Automated Grading for Bubble-Sheet Exams

Gregory Nero, Wade Pines, Vlad Simion, Ian Smith, Haley Wiskoski
Rochester Institute of Technology
Rochester, NY
IPCV II : Group Scantron Project

***Abstract*—Grading a large number of exams by hand is tedious. By using image-processing techniques, the process of grading exams can be automated. Automating grading has the potential to save valuable amounts of time and minimize human-based grading errors that could result from fatigue and frustration. This automation would only require the administrator to digitally scan the students' responses, the blank bubble-sheet used for examination, and the answer key before importing those scanned files to perform the automated grading. After the grading is done, the administrator will then have access to a spreadsheet of meaningful results including the exam score for each student. *Section I* serves as an introduction to the task of automated grading. *Section II* goes over the methods used to solve this task. *Section III* is a presentation of the results. *Section IV* concludes with a discussion of the results and a few additional auxiliary comments.**

Key Words: BSS - blank scan sheet, SSS - student scan sheet, HCT - hough circle transform, OMR - optical mark recognition

I. Introduction

Automated grading though image processing is extremely important for large-scale student examination. According to college board, more than 2 million students in the class of 2018 took the SAT! [1]. This quantity of student responses is especially suited for automated grading because of how time-sensitive these responses are for anxious students applying to school. Having a reliable and efficient way to grade a large number of exams is essential in cases like this. Even on smaller classroom scales, this automation can still be very useful, especially for instructors with large class-sizes. This scheme of automated grading with bubble-sheets really only works when there is a "quantized" number of responses. For instance, performing automated grading of an English exam with open-ended worded responses would not work well. (Though, with the advent of automated character recognition and machine learning this may not be such a lofty goal!) This bubble-sheet format works especially well with multiple choice response exams, where each student response can be categorized with a binary "right or wrong" label. By streamlining student responses into this path, certain freedoms are forfeited (such as partial credit) but the trade-off is that the grading process is much easier and quicker for the professor, especially if the professor chooses to adopt an automated grading scheme. These are personal choices that are up to the instructor, however, so it would depend heavily on the pedagogical morals of each instructor.

Figure 1 demonstrates a blank bubble sheet. Certain design factors were taking into consideration for this sheet. The three circular configurations at the three corners can be used for image alignment. Each approach to this problem might require different bubble-sheet features specifically tailored to help that one unique implementation method, but the approach described here enjoys a design like the one shown in Figure 1. Having a designated bubble-sheet allows for the normalization of the bubble-detection/location process and makes answer comparison easy (like comparing differences in a drawing using tracing paper). Switching to a new blank sheet format would require some additional work to compensate for, but sticking with one is the easiest path. Having students fill in the bubbles in a clear, standardized manner (completely filled with a dark pencil, for instance) is of great importance. Essentially, keeping everything as standardized as possible is the best, and any deviation from that will make the job of the developer harder. However, it is up to the developer to expect and account for these abnormalities. Since the instructor will have direct access to all of the data visually at first, there is some relationship between the computer and teacher in this regard. The instructor should be aware of the automated results

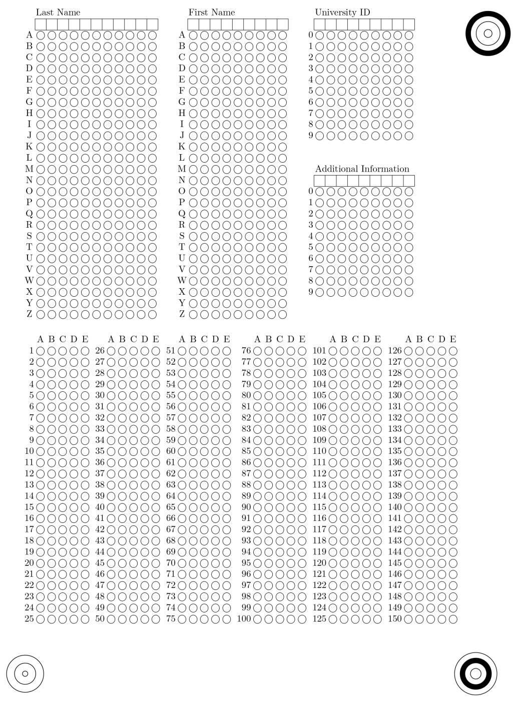

Fig. 1. A blank scan-sheet

and use them with respect. The grade of the student should not be left to chance at all, so in all situations of grade conflict, human intervention should be utilized.

1) Optical Mark Recognition: Optical mark recognition (OMR) is the process of capturing human-marked data from certain documents. In this facet, test answer sheets were the target of our data capture, governed by the size and placement of circles throughout the paper. In this case, Students mark their answers, or other personal information, by darkening circles marked on a pre-printed sheet. Afterwards the sheet is automatically graded by the program that was designed. Utilizing this design is beneficial to the user as the OMR process return data in an accurate and time efficient manner. Applying this to an academic setting, where professors need to grade tests in a accurate and timely fashion, OMR is the way to go.

2) The Hough Transform: The Hough transform is way to represent lines, curves, shapes, and other geometric features in an image in a parameter space that can be used for the detection of such objects. The generalized Hough transform is useful for detecting all sorts of object like lines and curves, but a more specific (and in this case, useful) subset of this detection scheme is the Hough Circle Transform (HCT). This is specifically designed for finding circles in an image, given that the image has been treated with a few pre-processing steps to reduce noise and detect edges. Some of the pre-processing done to the scan sheets is discussed in the **Methods** section. Figure 2 shows a picture that aims to explain the fundamental idea behind the HCT.

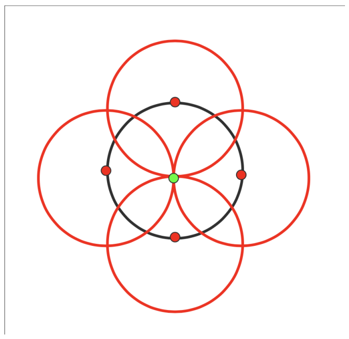

Fig. 2. The fundamental geometry behind how the HCT detects a perfect circle. By drawing circles of equal radii (red) along the perimeter of the circle that is trying to be detected (black) it can be seen that a max number of intersections occur at the center of the black circle, providing evidence that the origin of a circle of that specific radius lives at that location (green).

Since the bubble-sheet contains perfect circles like the one in Figure 2, this method works quite nicely. By drawing circles who's origins all reside on the perimeter of the circle in question, there will be an ideal geometric configuration for each circle of radius R where all of the circles of radius R that are being drawn will intersect at a single point: the origin of the circle that is under inspection. Assuming that the radius of the circle is not known to begin with, this requires the algorithm to loop through a whole range of radii to test. By drawing circles along each edge point in this manner using an ”r-theta” polar scheme and looping over a range of radii specified by the user, this peak point of intersection can be identified by looking for a peak in the response profile that tracks intersections. These intersections should be tracked cumulatively, using a process commonly referred to as ”voting.” This [2] paper does an excellent job of conceptualizing this process. Essentially, intersections can be tracked in a so-called ”accumulator matrix” which is a three dimensional matrix of the same planar dimension as the input source and a third dimension with length equal to the number of radii being tested from zero to R. In practice, a vector of images can be used to store these radial responses, but a three dimensional matrix would also work. By evaluating each edge point with this circular treatment and cumulatively tracking intersections, peak values can be found given a threshold, which would therefore indicate that a circle has been detected. Since the radial layer of the peak is knowable by iterative inspection and the location is known due to the one-to-one dimensional relationship between the source and the planar dimension of the accumulator matrix, the radius and image-location of each circle can be know. This detection is of fundamental importance for this approach. In practice, an OpenCV function was used, but the theory presented here should give some appreciation for this algorithm. It should be known that the HCT does not work very well with noise (unless it is appropriately filtered out) but in a curated case like this (scanning sheets) this is not much of a concern unless the scanner is bad.

II. Methods

To begin the process of scanning and grading, the test data must first be ingested and pre-processed. It necessary to first take in a blank scan sheet (which will be referred to as ”BSS” through the rest of this paper) for calibration purposes, the answer key scan sheet, and all student scan sheets (SSS) to be graded in the end.

The blank scan sheet is held in its own matrix, and a 2D matrix is created to hold however many images the user inputs. The blank scan sheet is then converted to grayscale for ease of processing and manipulation of data. Next, Hough Circle Transform, as described earlier in this paper, is applied to the BSS. The purpose of this is to initialize the circle locations for each and every bubble on the sheet. It is known that the BSS contains a total of 1450 answer bubbles, but the Hough Transform detects 1457 circles. Knowing the correct number of circles needed and their radii, the unnecessary circles may be removed. After the Hough Transform is applied to the BSS, each circle coordinate is stored in a vector of points. The radius of every circle is discounted since it is the same for every circle, and thus is redundant information.

One thing to point out, though, is that although we have all of the circle coordinates stored in a vector, the Hough Transform detected them randomly throughout the image. Thus they are not stored in any logical order in the vector, which is an issue that needs to be remedied. By using a simple sorting function, the circles are then ordered in an organized manner based solely on their x,y coordinates.

The next step in this process, is to then assign which circles belong to which column / section within the BSS. The sections are comprised of first name, last name, university ID, additional information, and six columns of answer bubbles. These are organized again by looping through the previous organized vector of coordinates, and sorting them into 10 separate vectors of points for each individual section of the BSS.

Once the points are organized, it is important to account for any unwanted rotation or scaling of the input student scan sheets. This is simply due to the fact that it is not predictable the orientation of which they will be scanned individually, and for calibration and accuracy purposes the SSS must align with the already processed BSS. To begin this portion of the algorithm, the SSS are converted to grayscale, again for simplicity in data manipulation further on. Fortunately, the BSS and SSS all contain three circular targets on the upper right, and lower two corners. The Hough Circle Transform is applied again to the BSS and SSS to detect these targets. Knowing the radii and locations of the targets on the BSS, it is possible to perform a perspective transformation on the SSS to align with the targets on the BSS. The SSS are then scaled and rotated with respect to the desired radii of the targets. Finally, the SSS and BSS will have the same dimensions!

The next step is to perform even more pre-processing on the SSS so as to enhance the bubbles for easier detection and comparison later on. First, inverse global thresholding is applied to binarize the SSS. The threshold value, T, is determined as 255 minus the standard deviation of the image. This mathematical value was chosen due to the overall better representation of details within the image after binarization. Next, the image is eroded to remove any stray marks on the SSS, and to thin the bubble outline thickness. Again, this is for easier comparison and detection later on when comparing to the BSS. Erosion works simply by running a kernel across the image to remove unwanted pixels. Figure 3 demonstrated the process of erosion.

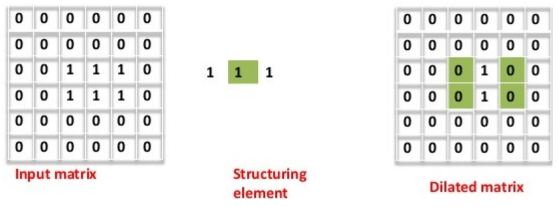

Fig. 3. Erosion process of removing unnecessary pixels. The left matrix represents the original image, the middle represents the dilation kernel, and the right matrix is the resulting eroded image.

The SSS are then dilated to expand the student filled responses, so that any partially-filled answers will be more prominent. Erosion is completed in the same manner as dilation, just using a different kernel with erosion values. Figure 4 demonstrates the process of erosion. Once the morphology is complete, the scan sheets are re-inverted.

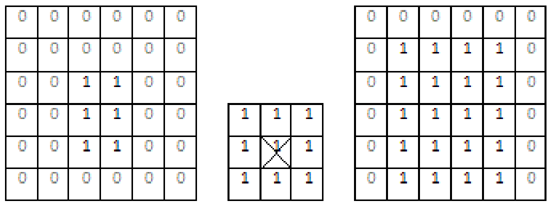

Fig. 4. Dilation process of accentuating features. The left-hand matrix shows the original image, the middle matrix is the dilation kernel, and the right matrix is the final dilated image.

The SSS are scaled to 25 percent of their original size to increase speed of the algorithm. Having less data to sort through and compare will make these final steps much more efficient. Recalling that the bubbles within the BSS were already detected and organized, these are then also reduced in size by 25 percent to remain consistent with the SSS. This step then results in the detected BSS bubbles falling directly over the bubbles on the SSS!

The final step in this algorithm is to determine which bubbles are filled (answers) and which are not. To do so, the mean of each bubble on the SSS is taken and compared to the threshold. Knowing the location and deemed section of each bubble, it is possible to determine the ”meaning” of a filled bubble - whether it is an answer to, for example, question 10 in the first answer section, or a letter in the ”first name” section. These answers are stored again in separate vectors pertaining to their meaning.

Once the SSS are processed, the answers must be compared to the answer key. To find the input answer key, the ”last name” vector is searched through until the answers are equal to ”KEY”. It is important to point out, though, that the algorithm throws an error for SSS that don’t contain both a first and last name, but this is accounted for in the case of the key scan sheet. The process just outlined in the previous paragraph is completed again on the key scan sheet, detecting answers and storing them in their own ”correct answer” vector.

To conclude, the SSS answers are individually compared to the key answers. If the answers are the same, they are considered correct, and if not, it is incorrect. Each student’s final score is calculated with the number of questions in mind (simply from counting the number of answers from the key). A .csv file is output for each individual SSS, which includes the student’s first & last name, university ID, additional information, final grade, student answers, and key answers.

III. Results and Discussion

The algorithm was tested on a data set provided by the instructor, which included a blank scan-sheet, a key, and the student responses. In these cases, there was an over 96% accuracy. Other tabulated data can be seen in the Appendix. The algorithm was also able to account for potentially ”mis-graded” responses on the account of a bad scan, and it would alert the instructor if a grade was especially low regardless of it was ”intentional” or not. The algorithm was also tested on a new data-set recently and it did not perform as well as the initial data set which might indicate that there was not enough ”problem-generalization” in the initial model.

We believe the major drawback of this algorithm was the fact that it relies on predefined Hough Circle thresholds and it relies on the presence of the key. If the key is absent, the algorithm does not grade any of the sheets, returning a score of 0 to all of them. If the algorithm cannot properly detect the 3 Hough circles it also fails to grade the selected sheet. It was noted that some sheets had a non-uniform slant, and since the algorithm solely uses the 3 targets it cannot properly take into account this slant and it fails to grade some questions, depending on the slant.

Given these discussed drawbacks, there are certainly a few areas which may benefit some improvement - not only this but there are other additional features which may be added to enhance the overall user experience. First and foremost, if given the opportunity to go back and refine some parts of this algorithm, it would be useful to create the algorithm in such a way that it would be able to take is several different types of scan sheets. This algorithm in particular only works with this very specific scan sheet, which limits the user experience. One way in which to do this is to perform a sort of comparison algorithm between the user-input blank scan sheet and a student scan sheet. This way, there would be similar features detected beforehand which could be used in calibration, similar to the circular targets used in this instance.

Secondly, implementing some sort of procedure to account for non-uniform slant would be incredibly useful, since it is impossible to predict how well the user may be able to scan the sheets in a proper manner. Thirdly, as mentioned briefly before, a very loft goal would be to implement a character recognition algorithm into this process which would allow the system to grade not only multiple choice answers, but also other forms of questions such as fill-in-the-blank, crossword puzzle, or vocabulary matching.

IV. Conclusions

Concluding on this methodology, when the sheets were read in in a uniform fashion the program worked objectively well. When the program was run over twenty different answer sheets, it returned a 96% correct answer rate as seen in Table IV of the Appendix. Problems arose when answer sheets were read in on a non-uniform slant, and when students used various mediums of utensils such as sharpies. Finally, with the all the errors thrown at us for this problem, the program would be able to preform under conditions when these problems would not occur. All it would take for this program to run smoothly would be have a set method of implementing the answer sheets so that it could return 100% accuracy on returning the answer provided by the test taker.

V. Appendix

TABLE I
Answer Sheets Results

Name	Score	Scored Correctly
Key	100%	100%
Carl Salvaggio	28%	84%
Barack Obama	0%	72%
George Bush	72%	96%
Ronald Reagan	100%	100%

TABLE II
Reference Guppy Results

Name	Score	Scored Correctly
Key	100%	100%
David Kelbe	100%	100%
Emily Berkson	92%	98%
Amanda Ziemann	100%	100%
Ryan LaClair	22%	98%
Paul Romanczyk	32%	100%
Phil Salvaggio	0%	100%
Carl Salvaggio	98%	100%

TABLE III
Test Guppy Results

Name	Score	Scored Correctly
Key	100%	100%
David Kelbe	100%	100%
Emily Berkson	92%	98%
Amanda Ziemann	100%	100%
Ryan LaClair	22%	98%
Paul Romanczyk	32%	100%
Phil Salvaggio	0%	100%
Carl Salvaggio	98%	100%

TABLE IV
Averages of each group and a total program accuracy.

Name	Average
Answer Sheets	90%
Reference Guppy	99.5%
Test Guppy	99.5
Overall Accuracy	96.3%

References

[1] https://www.collegeboard.org/releases/2018/more-than-2-million-students-in-class-of-2018-took-sat-highest-ever

[2] *Use of the Hough Transform to Detect Lines and Curves in Pictures*, Dude & Hart, 1971