An Application for Automatic Multiple-Choice Test Grading on Android

Rodrigo Teiske China, rodrigo.china@aluno.ufabc.edu.br
Francisco de Assis Zampirolli, fzampirolli@ufabc.edu.br
Rogério Perino de Oliveira Neves, rogerio.neves@ufabc.edu.br
José Artur Quilici-Gonzalez, jose.gonzalez@ufabc.edu.br

Universidade Federal do ABC, Santo André, Brasil

Submetido em 01/09/2015
Revisado em xx/xx/xxxx
Aprovado em xx/xx/xxxx

Abstract: Considering the need to grade a large number of tests in short periods of time, sometimes with limited access to technology and resources, this work presents a low-cost alternative for the task. The mobile application "MCTest – Multiple Choice Test" for Android systems, available for free on Google Play Store, can grade tests instantaneously with up to a 100% accuracy and includes tools for statistic analysis of the test results.
Keywords: Multiple-Choice Tests. Mobile Devices. Image Processing.

Título: Um Aplicativo para Correção Automática de Teste de Múltipla-Escolha.
Resumo:
Tendo em vista a necessidade de corrigir um grande número de testes em curtos períodos de tempo, muitas vezes com limitações tecnológicas e de recursos, é apresentada neste trabalho uma alternativa de baixo custo para a tarefa. O aplicativo móvel "MCTest – *Multiple Choice Test*" para sistemas Android, disponível gratuitamente na *Google Play Store*, corrige provas de múltipla escolha instantaneamente e com precisão de até 100%, além de fornecer ferramentas estatísticas sobre o resultado das provas.
Palavras chave: Teste de Múltipla-Escolha. Dispositivos Móveis.
Processamento de Imagens.

Introduction

There are several popular programs used to generate tests (CIENCIAMAO, 2015; FISTEUS et al., 2013), as well as software and devices aimed to this specific task, generally involving a significant investment. In addition, to the best of our knowledge, there is no specific literature describing methods and techniques to implement test grading or similar image processing problems. Using a technique known as *Optical Mark Recognition* (OMR, 2015) it is possible to acquire discrete data contained in predefined forms and, with an image *scanner*, detect the presence of marks in the reserved spaces. This technique often employs OMR specific scanners or image scanners, in which case software does the processing, sacrificing performance to the advantage of lower costs and flexibility when employing custom forms (SPADACCINI, 2009; NGUYEN et al., 2011; SEN et al., 2010; SAIPULLAH, et al., 2012).

With the use of mobile devices, the processing is subject to errors and prolonged processing times, due to the manual acquisition of images, resulting in difficulties to position the sheet at the right angle and distance, parallel to the camera lenses. Furthermore, factors related to image quality, such as uneven lighting, shadows and camera distortions add new technical difficulties. Nevertheless, there are several OMR and OCR programs available in the market, including applications for smartphones and tablets capable of recognizing marks, barcodes and QR-codes (SPADACCINI, 2009; NGUYEN et al., 2011; SEN et al., 2010; MOLLAH, et al., 2011; CHINNASARN; RANGSANSERI, 1999).

There are also several papers and application quizzes demonstrating how to use mobile devices and web services, such as the iQuiz (RODRIGUES, et al., 2013), an integrated assessment environment to improve Moodle quiz. The MobiMonitor (LUCENA, et al., 2014) is a mobile App for Monitoring distance courses in the Amazon region. An authoring verification environment for quizzes applied on the web is presented in (GORDILLO, et al., 2014) and an interactive environment suitable for classrooms that use mobile devices based on problem-solving skills is presented, (DEB, et al., 2014) showing how mobile devices and distance learning can mutually benefit from each other.

In this paper we describe the implementation of the App “MCTest”, an automated multiple-choice test grader for mobile devices running Android OS. MCTest uses a simplified answer sheet, when compared to regular OMR forms, with variable size, according to the desired number of questions and possible answers for each question. Using any word processing software, such as LibreOffice or MS-Word, a table can be constructed to include rows and columns according to the test specifications, even differentiating among multiple templates, for variations of the same test, also supporting weighted questions.

MCTest aims to help teachers and evaluators in the time consuming task of grading student tests, but other areas may benefit from its use, such as quickly access candidates scores in selection processes. Test correction data is stored for further analysis, giving evaluators the possibility to obtain qualitative information about the test and its overall efficiency.

Methodology

Figure 1 summarizes all the activities in the App for grading templates and tests. Details of this figure will be presented in this section.

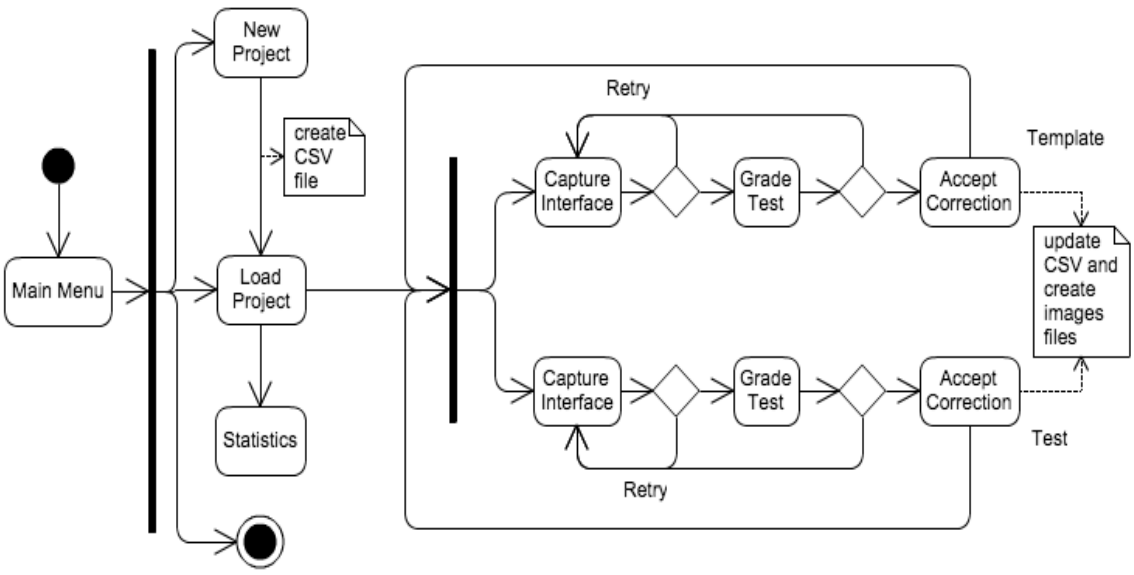


Figure 1. All stages of the MCTest App can be seen in this fluxogram.

A. MCTest characteristics

The maximum number of questions in one specific test depends only on the resolution of the camera of the device used. Despite the fact that our tests have never considered more than 40 questions, it is possible to expand the number of rows and columns or even to divide the answers into multiple tables, separating one from another by at least one centimetre. In this case, the user is required to add the results manually to obtain the final grade.

A project created in MCTest is stored in a subfolder within the device, supporting up to 16 different test variations with 5 alternatives, what doubles for each extra alternative. This way, a teacher or evaluator can use several different templates for grading one project or class. As an example, the same test can have questions and alternatives shuffled in order to prevent cheating, using different templates for grading. Our trials used up to 10 questions with five alternatives each, resulting in a maximum table size of 12x7, counting the edges used for detection. The last column at the right edge is used for test identification purposes, giving five bits, four for test type identification and one reserved for template detection, thus allowing 16 different template identifiers (see red marks in Figure 2). The last line in the table allow for weighted evaluation of specific questions, adding an extra value for marked columns.

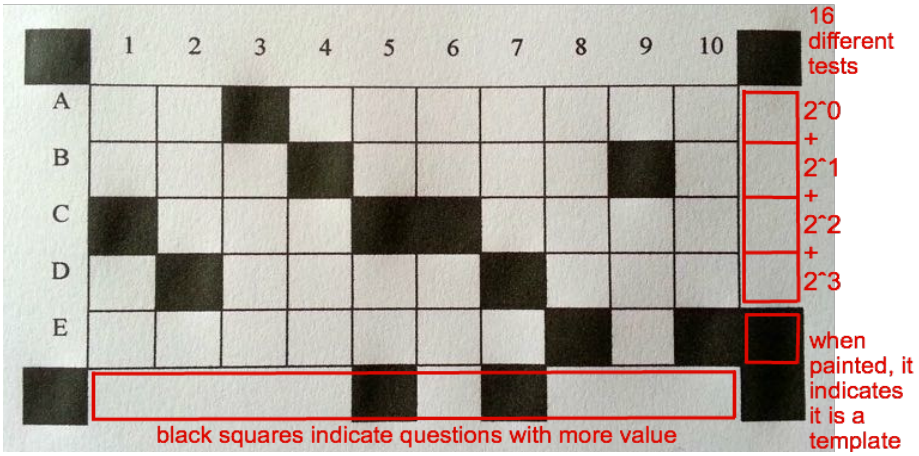
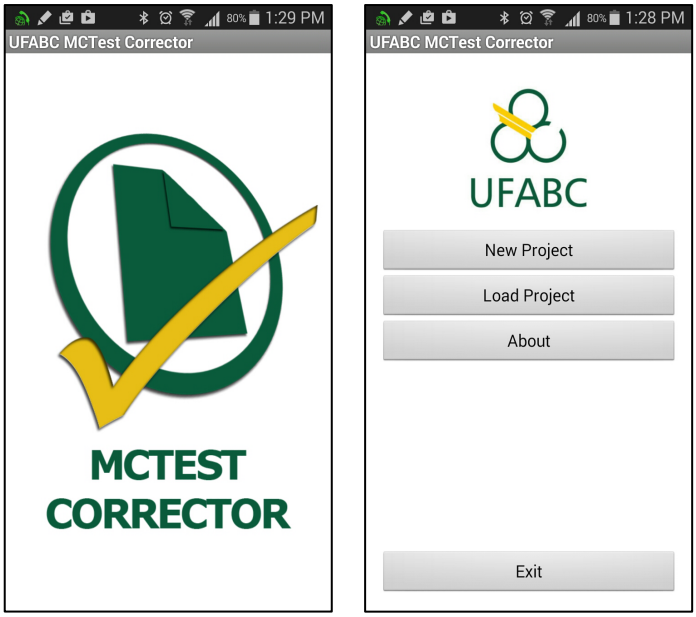


Figure 2. The red marks highlighted in the table indicate added value for weighted evaluation (bottom), different templates (right, top) and template identification (under).

The first step in grading tests is to scan the template, or templates, containing the correct answers for each question in each type of test, e.g., in our case

studies we used five squares for choices, therefore allowing 16 different templates used for grading. Having registered all templates, the user can start to grade tests. Trying to grade a test without having scanned the proper template results in an error message.

The following figures (screen captures) illustrate the procedure for grading multiple-choice tests with MCTest. The device used was a *Samsung Galaxy Note 2* smartphone running Android 4.4 (*KitKat*). Figure 3a shows the opening screen. Figure 3b shows the main menu, bringing the choices to create a “New Project”, “Load Project”, “About” MCTest and “Exit” to finish execution.



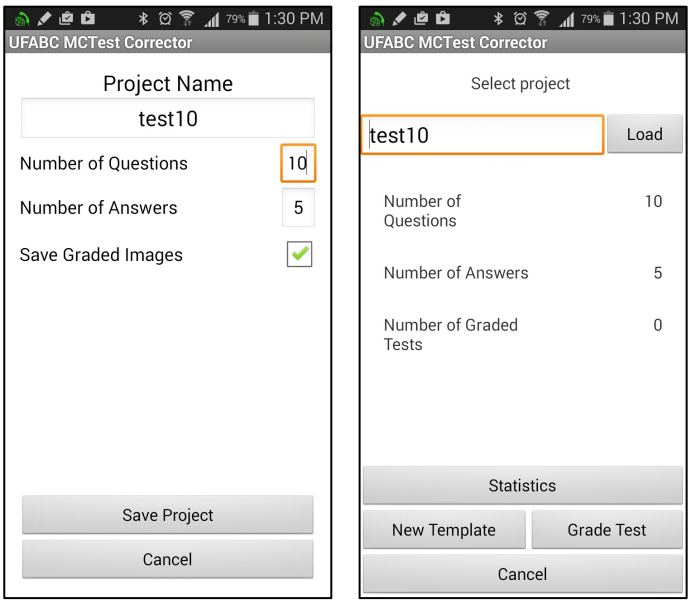
(a) (b)

Figure 3. a) Welcome screen upon opening. b) App main menu.

Figure 4a brings the “New Project” menu, where the user is required to name the Project and specify the answer sheet format, containing the number of questions and answers in the table. A checkbox gives the option to save the captured test images. Figure 4b shows the interface used to capture template sheets and test sheets.

By clicking “Save Project”, the App creates a folder inside the application to store the project related files. Then the user is required to provide the templates

for each test, using the option “New Template”, as shown in Figure 4b. Finally, having processed all the templates, the user can start grading tests using the option “Grade Test”.



(a) (b)

Figure 4. a) New Project and b) Open project screens. After naming a new project and capturing at least one template sheet, “Grade Test” is enabled.

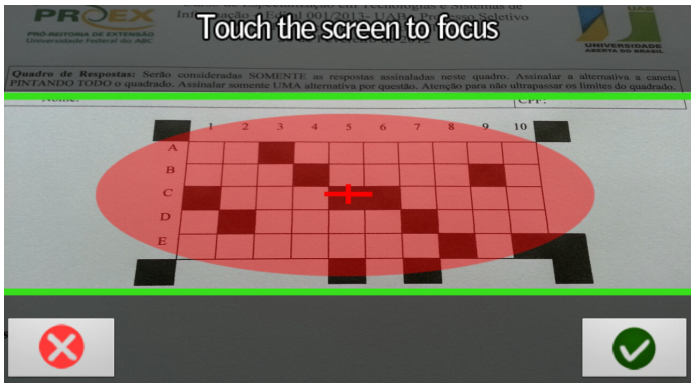
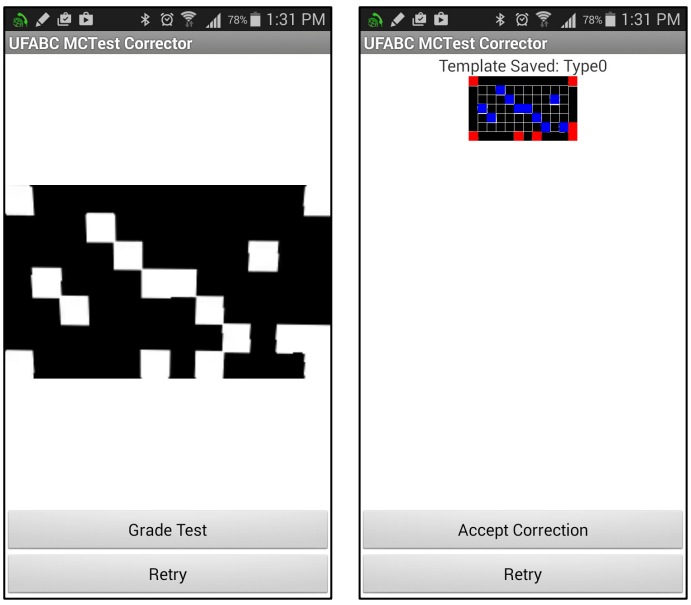


Figure 5. Capture interface. Test example with 10 questions.

In the capture interface, the user has to position the table inside the delimited capture area, using touches to focus the camera. The four black squares delimiting the table should remain outside the red ellipse and some extra white background added around the image to improve detection. After obtaining a

satisfactory fit, capture an image by pressing the green button. Figure 5 shows the interface used to capture template sheets.

The project website in (MCTest, 2015) contains several examples of test sheets in different formats for use with regular word processing applications. Figure 6 shows an example of captured template sheet before and after being processed.



(a) (b)

Figure 6. Captured template sheet a) before image processing. After proper detection, the option “send for grading” (“Grade Test”) sends the image for processing, as shown in b). If the capture was not successful, the user has the option to “Retry”. Clicking on “Accept Correction” generates a template sheet file in the project folder.

After capturing one or more template sheets for a given project (Figure 4), capturing and grading a test (Figure 7) follows the same procedure (notice the bottom-left mark distinguishing between template sheets and test sheets).

Figure 8 shows the correction and grading of the captured test. When the detection is satisfactory, the user can chose “Accept Correction” to perform grading.

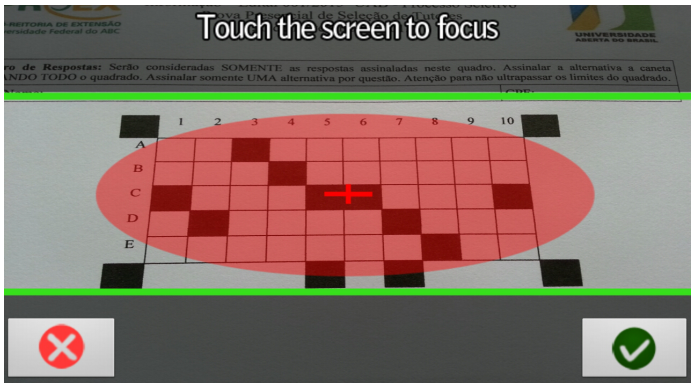
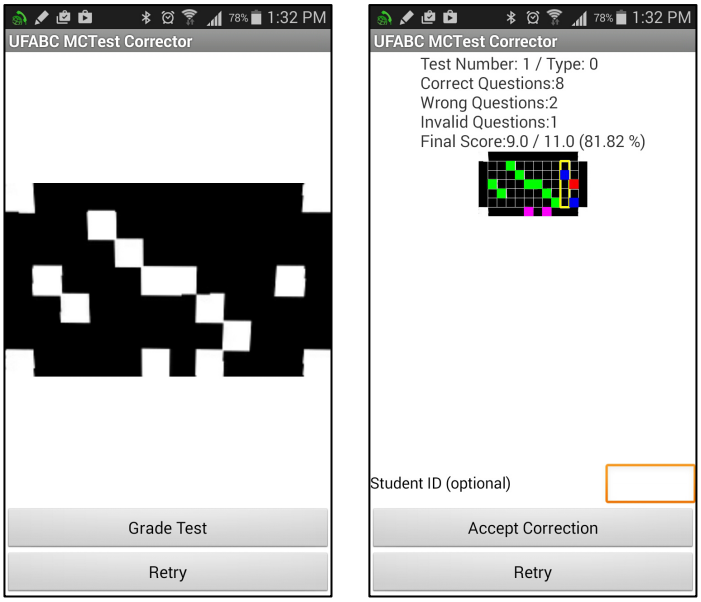


Figure 7. Capturing a test with the capture interface.



(a) (b)

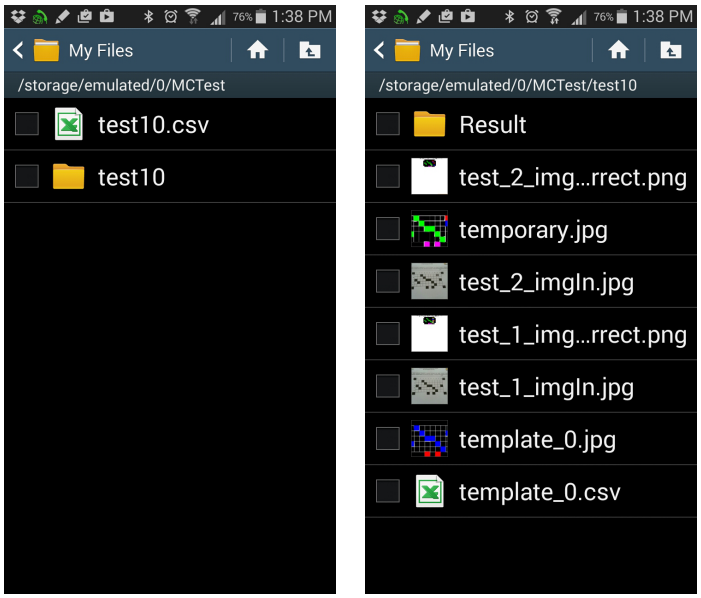
Figure 8. Captured test sheet a) before image processing. Selecting the option “Grade Test” generates the correction shown in b).

The corresponding template sheet is loaded and used to compare the data extracted from the two images, generating an image showing coloured blocks for correct answers (green), wrong answers (red) and highlighting columns left blank or with two or more answers for the question (yellow rectangles).

The program creates a spreadsheet to store the results sequentially, allowing the evaluator to perform test analysis and generate statistics later on. By clicking “Accept Correction”, the user adds an entry containing the test results to

a spreadsheet named “ProjectName.csv” (a text file in “Comma Separated Values” format). To retrieve the file, the user has to connect the mobile device to a PC and access the MCTest folder. Figure 10 exemplifies the contents of this file.

In addition, if the choice “Save Graded Images” (Figure 4a) is set, the program also saves the images as in Figures 7 and 8b to the project folder. Figure 9 brings the resulting file structure.



(a) (b)

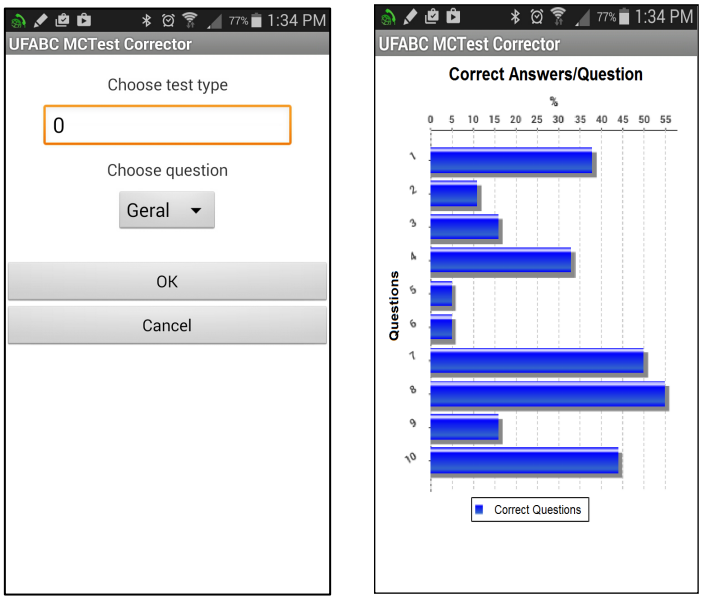
Figure 9. a) Image folder for project “test10” and the spreadsheet containing results; b) images corresponding to the template sheets and tests graded.

10	5	2	1													
num	ID	type	invalid	wrong	correct	end	answers	1	2	3	4	5	6	7	8	9	10
1	0	0	1	2	8	9.0		C	D	A	B	C	C	D	E	O	C
2	0	0	0	2	8	9.0		C	D	A	B	C	C	D	E	A	C

Figure 10. Contents of “test10.csv”, the spreadsheet with results.

Performance statistics allow one to analyse test results after grading all tests, detecting weaknesses and improving the contents of teaching material. The option "Statistics" (in Figure 4b) opens the screen shown in Figure 11, where

analysis can be generated for each test type with "Choose test type", for each individual question or for all questions (from the pull down menu "Choose question", choosing "Geral"). After choosing test type and questions (or general), pressing “OK” MCTest will present a graph as in Figure 11b.



(a) (b)

Figure 11. a) Statistics for a test of type 0, considering all questions, button “General”. b) Statistics relative to the number of correct answers.

The captured images are then processed inside the devices by an image processing algorithm developed in Java. The code makes use of the OpenCV library (Open Computer Vision) (OPENCV, 2015), imported into the developing environment ADT (Android Developer Tools, ver. 22.3) (ANDROID, 2015), which includes essential components for Android development.

B. Image processing

The image captured with the device camera is a colour image with 24 bits/pixel. The resolution may vary depending on the device’s camera specifications. The raw image (Figure 12, upper panel) is then cropped (Figure 12, lower panel) and has its colour map reduced to binary (0 is black an 1 is white), after inversion of the values for better visualization. This also saves

space in the limited device storage. These steps are explained in detail in section D “Algorithms”.

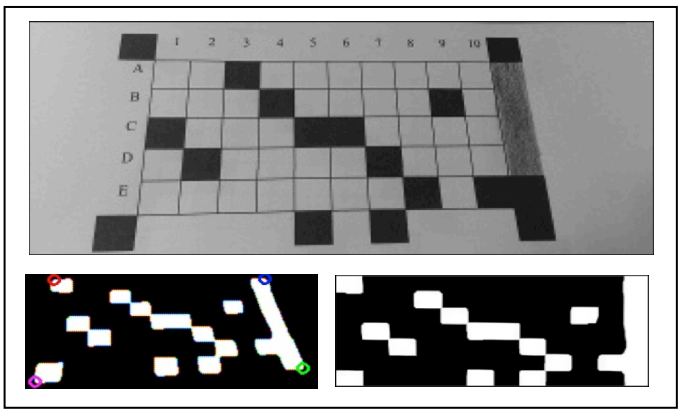


Figure 12. Example of capture and perspective correction of a template sheet.

Once acquired (Figure 12), the image passes through a series of filters and transformations (GONZALEZ; WOODS, 2010) in order to improve quality while removing noises and reducing the effect of shadows. Some of the filters and transformations used are:

- Decrease resolution and color depth to greyscale;
- Gaussian blur: smoothens the image to reduce artefacts;
- Adaptive threshold: adjusts the histogram and normalizes the light and dark areas in order to reduce the effect of shadows;
- Negative image: invert black and white;
- Morphological opening: consists of an erosion followed by dilatation, to further eliminate white dots or noises;
- Morphological closing: a dilatation followed by erosion, similar to the previous one, but to eliminate black dots;
- Distance transformation: Using the position of each corner, projected into an ellipsoid, it is possible to obtain the angle information (Figure 12, bottom left);

- Perspective transformation (WarpTransform) (OPENCV, 2015; GONZALEZ; WOODS, 2010): based on the information obtained in the previous step, adjusts the viewing angle of the image correcting the perspective.

After successful pre-processing, the result should be a binary image showing only marks in white, as presented in Figure 12, (bottom right image). The image is ready for translation into a table and subsequent comparison with the appropriate template, otherwise the image will create a table that is a new template.

C. Optical Mark Recognition

The final step is to compare the information presented in the image with the appropriate template, or to create a new template, in case the template identifier bit is marked (at the bottom of the right column, see Figure 2).

Upon comparison with the appropriate test template, the App calculates the score and displays the results along with information on each question. In the upper panel of Figure 13, green marks denote right answers, red denotes wrong placed marks and a yellow rectangle denotes a column where a mark is absent, undetected or with more than one choice. Notice that some questions with a mark at the bottom line have value 1.5 in this test, whilst the remaining questions values are rated 1 point. It is possible to adjust the extra value in “project configuration”. The information atop identifies test number (ordinary) and type (template number), number of correct and wrong answers, the final score in points and percentage. The lower table in Figure 13 is the template used for comparison. If the user accepts the correction, the program records the data into the spreadsheet “ProjectName.csv” as previously described.

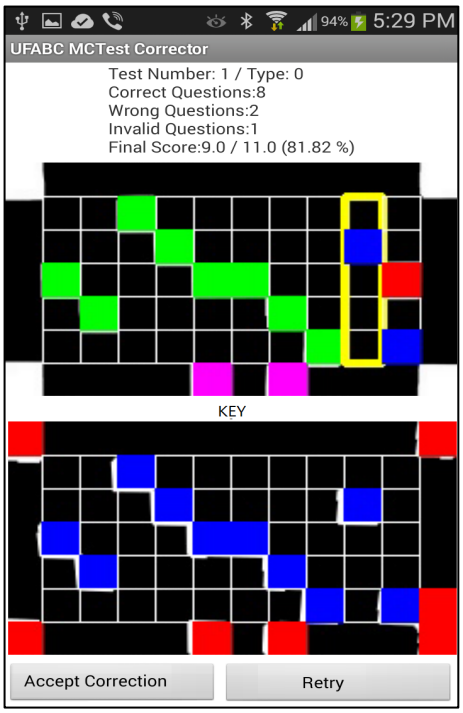


Figure 13. Evaluation example, with test results displayed on top.

D. Algorithms

Figure 14 briefly emphasizes the steps taken to process the information from the captured camera image (a) to the final binary bitmap used to extract the answer table (e). After cropping (b, left) the original image (a) to fit inside the green guidelines (as in Figure 5 and 7) it has its colour depth reduced to greyscale. Independent of the camera resolution, the image is resized to a fixed, manageable dimension. Both steps help to address compatibility issues across multiple devices and camera configurations. Then, the image is successively subjected to a Gaussian softening (b, center) and an Average Median Blur (b, right) to eliminate high frequency noise or salt and pepper noise. To reduce the impact of shadows, an adaptive threshold (c, left) is applied, followed by a combination of morphological filters (c, right) to eliminate more noises. After these steps, the image is ready for segmentation (d and e).

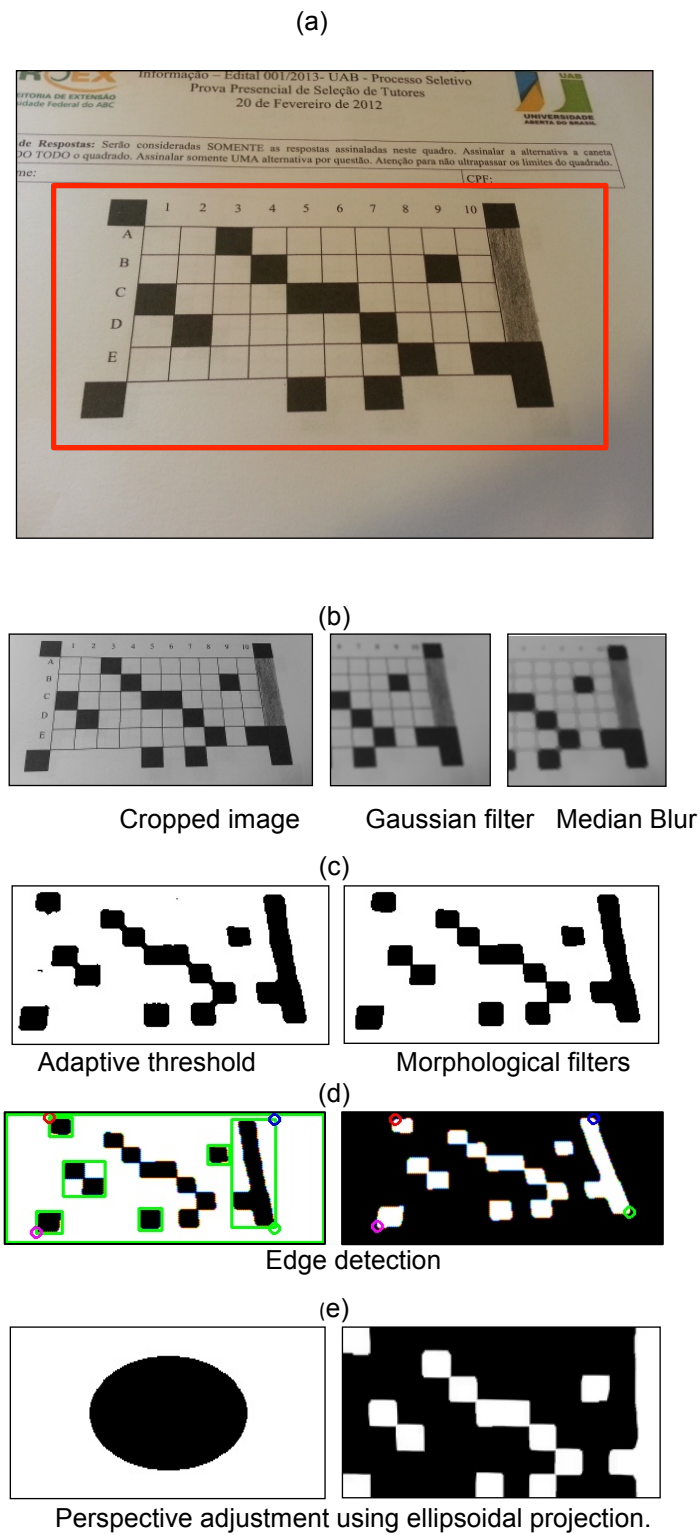


Figure 14. a) Original image (Temp), as captured by the device camera, with the cropped section highlighted in red (ROI). b) The image is resampled, colour depth reduced to greyscale and filtered. c) The binary image as result of an adaptive threshold and a combination of morphological filters to eliminate noise. d) Finally, the perspective is adjusted using e) distance transformation in an ellipsoidal projection.

Discussions

A. Tools for Education

Smartphones can boost productivity when employed in Education not only for students, but also for teachers as a tool for education. In fact, the convenience to access the digital world has given more emphasis on the ability to find information than to memorize it. On the other hand, the increase in the amount of information available everyday makes the kind of knowledge acquired by youths increasingly superficial.

When properly used, smartphones and gadgets in general can make learning easier. For instance, it is possible to teach mathematics and physics by means of games. Some important aspects of smartphones as tools for education are:

- They offer the possibility of mobile learning, i.e., one can get information everywhere, anytime, answer tests, upload homework, communicate with other students, send and answer questions, read emails, listen to podcasts, instructions etc.;
- Teachers can more rapidly give feedback to their students about their grades;
- Smartphones can enhance learning by interacting with objects of learning. It is easy to take pictures of places, animals, zoom in small insects and use this material to document the lesson learned;
- As a learning tool, the smartphone can make education cheaper, as a single teacher can reach more students;
- They can make learning a funny and familiar activity.

There are also studies suggesting that the use of gadgets in excess may compromise the capacity of concentration, hinder intellectual development and even make some people dependent on technology.

Before starting using the App, it is important to highlight some of its strengths and weaknesses, hence eliminating possible mistakes that can result in bad recognitions or even entire tests failing detection.

B. Preliminary tests

Before applying a test to a particularly large number of students, we recommended printing an answer sheet and some tests to evaluate its accuracy, given the parameters of shape and size of the built table. It is also a good idea to test the mobile device one is planning to use for grading.

Test models are available for download from the project site and can be used as a reference or modified for use in a particular scenario (MCTest, 2015).

C. Filling the answer sheet

The experiments indicate that malformed markings (blurred, not properly filled, e.g. just an X), leaving much of the background undetected, results in lower grades than expected. This requires the intervention of the grader to do a proper marking or to increase the grade on a need basis. To spot possible misdetection, the App shows a yellow rectangle around every question left blank, improperly filled or containing more than one marked square (invalid).

We made available as examples two tests graded with MCTest (2015) (under the names classA2 and classB2), for 41 and 64 students respectively. The applied tests of 4 different types (variations) all have 16 questions, each containing 5 possible answers. In the first example, student number 19 wrongly filled some squares with an X instead of completely painting the square. To resolve this sort of problem, it is necessary to fill in the square with a pencil and repeat the correction (shown in correction n. 42). The same problem occurred in the second example (classB2) with student 49. The names and identifiers of the students were blurred to protect their identity. In both case studies, performed in real scenarios, MCTest had 100% accuracy when students correctly filled in the answer sheets. MCTest considers blocks “filled” if they contain at least 50% of

the area painted. That is why it is important to include instructions in the test, explaining how to fill the answer table completely, using black pen or pencil, to avoid lower grades.

D. Include a white buffer

It is important to leave a blank area around the table (particularly in between the green detection lines in figures 5 and 7) to avoid additional noise on the area surrounding the ROI (Region of Interest).

E. Lighting and distortions

Dim light might affect detection, resulting in misdetections due to shadows and camera artefacts. Choose an environment with uniform ambient light and preferably no reflections in the sheet.

It is also important to capture the test in a plane surface, avoiding unnecessary distortions in the image.

F. Performance

Another important factor when grading a large number of tests is the processing time for each test. A preliminary version (in MATLAB) could return a result instantaneously, but required a predefined stable structure in place for capturing images (ZAMPIROLLI; QUILICI-GONZALEZ; NEVES, 2013). As expected, the processing time in mobile devices depend on the processing capabilities of each device. Yet, under conventional situations, it was still preferable to use mobile devices (even cheaper models) when compared to the PC-based solution. The added mobility and easiness to position the camera granted lower “per-test” processing times, varying from 3 (Motorola XT918) to 7 seconds depending on the device used. For instance, a Samsung Galaxy Note 4 takes less than one second from the capture of the test image to displaying the result. In other words, it is processing the result in real time.

G. Future works

We are currently working on an iOS (Apple iPhone/iPad) version of the App. A Windows Phone version is under consideration.

A server based correction engine and the archiving of test results are under study. Both will allow the use of protocols, such as FTP and VPN, to assist a team in order to split the efforts of grading large quantities of tests in a project. In this model, the mobile phone would send the image to the server to be graded and archived, adding the results to a shared spreadsheet. Parts of the code already exist to accomplish this task. In the server-based model, implemented using Python and OPENCV (2015) secure connections are established using access permissions to increase privacy and security.

Another study aims to integrate grading data with grade management systems, commonly used in schools and universities, taking special care about security issues involved in authoring authenticating and verifying tests and grades.

Conclusion

The image segmentation technique used in this work, known as Mathematical Morphology, demonstrated how limitations resulting from images captured under non-ideal conditions can be circumvented, giving even better results than experiments conducted under controlled conditions.

Despite the fact that the operation system of choice for the App was Android, development for any other platform in existence is possible, given that OpenCV is available for most platforms nowadays. It would be relatively simple to port this solution to any another OS, mobile or desktop.

The 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. Improvements are currently underway to enhance recognition rates in some specific cases.

References

ANDROID, Available in http://developer.android.com/sdk. Access in 24.05.15.

CHINNASARN, K.; RANGSANSERI, Y. An image-processing oriented optical mark reader. Conference Series, Society of Photo-Optical Instrumentation Engineers (**SPIE**) Conference Series, vol. 3808, pp. 702–708, 1999.

CIENCIAMAO. Available in www.cienciamao.usp.br. Access in 10.04.15.

DEB, D.; et al. Developing interactive classroom exercises for use with mobile devices to enhance class engagement and problem-solving skills. Frontiers in Education Conference (**FIE**). IEEE, 2014.

FISTEUS, J.A.; et al. Grading Multiple Choice Exams with Low-Cost and Portable Computer-Vision Techniques. **Journal Of Science Education And Technology**, vol. 22(4), pp. 560-571, 2013.

GONZALEZ, R.C.; WOODS, R. **Processamento Digital de Imagens**. São Paulo: Pearson Pentice Hall, 2010.

GORDILLO, A.; et al. Enhancing web-based learning resources with quizzes through an authoring tool and an audience response system. Frontiers in Education Conference (**FIE**). IEEE, 2014.

LUCENA, K.K.T.; et al. MobiMonitor: A mobile app for monitoring distance courses in the Amazon region. Frontiers in Education Conference (**FIE**). IEEE, 2014.

MCTest. Available in http://vision.ufabc.edu.br/MCTest. Access in 24.05.15.

MOLLAH, A.F.; et al. Design of an Optical Character Recognition System for Camera-based Handheld Devices. **International Journal of Computer Science Issues** 8(1), 2011.

NGUYEN, T.D.; et al. Efficient and reliable camera based multiple-choice test grading system**. International Conference on Advanced Technologies for Communications**, 2011.

OMR. Optical Mark Recognition. Available in: www.omrsolutions.com. Access in 10.04.15.

OPENCV. Open Source Computer Vision. Available in http://opencv.org - Access in 08.04.15.

RODRIGUES, J.R.A.; et al. iQuiz: integrated assessment environment to improve Moodle Quiz. Frontiers in Education Conference (**FIE**), IEEE, 2013.

SAIPULLAH, K.M.; et al. Measuring power consumption for image processing on Android smartphone. **American Journal of Applied Sciences**, vol. 9(12), pp. 2052-2057, 2012.

SEN, D.J.; et al. Modern Database Technology Needs Optical Mark Reading. **International Journal of Pharmaceutical and Applied Sciences** 1(2), 56, 2010.

SPADACCINI, A. A Multiple-Choice Test Recognition System based on the Gamera Framework. **Document Image Analysis with the Gamera Framework**, vol. 8, pp. 5-15, 2009.

ZAMPIROLLI, F.A.; QUILICI-GONZALEZ, J.A.; NEVES, R. Automatic Correction of Multiple-Choice Tests using Digital Cameras and Image Processing**.** Workshop de Visão Computacional (**WVC**), Rio de Janeiro, 2013.