer sheet is captured using low resolution smartphone. The image will be loaded and converted to binary image. Then, the image will be wrapped due to uncertain perspective of the image and mapped into matrix of boolean, indicating whether there is a circle that is marked or not. From the matrix, we may check process further by retrieving the name of exam participant, subject, and all the answers. In addition, we develop a web application for manual correction and ease of usage.

1. Introduction

Examination is one way to grade students and check whether the students have already understand the material that have been taught or not. However, some schools have many students and manual grading is hard to do. Manual grading may also be subjective and have human error. One way to overcome these problems is by automated grading using computer.

However, computer grading only limited on multiple choice questions. Here, students are asked and usually have 3 to 5 choices. In simple examination, one of the choices are the correct answer. In some examination, there maybe none or multiple correct choices. Some examinations also have score reduction for incorrect answers.

In Indonesia, examination is one of the methods to determine whether the student will graduate or not. The examination is called “Ujian Nasional” (National Examination) and held simultaneously across the whole country. The examination is held for students in elementary school, junior high school, and high school. The questions have up to 40 multiple choice questions.

Until 2015, national examination is still held using special computer answer sheet that can be scanned. Up until now, some schools use this examination for doing regular examination because of its ease. However, the scanner price range from 10 millions to 200 millions. Not all schools in Indonesia can afford the scanner.

However, using image processing, computer answer sheet should be able to be scanned and processed to get students grade. Using general scanner, the result will have high quality and easier to be

processed. But the scan will be tiring and take a long time. The idea of this paper is to use smart phone that is cheap and many people have. The image captured will have lower quality than general scanner.

2. Processing The Image

2.1. Python Imaging Library

At first, we try to use Python Imaging Library. This library can load any PNG or JPEG images into a list of pixels. Each pixels represent the RGB value of the images. Each pixel is denoted as a tuple of three values (blue, green, red) with maximum value of 255 and minimum value of 0.

Using this value, we may freely process, modify, and analyze our image. We have already tried this method. However, this approach has very bad performance in time. For example, we use breadth first search (BFS) algorithm for the blob detector. Even though the complexity is still linear to the number of pixels, it still took a long time to process.

2.2. Loading The Image

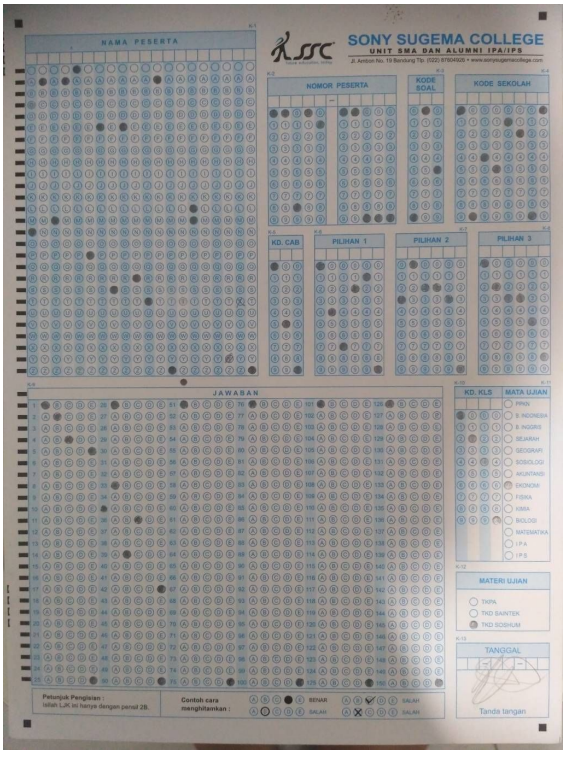

Figure 1. Example of the answer sheet captured by smartphone camera. The shadow is the shadow of the person that take the image.

The image is loaded as a grayscale image. We are aware that one of the best approaches is by loading only its blue channel, due to the fact that the answer sheet usually use blue color for the mark outline,

instructions, etc. We expect the blue channel of answer outline has nearest value to paper color (which is white), compared to the red or green channel.

However in the cases where the answer sheet is not using blue for the outline, we can't extract only the outline colour. For example, yellow is the mixture of red and green. We still need to determine how many percentage of red and green contributes to the final grayscale value. In general cases, we can use the standard RGB to grayscale coefficient, which is 0.2989 for blue, 0.5870 for green, and 0.1140 for red [2].

Then, we resize the image. We find that working with large resolution image will take very long time, so we decided that working with at least 1000 pixels width image is good enough.

2.3. Convert Image into Black and White

One of the challenges is to be compatible with every kind of brightness. Image taken by mobile devices may vary in brightness. Some use very high brightness, some use very low brightness. Simply find an integer (0 - 255) to be the threshold value will only work in some cases, so we need a better approach. One of the approaches is using Otsu's method [1].

In short, this method is exhaustively search for best threshold value. The evaluation is done by calculating the inner-class variance. OpenCV already support Otsu's method by adding `cv2.THRESH_OTSU` in the `threshold` function parameters.

However, the most challenging problem is the levels distribution. Image levels refers to how the pixels is distributed in the brightness -- is it mostly bright colors or dark colors, etc. Sometimes, the levels at one side of the answer sheet is different with the levels at the other side of the answer sheet. At other cases, the photo was taken with shadow on it, so the paper color is different on different side of the paper.

In order to solve this challenge, our approach is by using the function `adaptiveThreshold` provided by OpenCV itself with `ADAPTIVE_THRESH_GAUSSIAN_C` as adaptive method, block `size` of 11, `C` of 2, `THRESH_BINARY` as threshold type. We choose the block size according to the ratio of an answer mark size with paper size.

However, before we apply the adaptive threshold, we blur the image first. By blurring the image, we remove the noises of our image. On figure 2 we can see how our threshold works really well, even though the sample answer sheet has shadows on it.

In Indonesia, this kind of computer answer sheet usually marked using 2B pencil. 2B pencils are type of with some determined amount of black. However this kind of pencil reflects the light too well. If user takes a photo of computer answer sheet with light reflected on all the marks, the recognition will be failed. On the other hand, user needs to take the photo in quite bright place, so we get good brightness and contrast.

In order to overcome this problem, the user can create a shadow on top of the paper. However, this can create a bad shadow that is not covering all the answer sheets, like in figure 1. That's why we need adaptive threshold for creating good binary image even if the sheet has shadows on it.

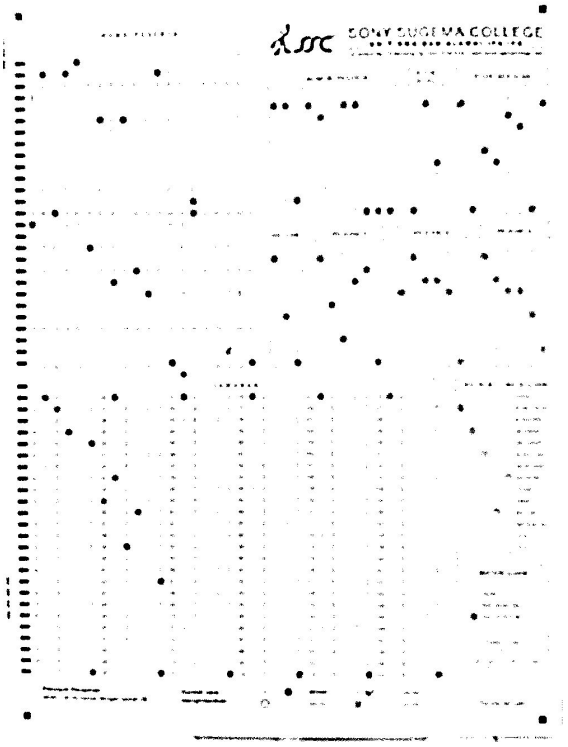

Figure 2. Answer sheet after our threshold

2.4. Wrapping The Image

Other challenge in processing answer sheet image that was taken by mobile phone is the perspective. We don't know the orientation of the mobile device, where the paper margin, etc. Additionally, the paper may distorted (folded, wavy, etc). Our solution is by wrapping the image before it is processed further.

The interesting feature of a computer answer sheet is that it always has 4 squares on its corners. Some answer sheet only have three squares, but three squares should be enough, because the fourth squares can be approximated. Usually, this square is also has the same area and proportions as answer mark, so we can use one procedure to find both the corner square and answer mark.

First of all, we try to find the key points. We use the OpenCV's `SimpleBlobDetector`. We use circularity filter range 0.75 to 1 and convexity filter range from 0.85 to 1. We also use area filter range with reasonable minimum and maximum area - it depends on how large the answer circle. Figure 3 shows how all keypoints successfully detected.

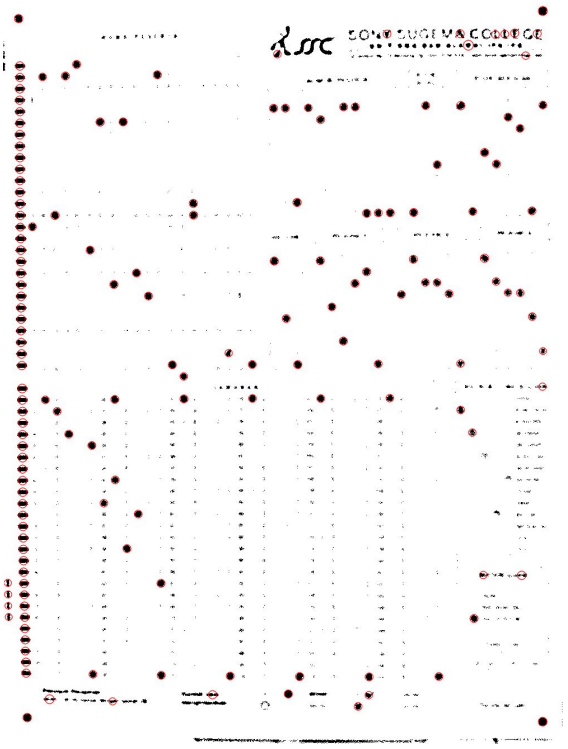

Figure 3. Detecting all keypoints

After finding all the keypoints, we pick four of those which is the nearest to the corners. Even if some of the answer sheet paper wasn't taken on the image, as long as the four corners are still present, we can wrap the image. If there is margin between edge of paper and photo taken, this margin won't be detected as key points, because we already filter our key points by circularity, convexity, and area.

To find these four square corners, we imagine if there is a 45° line sweeping from bottom left to top right. The first point touched by this line will detected as bottom left point and the last point touched by this line will detected as top right point. The same goes for the bottom right and top left, but this time we use 45° line sweep from bottom right to top left.

Lastly, we create the equation of the original corners and the intended corners. The original corners (X) are represented as matrix 4 row times 2 columns, each row representing the index of row and index of the column. The intended corners (Y) are represented as 4 times 2 matrix too, but this time the value will be the (0, 0) to the (height, column). We try to solve the equation and get 4 x 4 matrix A.

$$AX = Y \quad (1)$$

Lastly, we remap all the pixel to the new one. There will be some rounding, but because the image is binary, then the rounding will be 0 or 255 only. Figure 4 shows the result of our perspective transform.

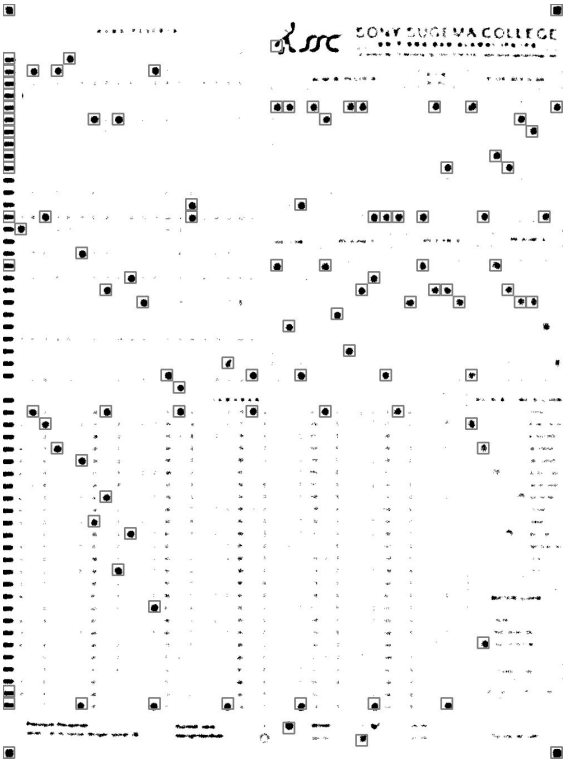

Figure 4. Answer sheet after we wrap the corners

2.5. Key Points Mapping

We already have key points, now we have to extract it into answer key points matrix. Then, from matrix, we may determine whether a question is answered A, B, C, D, or E. The same goes for the participant's name, number, etc.

First, we create an empty matrix. We map the image into some number of rows and columns. As we can see in figure 5, good answer sheets usually have good alignment of the rows and columns. We can try to guess the number of rows and columns by finding the minimum error of blob location.

When determining the number of rows, we just considering the y coordinate of all blobs. We try to find the R so that the E is minimum in equation (2). N stands for the number of blobs, $Height$ is the height of the image, and y is list of row indexes of blobs.

$$E = \sum_{i=1}^{N} \left| (y_i \,\%\, (Height / R)) - (Height / 2R) \right| \quad (2)$$

Because the expected number of rows is quite small, then we can find the R with bruteforce method. The same method goes for the algorithm for finding the number of columns. Then for every keypoint, we determine which cell it belongs to. Because we know the number of rows and columns, we can determine the boundary for each cell. We find where the center of blobs are lied.

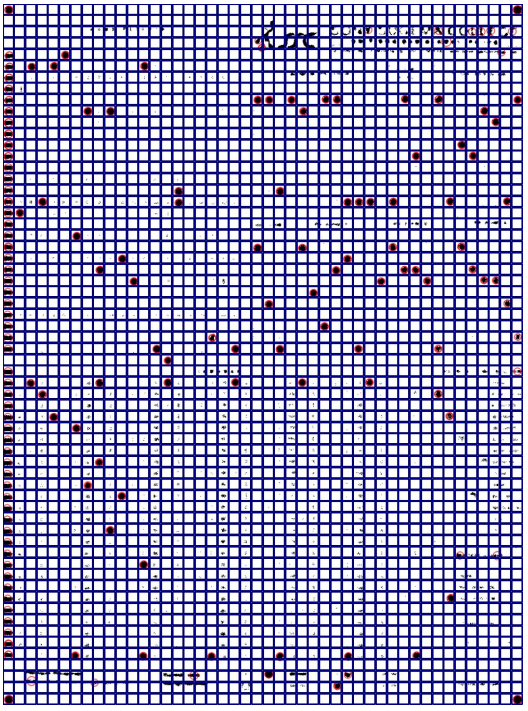

Figure 5. Mapped answer sheet into 62 rows and 46 columns

After this step, we have matrix that each cell contains 1 if the corresponding circle in answer sheet is filled, 0 otherwise.

2.6. Grading The Answer

The last step is processing the boolean matrix. We determine that a question is answered if and only if exactly one of the options is marked. No more, no less. For doing that, we create two functions, called `get_multiple_choice_horizontal` and `get_multiple_choice_vertical`. The parameters are the boolean matrix, first cell of multiple choice offset, and number of choices. If one of the choices are filled, then we know which answer is chosen.

These functions applies for other types of choice such as participant name, participant number, subject code, etc. However, the first coordinates of all choices need to be specified. By knowing the boolean matrix, we still can't determine the exact layout of the answer sheets.

2.7. Web Application

Because the goal is for easy usage, we deploy the optical mark recognition as python web backend server and using react.js as frontend. The backend will give the recognition result and the image that has been marked. This way, if there is one or more answers that is nor detected, user may manually correcting the result of the recognition.

3. Conclusion

Optical Mark Recognition can be applied to computer-based answer sheet. This enables scoring a computer-based answer sheet using simple program and mobile phone. Scoring a computer-based answer sheet will no longer require special scanner.

4. References

[1] Nobuyuki Otsu, *A threshold selection method from gray-level histograms*, IEEE Transl. Sys., Man., 1979.

[2] Charles Poynton, *The magnitude of nonconstant luminance errors*, John Wiley and Sans, New York, 1996.