Mobile Based Auto Grading Of Answersheets

Vaibhav Chidrewar, Junwei Yang, Donguk Moon
Department of Electrical Engineering
Stanford University
Stanford, CA
{vaibhavc, junweiy, dongukmoon}@stanford.edu

***Abstract*—We explore an image-processing algorithm for auto grading of answer sheets using the mobile phone. Multiple-choice questions are widespread mechanism used by schools and universities to test student performance. They are common on standardized tests. Big organizations use OMR forms coupled with OMR software and dedicated scanner for grading multiple-choice questions. But small institutes and individual teachers cannot afford this costly setup and have to do time consuming manual grading. So we propose a mobile application that can be used by anyone at any time to grade answer sheets and save the valuable time spent grading manually. First we will discuss the OpenCV implementation of the image-processing algorithm and report on its challenges. Second we will give an overview of what was implemented on the android phone. Finally we will summarize the challenges and possibilities for future work.**

Keywords—image-processing; OMR; homography; key-point detection;

I. INTRODUCTION

Not only in America but also throughout the world, multiple-choice questions have become an integral part of the educational system. Standardized tests also use multiple-choice questions to judge student's academic performance. The American College Testing (ACT), the Scholastic Aptitude Test (SAT) and Law School Admission Test (LSAT) are just some of the many standardized tests conducted in the US to assess candidates on a common platform. Every year millions of students take standardized tests and they have to answer various questions asked by darkening bubbles in OMR sheets. Current solutions for evaluating these OMR sheets are expensive and need dedicated scanner, OMR software and buying customized OMR sheets. So small organizations, institutes, individual teachers and tutors cannot use this convenient method of grading without spending lot of money. They resort to manually grading answer sheets. To grade a standardized test responses of a student takes 10 minutes on an average. The motivation of our project comes from the idea that we could build a mobile app that will assist the instructors in auto grading these answer sheets and saving their valuable instruction time. Now a day's smartphones are ubiquitous. So there will be no extra cost associated with using this smartphone-based solution. And it is very easy to use solution without lot of steps or setup.

Figure 1 shows a sample answer sheet that we designed for this project.

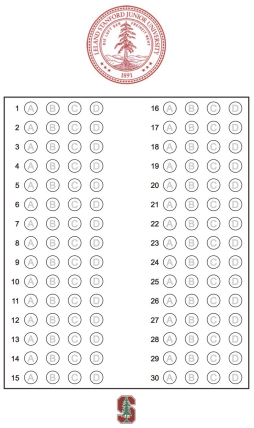

Figure 1 Answer Sheet Format

We envisage that our app would be able to do the following: (i) automatically identify the answer sheet in the photo, (ii) detect response/s for each question and (iii) compare student responses with reference answers.

We have designed our algorithm assuming that we know the position of circles corresponding to each question.

The rest of this report is organized as follows. In Part II we provide the block diagram and explain the running sequence of our application. Part III provides some insight into the Android implementation of our algorithm. Experimental results about our algorithm are mentioned in Part IV. Finally, Part V summarizes our results and lists possible improvements to our application.

II. APPLICATION FLOW

Fig. 2 shows the complete application flow. This application starts with a main page that shows three buttons. The first button, called Capture Reference Answer Sheet, is to take a picture of reference answer sheet and extract the correct answer table from the image. The second button, called Capture Student Answer Sheet, is to take a picture of student's answer sheet and extract student's answer for each question. The last button on the bottom of the main page, called View Result, is to compare the correct reference answers with student's answers, and show the result table with different highlighting on correct and wrong answers.

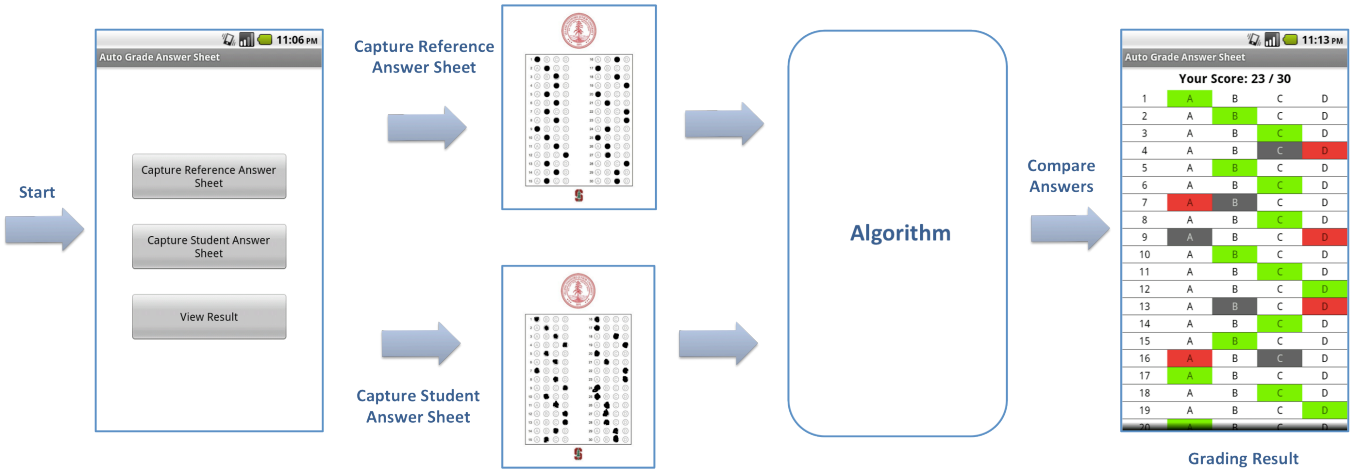

Figure 2 Application Flow

A. Capture Reference Answer Sheet

This application works based on pictures taken by Android smart phone camera. When Capture Reference Answer Sheet button is pressed, the camera application gets started so that a user can take a picture of reference answer sheet marked with correct answers. After taking a picture, our main image-processing algorithm is invoked and the captured image is processed to extract correct answers. These extracted correct answers are stored in a list data structure to be used later when grading function is invoked by View Result button.

B. Capture Student Answer Sheet

The functionality of this button is the same as Capture Reference Answer Sheet, except that a taken picture is student's answer sheet. Our main image processing algorithm is used to extract student' answers and the extracted data is stored in a list structure.

C. View Result

When this button is pressed, two list data structures with student answers and correct answers are compared to generate a graded result table that show correct answers and wrong answers with different color highlighting. Correct answers are highlighted in green. Incorrect answers are shown in red and correct answer for it is shown in grey color.

III. ALGORITHM

As described in the previous section, after taking a picture of answer sheet, our main algorithm is applied to the picture to extract the answer location. This section aims to describe our algorithm details.

Our algorithm is mainly divided into two parts. The first part is to apply image registration techniques to align a taken image with the template image and compute the transformation matrix. The second part is to apply the transformation matrix to center pixels of all answer circles and count the number of black pixels around the region to figure out whether the circle is filled or not.

A. Image registration by finding homography

In the field of computer vision, any two images of the same planar surface in space are related by a homography. As the answer sheet and the template has the same planar, we can find the homography between these two images to get the transformation matrix. Then by using this transformation matrix we can calculate new location of center point of circles in the captured image. In order to get an accurate homography, we need some distinctive features which can be matched easily. So we had two logos in the template, one at the top and the other at the bottom.

The image registration can be performed in the following 5 steps as shown in figure 3: key points detection, key points extraction, key points matching, finding the transformation matrix and finally finding all the answer locations using the transformation matrix.

- Step 1: key points detection

This step is to detect the key points of the template and the answer sheet. There are several options we can choose : FAST detector, ORB detector and SURF detector. After trying all these 3 detectors, as expected we found that FAST detector is the fastest one to extract features. But SURF is the most robust one, which detects better features. Therefore, we decided to use SURF. SURF detector (Speeded Up Robust Features) is a robust local feature detector. It uses an integer approximation to the determinant of Hessian blob detector, which can be computed extremely fast with an integral image. For features, it uses sum of the Haar wavelet response around the point of interest. These also can be computed using the integral image.

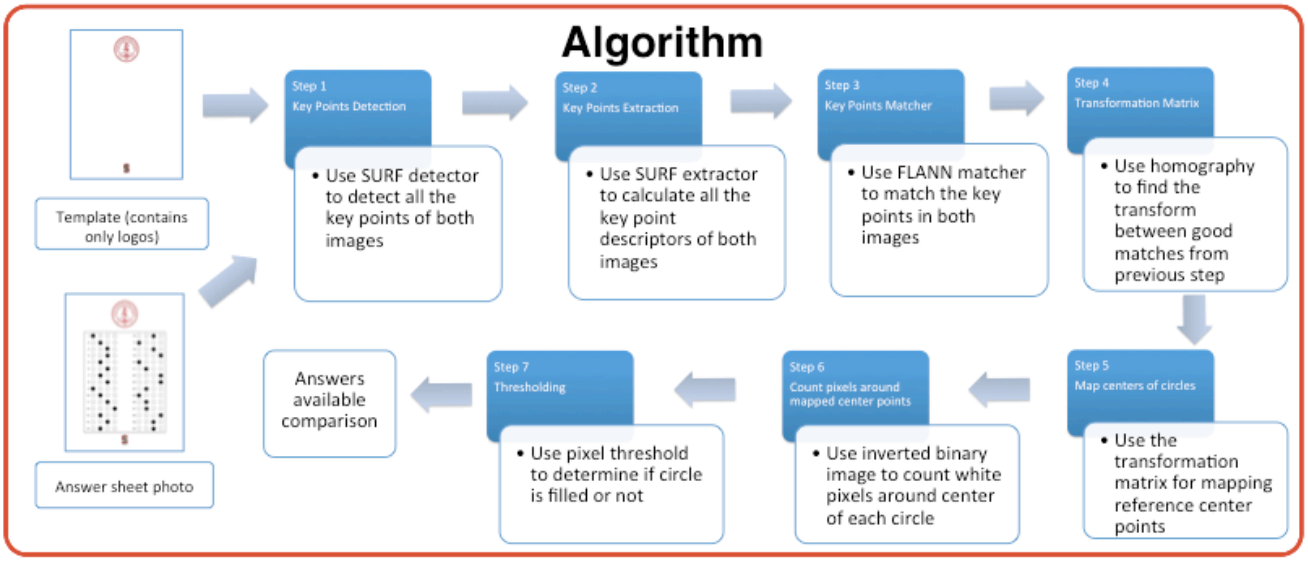

Figure 3. Algorithm Sequence

➢ Step 2: key points extraction

This step is to compute the descriptors of the key points detected in the first step. We observed that this is the most time consuming step when we are processing image taken by camera since it has lot of key-points descriptor extraction takes lot of time.

➢ Step 3: key points matching

For this step, we considered two matchers: FLANN-based matcher and the brute-force matcher. FLANN means Fast Library for Approximate Nearest Neighbors so this matcher is generally faster to compute the matches than the brute-force matcher. Thus the FLANN-based matcher was chosen to use. After using FLANN matcher to get all the matcher, the invalid results and bad matches need to be discard. To implement this, Nearest Neighbor Distance Ratio is used to filter out bad matches. Basically the minimum match distance is found at first and then we multiply this minimum distance by a reasonable ratio to filter out matches having longer distance than this number. In our case, we selected 3 as reasonable ratio.

➢ Step 4: find homography

After all the good matches are identified, we can use the result of key points matches to find the homography and get the transformation matrix.

➢ Step 5: map center of circles

In this step, we apply the transformation matrix to the location of the center of all circles of the template. This computation results in the circle center points projected on the captured answer sheet.

Figure 4 shows the result of image registration. The picture on the left is a student answer sheet taken by camera. It clearly

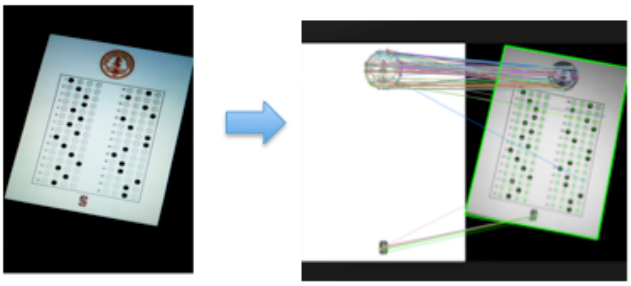

Figure 4. Image Registration Result

has some rotation and tilt. Then we applied our image-processing algorithm and the result is shown on the right. The green circles on the image are all the answer circle centers we identified. As it is shown, all the answer circles are detected accurately.

All these steps have been implemented by using OpenCV Package for Android.

B. Post-processing to extract answers

Since all the locations of circle centers have been computed, the remaining step is straightforward. In step 6 shown in figure 3, the inverted binary image is used to count the white pixels around center of each circle. To get the binary image, we use Otsu's method to get the threshold and then apply this threshold to the original image to get the binary image. After this step, for each answer circle center, the number of white pixels around it is computed. Finally in step 7, a threshold of the number of white pixels is applied to each answer circle center to determine if the circle is filled or not. After applying

this technique for every circle center, we get all the information about answers.

IV. Experimental Result

The application is based on camera images whose quality can be affected by capturing conditions. Images taken by camera can be tilted, rotated, or blurred by poor illumination condition. In order to test the robustness of our algorithm on these variations, we took pictures from different perspectives and illumination conditions, and applied our extraction algorithm on these images.

- For well-aligned images with uniform brightness, our algorithm works well. Our iterative experiments show almost 100% accuracy in this case.
- For small tilted images with no rotation, our algorithm still works fine, but less accuracy than the first case.
- For images with large tilt and rotation, the accuracy is hampered with up to 26% result being inaccurate.
- For images with blurred logos at the top or bottom due to bad illumination condition, our algorithm works bad, because logos are key features used for key point matching.

We summarized our experiment result on images with different rotation angle and tilt. We iterate 20 times for each case, and calculate average and standard deviation.

Rotation/Tilt Angle	Average Accuracy	Standard deviation
Flat/No rotation	100%	0
Flat/90 degrees	92%	0.15
Tilt/No rotation	85%	0.12
Flat/30 degrees	79%	0.3
Random tilt and rotation	74%	0.33

V. Future Work

Current algorithm only uses FLANN matcher to find key-point correspondence, and this turned out to be not optimal, because it does not take into account any geometry information around key-point. By introducing algorithms that uses geometry information around key-points, we expect to see more robust result even with large rotation. Our algorithm currently takes ~8 sec for processing a captured image. We can reduce this time by adjusting various parameters of SURF key point extraction. But we could not do these adjustments since we used Java wrapper for OpenCV. It does not have the flexibility offered by native OpenCV code. Current answer sheet format does not have student and test identification feature. We can use the same approach of filled circle detection for detecting test id and student id.

VI. Acknowledgent

We would like to thank David Chen and Matt Yu for their invaluable help and advice regarding the Android implementation of our algorithm. We would also like to thank Professor Bernd Girod and our mentor Huizhong Chen and for their continued support through our project.

References

[1] Richard Szeliski, Image Alignment and Stitching: A Tutorial, December 10, 2006.

[2] EE368 Class Project, Mobile Lottery Ticket Recognition Using Android Phone, Ashish Gupta, 2012.

[3] Ashish Arora, GUI Based OMR System for Recognition of filled Bubbles in Scanned OMR sheets in Absence of OMR Machine, Course Project at IIT, 2011.

[4] David Lowe, FLANN – Fast library for Approximate Nearest Neighbors, 2010

[5] Features2D + Homography to find a known object, http://docs.opencv.org/doc/tutorials/features2d/feature_homography/feature_homography.html, 2011

[6] http://morf.lv/modules.php?name=tutorials&lasit=2#.UbAu0fZhs5K, Complete guide using SURF feature detector, 2012

[7] B. Girod. "EE 368 Digital Image Processing Notes". EE 368 Digital Image Processing. Spring 2011-2012