University of Bucharest
Faculty of Mathematics and Computer Science
Department of Computer Science

Andrei Dumitriu
MSC in Computer Science
April 2020

Computer Vision - Project 1

Automatic grading of multiple choice tests

The controversial approach

Introduction

In a world of boring projects, the Computer Vision course promises modern, interesting approaches to learning. Not only we get to use interesting techniques on real data, with a real application (although my inside info tells me that the problem is already solved) but we can also apply controversial methods, as you will see in my approaches. The following documentation is a description on my approach at solving the automatic grading of the admission to the Faculty of Mathematics and Computer Science at the University of Bucharest.
Disclaimer: The approach is not good and should not be used in a real-world situation, nor is it a good reference for future students to try it. If you do stumble upon this, take it as an interesting approach, but do try and solve it in a classical way, using the tools openCV provides. I had loads of fun solving this project.

Objective

The goal of this project is to develop an automatic system for grading multiple choice tests completed by candidates during the admission exam for the Computer and Information Technology specialization at the Faculty of Mathematics and Computer Science, University of Bucharest.

Approach

Scenario 1
Initially, I applied line and column detection, but I have moved to a more hardcoded version as it yielded better results. I know that automatic detection can be the better solution, but given that the deadline is here, I will stick to this.

First of all, I apply warp on all images. The warp function is the same as in the laboratory, with the only difference that I used **nfeatures=**10000 instead of 5000. Initially I tried tweaking the function by taking only the first k features (best results were with k between 200 and 500) but I managed to get better results for scenario 2 with the basic function and 10000 features.

I apply warp **twice** on each image, to be sure. It is not needed for scenario 1, but better to be safe than sorry.

For warp, I selected a random image and removed the “X”s and the selected option and variant (“fizica” or “informatica”). I have tried multiple approaches here:

1. Using the full image and this yielded the exact same results, but I didn’t risk it and only took the image from h * 0.36 : h.
2. I removed “raspunsului” from the bottom of the page, thinking that it will be similar to “Raspuns” on each table, but this yielded worse results

Used template is provided with the solution.

After warp, I hardcoded the following steps:

1. Resize **twice** with 0.5. This is due to an initial code structure that has since changed. I tried resizing to 0.25 directly but the results are different. I assume this is due to the resize algorithm and I will leave it as such.
2. Extract the table
3. Grayscale it
4. Place the rectangle where each X should be
5. Take the min of all 4 rectangles on each line
6. I read the option (F or I) and variant (1, 2, 3 or 4) from the filename and compare them to ground truth.

I print the results in a .txt file as instructed.

Scenario 2

Identically to scenario 1. Warping 2 times basically brings us to scenario 1, with the only difference being in how the images are read (different name). It makes no difference if they are rotated or in perspective.

Initially I got 100% on both scenarios (1 and 2, with rotation and perspective) but afterwards I renamed all the files with cv.save and the results went to 86%, then 91% and now 99% on the test data. They are the same resolution but I assume that this is due to different encoding. My only worry is that I overfitted the hardcoded parameters on the training data.

Scenario 3

Again, I apply the same 4 steps from 1 and 2, the only difference being how to select the option and variant when grading

Initially I used a pretrained model from minst but the results were around 86% accurate. For the fun of it, I grade the answers from each paper on each of the 8 variants returning the variant with the highest grade and got a 92.7% accuracy. Since the results are better, I’ll take my chances and stick with this (it’s also more fun). Other combined approaches I did not have time to try:

1. Read the option (F or I) and get the max out of there
2. Use the pre-trained model from minst and only compare the max when the variant is 1 or 4 (this is the main confusion in the minst model). I assume this would have yielded better results but I did not try it.

Scenario 4

I've had loads of fun with it. First of all, it's all a guess.
I apply the scenario 3 algorithm on the images and get a grade (which is most likely wrong). I mapped the frequency of each grade from the test images and I basically print the closest grade from the test set to the one that is computed. If there are 2 grades that are of equal distance, I print the one with a higher frequency.
Ironically, I do not print the grade if it is the guessed one. If the scenario 3 algorithm does guess it correctly by chance, I will print the closest one to the correct guess.

It was fun coding this situation. I hope it guesses a few.