Computer Vision Project 2
Video Analysis of Snooker Footage

Matei Bejan, group 407

June 2020

Task 1: Single-frame ball detection & count

We present a combined approach, using 3 detection strategies:

- Detection via BGR thesholding.
- Detection via HSV thresholding.
- Detection of the light refletion on the snooker balls.

Each of those three has its own downsides, but combining them in a cascade fashion, by applying the BGR detection, then the reflection detection and then the HSV detection, leads to an optimal result of 96% accuracy.

The BGR and HSV thresholds have been manually computed by us, using the threshold-picker code provided in the 5^{th} laboratory.

To minimize error, we apply a HSV threshold in order to detect the snooker table, and then crop the bounding box of the table.

Note: In tasks 2 and 3, whenever we refer to "frame", we actually use the bounding box of the snooker table, just like in task 1.

Task 2: Potted ball detection

For this task we parse all the frames in the video and call the detection algorithm from task 1 for each frame. The results, consisting of the colours and positions of balls, are saved in an array.

We check the first element of the array versus the last element to see if the number of balls differ by 1. If yes, then use two heuristics to decide which hole has the ball been potted into:

Heuristic 1: We isolate the "hole areas" (rectangle-shaped areas around the actual hole) by tracing lines relative to the image's width and height and checking whether the ball has entered said area and never returned.

Heuristic 2: In the event that the 1^{st} heuristic fails, we check the position of the cue ball with respect to the position of the potted ball after a certain frame. Based on the position of the cue ball and that of the potted ball, we determine the hole.

Task 3: Single-view ball tracking (cue & coloured)

We have built 2 detectors, for both the cue and the coloured ball.

The cue ball detector applies a binary threshold to the first frame of the video and compares it with the other (thresholded) frames in order to detect the moving object. If that fails, we use a combination of thresholding, erosion and dilation in order to find the largest white object. In this follow-up strategy, we consider a 50-by-50 pixels bounding box around the i^{th} frame in which we search for an object in the $i + 1^{th}$ frame.

The coloured ball detector uses histogram-of-colours matching to compute object similarity. For every $i + 1^{th}$ image, we search for a similar object that is lying in a 50-by-50 pixels bounding box around the detected object of the i^{th} frame.

Task 4: Multiple-view cue ball tracking

We have employed a similar strategy as the one for single-view cue ball tracking.

We apply a very high binary threshold (230-255), erode the frame and search for the largest white object. We manually set an "enlarging" factor for the cue ball bounding box.

Miscellaneous

Regarding the project format, all functions used in a certain task have been placed in said tasks' section in the Jupyter notebook. Task 2 also makes use of procedures defined for task 1.

Please **note** that for tasks 3 and 4, the first line of the output text files is of the following form:

number_of_frames_in_video, -1, -1, -1, -1

Lessons learned

We have tried various methods in this project, among which we have HSV thresholding, template matching, YOLO, non-maxima surpression, different trackers (CSRT, GOTURN, TLD etc.), histogram of colours for detection. However, pretty much all of those listed gave subpar, unsatisfactory results.

It turns out that the simplest approaches - by which we mean thresholding (BGR, HSV, binary etc.) for detection and historgram of colours and background subtraction for tracking - have provided the best results.