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

Andrei Dumitriu
MSC in Computer Science
April 2020

Computer Vision - Project 2

Snooker, balls and way too many hands over the table

The “I really could’ve used 2 more days to work on it” approach

Introduction

After promising interesting projects and amazing with the 1st project, the Computer Vision course has come with a 2nd project, an analysis on snooker game recordings.
While interesting (and in the end not too hard), unfortunately I did not allocate the required time to ace this project. It is a pity, as after working on it, the needed approaches have become clear.

Objective

The goal of this project is to develop an automatic system that analyses different videos / pictures of a snooker table and solves 4 required tasks

Approach

Scenario 1
First of all, I extract only the table. For this, beside the hsv filtering, I take the longest connected component. This also removes the score table when it is green (something that the hsv does not). After that, I apply a mask that only keeps the white parts. The logic is that I am looking for the “shiny white part on each ball”. It mainly fails either when the balls are too close (I check for the intersection of the bounding boxes) or when there is a hand or something else on the table. I take each detected circle and find out if it is a ball, using template matching. With this, I finally calculate the number of balls. I do not analyze the type of ball, but infer from the snooker rules what balls should be on the table, considering their number. I know that sometimes it can fail (such as immediately after pocketing a white / colored ball), but the approach should work most times.
However, template matching doesn’t work well and the accuracy is low. I initially tried with Hough Circles but that went even worse. The “detect_circles” etc functions are from a far away time, when I used Hough Circles. I changed the approach 1h before submitting and don’t wanna risk deleting anything.

Scenario 2
For this scenario, I calculated where each pocket is located. This proved to be quite the task for the upper pockets, but I was up to it and it was fun. For the middle pockets, I found the point of intersection of the diagonals, and then drew a line on the intersection Y coord, marking the intersection between that line and the vertical ones.
After detecting the pockets, I crop each pocket and do background subtraction on each, using createBackgroundSubtractorMOG2. After recording all the frame differences for each pocket, I sum all the pixels in all frames for each pocket (except 1st and last frame) and choose the one with the highest value. Basically, this approach gives us the pocket with the highest change. This approach is susceptible to the problem when someone moves over the pocket.
After I compute the pocket with the highest value in frame changes, I compare it to a threshold (predefined by me), returning it if it is larger than the threshold. If it is not, I decide that the ball was not pocketed.
Each time I return red, being the most common ball. I didn't have time yet to detect the color. The approach is quite good, but it needs more tweaking to work efficiently. The rectangles could be drawn better and the logic of the movement could be improved. (detect shape of what moves?)

Scenario 3
Not much to say here. I just used the particle filter from the lab, mostly "as-is". I simply enclosed it in a function and instead of roi, I pass the values from the provided coords.

I also tried the approach from scenario 4 (alone and combined with cropping) but particle filtering still yielded better results.

Scenario 4
I apply a hsv mask that leaves only the white points. After applying morphological operations, I select the area with the largest contour and return it. Seems to work decently.

Disclaimer:
Code is written horribly (in a hurry). Haven't written such ugly code since, well, probably since other projects in the Faculty. There are many functions taken from the lab and there is only one function taken from stackoverflow, to compute the intersection of two lines. I found no reason to try and mask that the function is copied, but referenced it as such.

Final Thoughts:
If from the 1st project we learnt that many aspiring students do not really know how to properly draw an "X", from the 2nd project we learnt that snooker players should really lean less over the table (when not needed for a specific move) and should not put their hands over the table so much.

Oh, and for the love of God, TV stations. Stop using the same color as the snooker table for the score table. Someone should make this a rule.

But overall, snooker is probably a game that should be able to be scored automatically only using Computer Vision, as there is enough data available.

References:

1. How do I compute the intersection point of two lines?