Computer Vision & Pattern Recognition

SP 2020-2021

Project

The Late Band

Umberto Cocca - Lorenzo Ferri - Lidia Alecci - Tommaso Vitali

Università della Svizzera italiana,
Via Giuseppe Buffi 13, 6900 Lugano, Switzerland

June 6, 2021

Abstract

The goal is to reconstruct the snooker table and balls from side view starting from a video. Specifically, the tasks are:

1. pre-process video and filter only those frames that show side view;
2. reconstruct camera position;
3. reconstruct ball positions;
4. automatically detect balls;
5. find 2D patterns among the red balls.

Usage

Link repository: https://github.com/okamiRvS/CVPR-project.

All the source files are on Drive (link: https://drive.google.com/drive/folders/1f2SiojF4bl9cYvgPKp8Cj6da0ohv8JRs). Download all the source files and put into the src folder.

For the second part of the Project "tracking_red_balls.py" needs to be run, in order to see the tracking of the red balls in real time on the video "ouput" that it is in Drive.

1 Task 1 - Pre-process video

To solve this task we have seen that for each frame, where there is the snooker table as top view, the camera is totally fixed (so no camera movement). Therefore, we apply a mask in each frames of the video with the goal of reduce the domain of pixel information. Since the top view of the snooker table is composed for the most of green colour, we averaged the amount of colour in the "WSC sample.png" picture to get a threshold and with a slight range flexibility we used it to to extract only the frames where the table is showed as in "WSC sample.png" picture. We've got approximately 28gb of images and for this reason we rendered them as a video (visible in the drive folder). As it can be seen from the 1.1, all the frames we highlighted in yellow are "good"

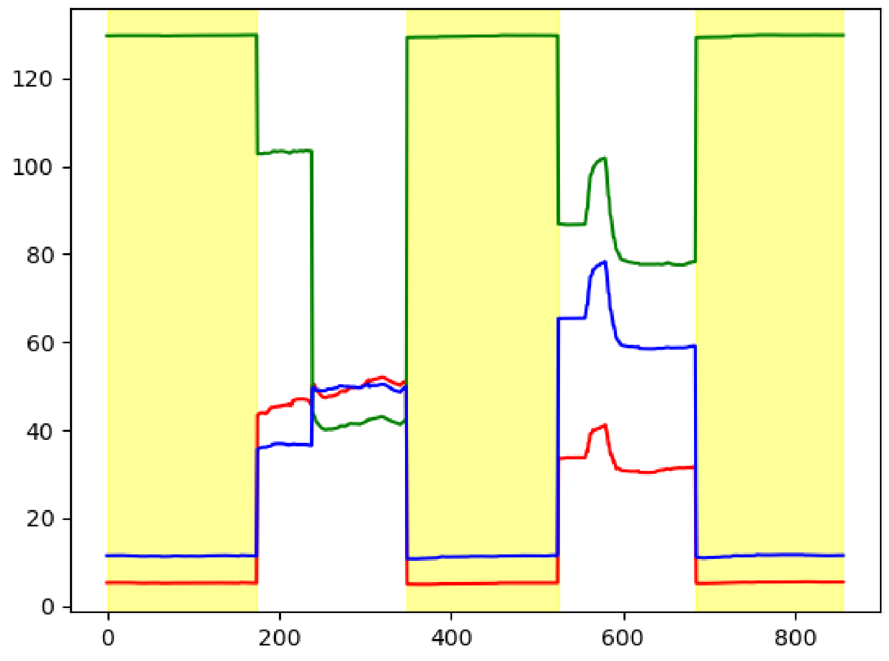


Figure 1.1: This plot show us where bad and good frames are visible in the "WSC_trimmed" as test

frames – the ones we have to extract from the video. The simple approach given by using a threshold on each channel of the RGB image worked well. We used the following thresholds:

- $115 \leq G \leq 135$
- $R < 15$

- $B < 25$

Suitable values for the threshold have been calculated empirically, given the values for some "good" frames and for some "bad" frames.

1.1 Improvements made after Project Part 1

We realized that with only the above constraints we still had some frames that we don't want: the ones when the camera zooms on a part of the green table.
So, in order to recognize and remove these "bad" frames too we updated the constraints as following:

- inside the table mask:
  - $120 \leq G \leq 130$
  - $R < 12$
  - $B < 23$
- outside the mask:
  - $G \leq 55$
  - $R > 90$
  - $B < 50$

In this way, unwanted frames with zoom on the green table were removed.

2 Task 2 - Reconstruct camera position

We detect the 2d points of the inside and outside corners with different approaches.
To detect the outside corner we first of all moved from RGB to HSV channel to take the Hue, then we applied a filter to focus only on green values. Afterwards we applied morphology on the picture, to clean a little bit the area, through erosion and then dilation operator to return at the original dimension. Finally we computed the edges and used them as input for the Hough Line Transform (we threw away the non-local maxima with 10 as threshold). In this way we were able to find the corners.
To find the balls on the table we moved from RGB to YCbCr channel and we used the luminance as input for the Hough Circle Transform.
The inside corners were computed manually. The follow array represents the homogeneous coordinates of the 2d points.

```
[[ 903   55  1]    # top-right-inside corner
 [1026  610  1]    # bottom-right-inside corner
 [ 255  610  1]    # bottom-left-inside corner
 [ 378   55  1]    # top-left-inside corner
 [ 920   37  1]    # top-right-outside corner
 [1056  625  1]    # bottom-right-outside corner
 [ 216  625  1]    # bottom-left-outside corner
 [ 362   37  1]    # top-left-outside corner
 [ 548  143  1]    # yellow ball
 [ 726  143  1]    # green ball
 [ 638  143  1]    # brown ball
 [ 640  288  1]    # blue ball
 [ 638  433  1]    # pink ball
 [ 638  544  1]]   # black ball
```

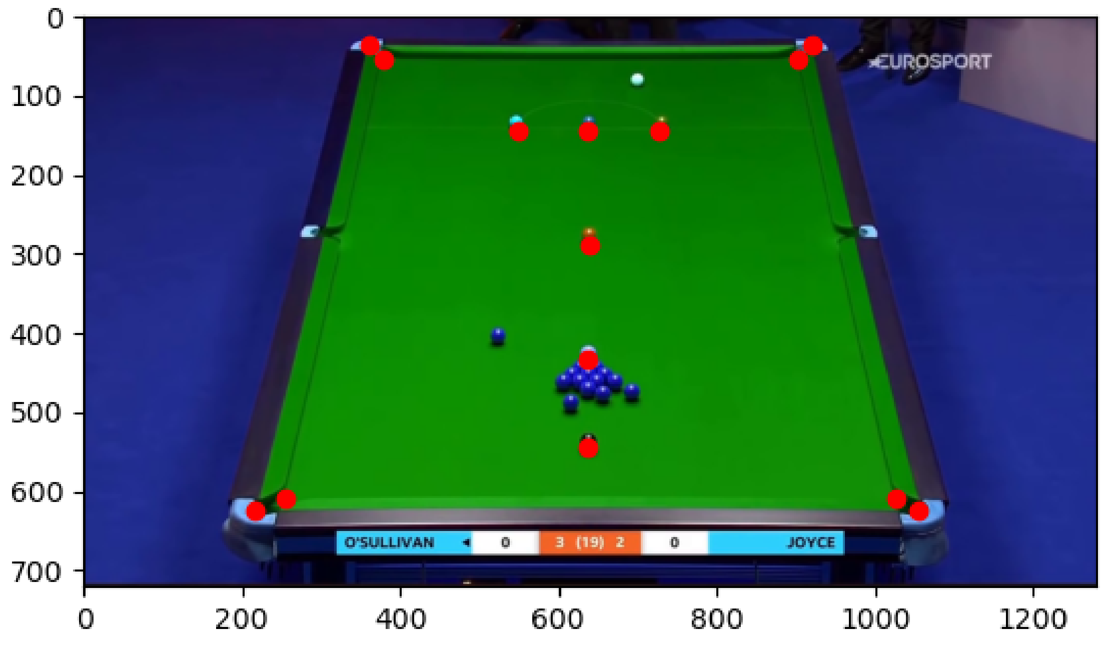


Figure 2.1: This represents the image after we detect the balls and the inside and outside corners of the snooker

Then we mapped each 2d point in 3d homogeneous coordinates using the information of the official dimensions of the snooker table:

```
[[ 0.889   1.7845   0.     1. ]     # top-right-inside corner
 [ 0.889  -1.7845   0.     1. ]     # bottom-right-inside corner
 [-0.889  -1.7845   0.     1. ]     # bottom-left-inside corner
 [-0.889   1.7845   0.     1. ]     # top-left-inside corner
 [ 0.894   1.8345   0.04   1. ]     # top-right-outside corner
 [ 0.894  -1.8345   0.04   1. ]     # bottom-right-outside corner
 [-0.894  -1.8345   0.04   1. ]     # bottom-left-outside corner
 [-0.894   1.8345   0.04   1. ]     # top-left-outside corner
 [-0.292   1.0475   0.     1. ]     # yellow ball
 [ 0.292   1.0475   0.     1. ]     # green ball
 [ 0.      1.0475   0.     1. ]     # brown ball
 [ 0.      0.       0.     1. ]     # blue ball
 [ 0.     -0.89225  0.     1. ]     # pink ball
 [ 0.     -1.4605   0.     1. ]]    # black ball
```

With these points we computed the camera matrix P that describes how a camera maps world points (in 3d) to image points (in 2d).

3 Task 3 and Task 4 - Reconstruct ball positions and automatically detect balls

To find the position of the balls we initially searched for the light reflected on the balls using the following constraints on the HSV image:

- $H < 120$
- $S > 220$

Then, we use the method `cv2.connectedComponentsWithStats` on the previous resulting image (only in the space of the snooker table) which gave us for each connected components the centroid and some addictive stats. This allows us to take only the connected components with an area ≤ 40 – in order to match only the white reflection on the balls, the one caused by the overhead lights. By some experiments carried out, the connected components above this threshold are not what we are searching for.
Now, we have to select the position of only red balls – the others must be excluded. To do that we take the pixels under the end of that light portion and we search for a red color portion (5x5) that match the following conditions:

- average of $R > 110$
- average of $G < 55$
- average of $B < 55$

In this way we are able to obtain the position of the red balls and we are able to follow them moving. In the following image we show the result obtainable by running the `"tracking_red_balls.py"` file.

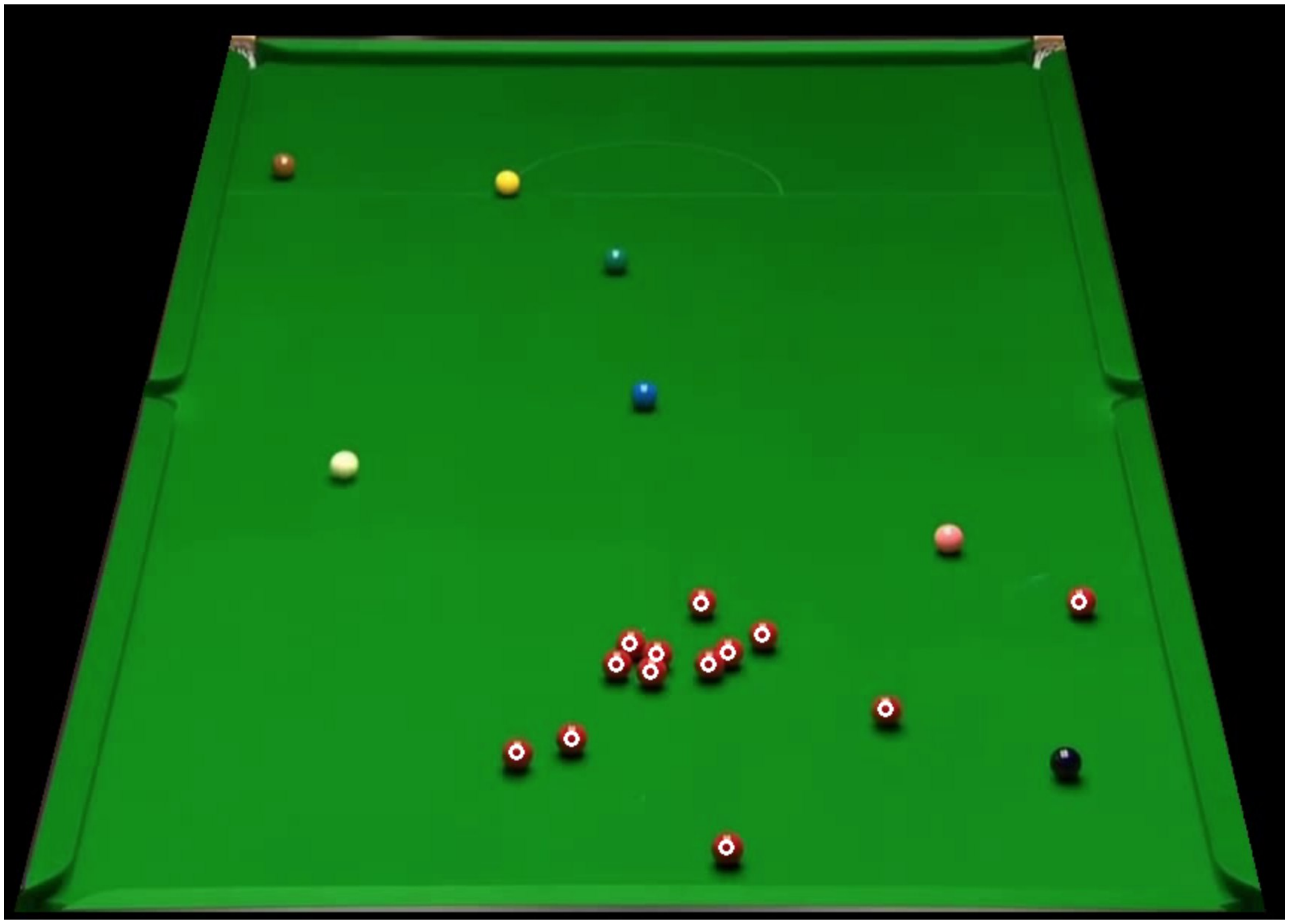


Figure 3.1: This represents the frame of the videos obtained when all red balls are detected

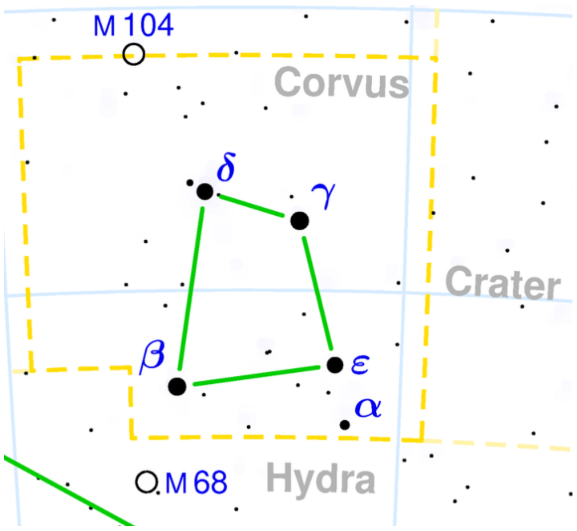
4 Task 5 - Find 2D patterns among the red balls

We chose "Corvus" (Figure 4.1) as the constellation to match with the red balls.

In order to speed up the algorithm we decided to take only the frames with four balls and we saved the following information of these frames in a csv file: number of frame and 2D coordinate for each of the four balls.
We took the coordinate of the constellation from an image and we centered the coordinates taking the barycenter, in order to have the coordinate computed from the barycenter.
To find a 2D similarity transformation (uniform scaling + rotation + translation) we took:

- for the rotation matrix all 45° angles and each values between -1 and 1.5, with a step of 1/4, for xs and ys;
- for the scaling matrix values between 0.5 and 2, with a step of 1/4, for the parameters cx and cy.

After some trials the values above seemed reasonable for our problem.
To speed up all the process we chose to implement a multiprocessing solution, that based on the number of cores splits the csv file with all the frames that contain 4 balls and run the same method in parallel.
We valuate as good solution the one with the minimum euclidean distance, the best one we obtained has min error 211 (was the one shown in the figure 4.2).



[H]

Figure 4.1: Corvus constellation.

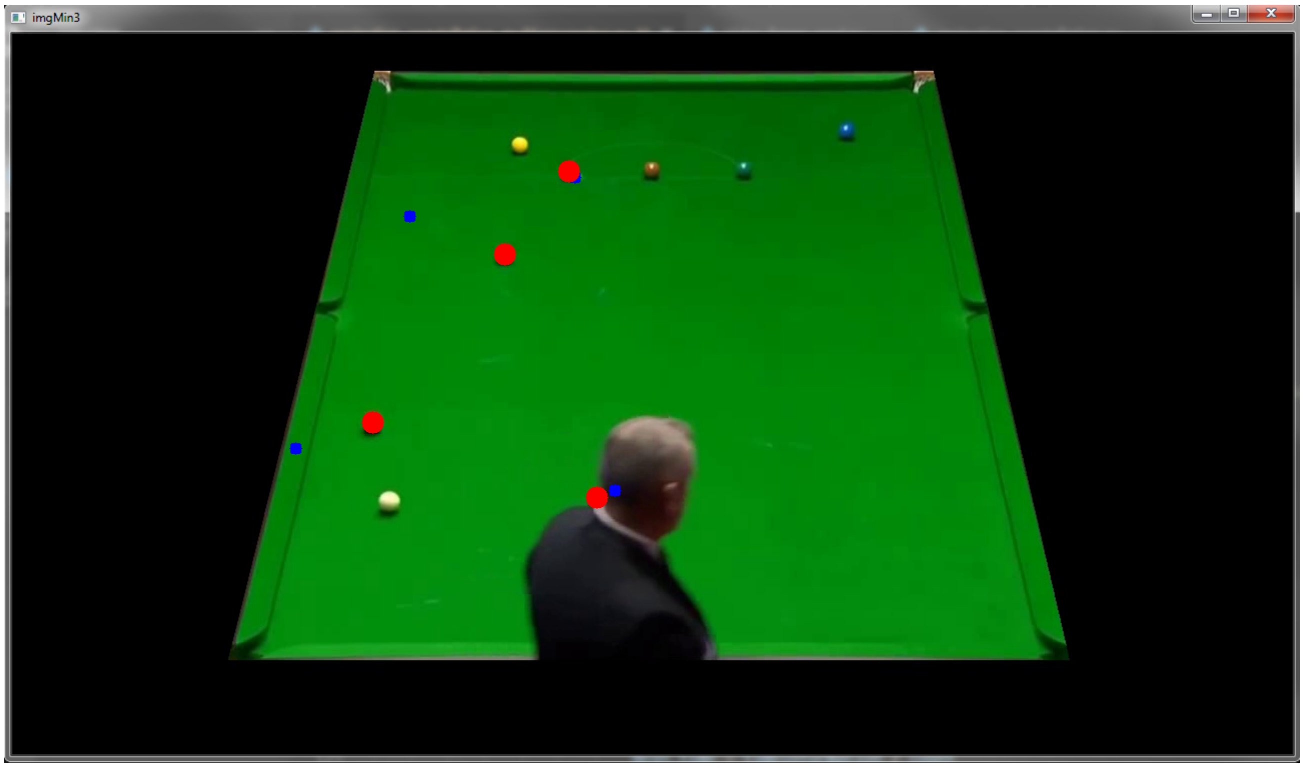


Figure 4.2: Best result in finding constellation Corvus among red balls. The red points are the red balls while the blue points are the stars projected on a table.