Exam Evaluation Using Computer Vision

Verónica Rocha (68809), Miguel Ferreira (72583)

***Resumo* - O trabalho descrito neste relatório consiste na utilização de diferentes métodos e técnicas de Visão Por Computador de modo a criar um programa que permite a leitura, análise e classificação de testes de escolha múltipla com um certo template de modo quase instantâneo.**

***Abstract* - OpenCV tool will be used to evaluate a multiple-choice exam. This involves the detection of a table's edges, corners and lines and, afterwards, retrieve the answer in each grid and detect if it is right or wrong.**

I. INTRODUCTION

The current assignment was developed in the context of the Computer Vision optional subject of the fifth year in Mestrado Integrado em Engenharia de Computadores e Telemática of Universidade de Aveiro.

The main goal of the project is to use the different elements learned in class for problem solution in the area of Computer Vision using the OpenCV [1] technology. In this case in particular, it will allow the exam evaluation of a multiple choice exam almost instantly. This means that it will allow the detection of a table given a certain template and answer detection.

II. EXAM EVALUATION

The project can be divided in two different parts: The detection of the table using its corners, in a way that allows to isolate the table and detect its lines and filled in fields and the detection of the answers given using the zeros present in the table already retrieved.

A. Corner Detection

Initially, QR codes were used in order to identify the edges of the table. To make it less "bulky", some new edges were used to delimit the table and determine its rotation, however, this did not work as expected when the table was not in a vertical or horizontal position.

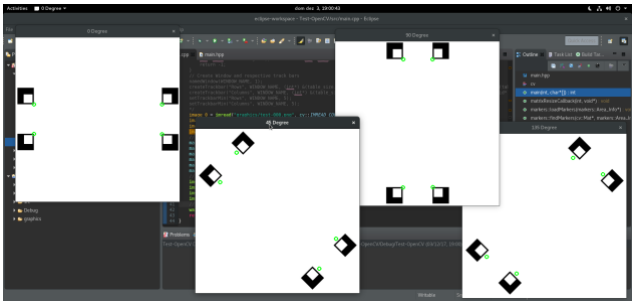

Fig. 1 - Corner detection using template matching

Because the corners of the table were not correctly detected, this resulted in the search for a better solution.

B. Table Template

	a	b	c	d		a	b	c	d		v	f
1					8					1		
2					9					2		
3					10					3		
4					11					4		
5					12					5		
6					13					6		
7					14					7		

Fig. 2 - Multiple Choice Template

The border is composed by bold lines that limit the table and the black square on top is used to know the orientation of the table.

C. Filtering the image

In order to find the table, it is necessary to filter the image. To achieve this, a blur effect is applied, followed by an adaptive threshold.

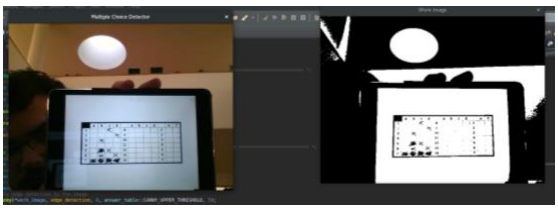

Fig. 3 - Image Filtering

As it is possible to see in Fig. 3, the image is filtered and it is easier to detect a table considering the background is white.

D. Edge Detection

In order to detect the table, it is necessary to detect its edges, the contours and the rectangles.

For the edge detection, the Canny filter is applied to detect the contours of the entire image. Afterwards, these contours are dilated so that the polygons are correctly detected and there is no break on the lines.

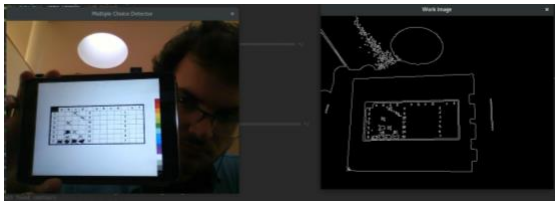

Fig. 4 - Edge Detection

E. Rectangle Detection

The results of the contours are going to be filtered to determine which ones joint in a point. It is possible to retrieve a polygon with 4 corners that is mandatory convex.

After this, it is necessary to verify that the maximum cosine of the corner's angle is 90º or very similar to that.

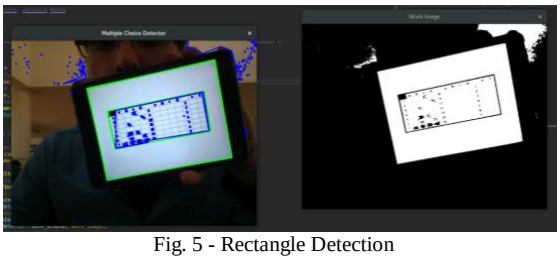

Fig. 5 - Rectangle Detection

F. Table Detection

Comparing all the rectangles found previously, it is possible to retrieve the one with the biggest area that represents the table and crop it out of the image.

The new image is going to consist in the detected table and all the operations will be executed in this cropped part.

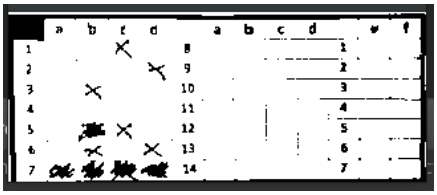

Fig. 6 - Table Detection

The left superior corner of each table is verified. If it is more than 80% black, the current orientation is considered the right one. Otherwise, the rectangle is rotated 90º and the same verification is made. And so on until all the orientations are verified (0º, 90º, 180º, 270º). If there is no mostly black rectangle, then it is discarded and the same process will happen to the next biggest rectangle until the condition is met or there are no rectangles left.

When the table is successfully found, the table, the corners and its structure is stored in a data structure and the flag that indicates if the table was found is set to true.

G. Answer Detection

The columns of the table will be analysed one by one and if it is detected as a column that contains an answer, the results from the lines of that column will be retrieved.

To get these results, for each rectangle, the percentage of black will be retrieved. If this percentage is between 7.5% and 20%, it will be considered an answer and stored as such.

	a	b	c	d		a	b	c	d		v	f
1			X		8					1		
2				X	9					2		
3		X			10					3		
4					11					4		
5		~~X~~	X		12					5		
6		X		X	13					6		
7	~~X~~	~~X~~	~~X~~	~~X~~	14					7		

Fig. 7 - Table with answers

For the example present in figure 7, only the "x" answers will be detected as such. The blank ones, as well as the scratched ones, will be ignored.

H. Answer Correction

For the correction of the answers, there are different values for the correct answers, discounts for the wrong ones and an array with the correct answers.

For each multiple-choice question, a verification will occur in order to determine if there is only one answer or more than that. If there is only one answer, it compares it to the correct answer. If this happens, it circles the answer in the table and update the overall score of the test. If there is more than one answer or the comparison fails, a red cross will be presented in the table and reduces the score according to the wrong choice.

The same happens for the truth or false answers.

I. Interface

There are four windows:

- One that presents the image that is captured using the camera;
- The image with the threshold applied and four track bars to allow to dynamically modify the dimensions of the table;
- The detected table with visual markers for each answer indicating if it correct or incorrect.
- The overall score of the test.

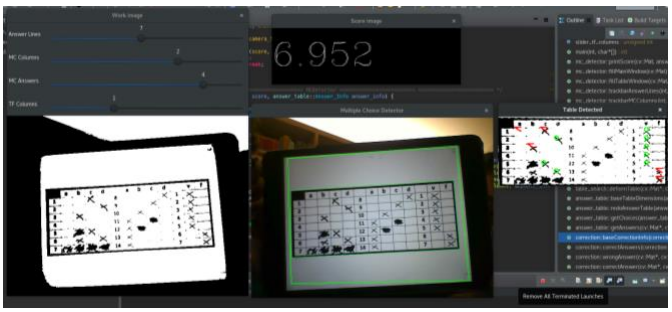

Fig. 8 - Exam Correction and score

III. PROBLEMS AND DECISIONS

Initially, four images were used to represent the corners and make it possible to identify the table, however, this method did not allow to detect if the orientation of the table was in the correct position or if it was upside down.

After presenting the assignment, it was suggested to use QR codes to indicate the orientation and the corners of the table, such as can be seen in the picture presented below.

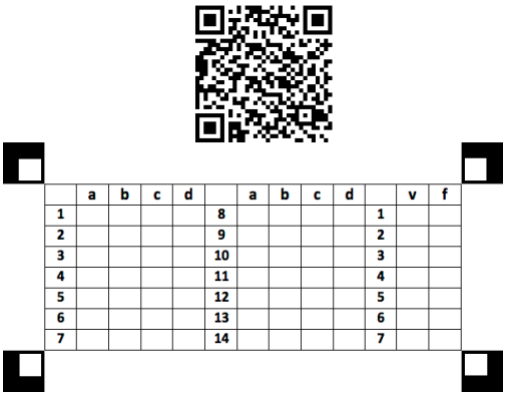

Fig. 9 - Table template with QR code

This, however, did not solve the problem because in addition to the fact that the use of the QR code and the corner markers is excessive and not very practical, it resulted in problems in the detection and cropping of the table.

After much ponderation, this was solved using one simple black rectangle in the superior left corner of the table and thicker table lines, which resulted in the templated presented previously.

Also, there are some problems related to lighting that were not resolved.

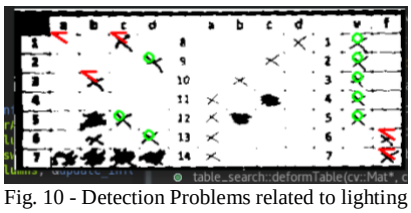

Fig. 10 - Detection Problems related to lighting

In Fig. 11, it is possible to see that the middle columns' answers were not detected. This is a consequence of the lighting that hits in that space in particular.

IV. CONCLUSION

The development of a program that allows to detect and evaluate multiple choice answers just by showing them to a computer camera using OpenCV allowed us to explore and use the concepts that were taught exercised in the practical classes. Also, it resulted in taking different approaches to specific problems that appeared through the project development.

V. REFERENCES

[1] "OpenCV Modules," 18 01 2018. [Online]. Available: https://docs.opencv.org/master/.