COMPUTER VISION

Sport video analysis for billiard matches

Final project - June + July 2024

Introduction

Sport video analytics relies on computer vision systems to identify and track objects of interest in input videos and generate semantic information. Considering billiard matches, such information includes real-time monitoring of the playing field, automated scoring and reconstructing the trajectories of the balls after a strike. A billiard practitioner could then exploit this data to analyze the movements and techniques used in each shot to improve his/her skills in the game.

The goal of this project is to develop a computer vision system for analyzing video footage of various "Eight Ball" billiard game events. Eight Ball is a call shot game played with a cue ball (white ball) and fifteen object balls, numbered 1 through 15. One player must pocket balls of the group numbered 1 through 7 (solid colors), while the other player targets balls numbered 9 thru 15 (stripes); the black ball numbered as 8 ("8-ball") has a special role and must be pocketed at last. The player pocketing all his/her group first and then pocketing the 8-ball wins the match.

The required computer vision system must provide high-level information about the status of the match (e.g., ball position) for each frame of the video; this high level information should be displayed as a 2D top-view minimap, as shown in Figure 1.

In more detail, for each frame of the input video the system to be developed should be able to:

1. Recognize and localize all the balls inside the playing field, distinguishing them based on their category (1-the white "cue ball", 2-the black "8-ball", 3-balls with solid colors, 4-balls with stripes);
2. Detect all the main lines (boundaries) of the playing field;
3. Segment the area inside the playing field boundaries detected in point 2 into the following categories: 1-the white "cue ball", 2-the black "8-ball", 3-balls with solid colors, 4-balls with stripes, 5-playing field;
4. Represent the current state of the game in a 2D top-view visualization map, to be updated at each new frame with the current ball positions and the trajectory of each ball that is moving.

Figure 1: Example of the system output for a few frames of an input video. On the left, the first frame of the video with a 2D minimap representing the initial state of the playing field from a top-view. On the right, a frame of the video after the player's shot with the minimap showing the updated state of the game (i.e., ball positions) and the ball trajectories. The minimap shows each ball with a different color based on its category (cue ball, solid, stripes).

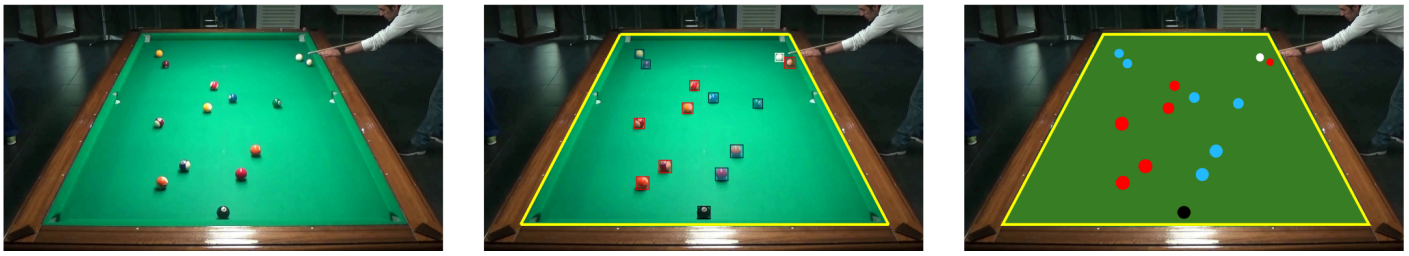

Figure 2: Example of the system outputs. From left to right: i) input image from a video; ii) playing field marking localization (yellow lines) and bounding boxes for ball detection; iii) balls and playing field segmentation. Each ball type is represented in a different color.

As case studies for the development of the required system, various game recordings are considered, differing in (i) the type of camera framing, (ii) the number of balls in play, and (iii) the color of the pool table. The system to be developed should be robust to all such conditions. In particular, it should recognize all the main elements (playing field and balls) albeit different camera perspectives and table colors, and it should ignore any person (e.g., players, referees, spectators) around the billiard table.

In developing the project, the following simplifications can be made:

- to define the playing field, you can use the lines belonging to the upper parts of the top rails of the table, that is the parts where the cloth covering the playing field surface is attached to the wooden structure of the table (as depicted by the yellow lines in the central image in Figure 2);
- you can assume that the pose of the camera does not change during a single video clip, so the table remains in the same position in the image for all frames of the same video;
- to compute the trajectories of the balls moving after a strike, tracking techniques from the OpenCV library[1] can be used to re-identify all balls that have changed position between two consecutive frames.

To assess the robustness of your system a benchmark dataset with annotations is provided at the following link:

https://drive.google.com/drive/folders/1dzNrhDpc2DXRqmQgbO5l2WMjzfhMdxVn?usp=sharing

The benchmark dataset consists of 10 different video clips of individual billiard shots extracted from 4 different match videos on YouTube. For each video clip, two frames were selected corresponding to the initial situation before the shot (i.e., “frame_first.png”) and the situation after the shot (i.e., “frame_last.png”).

Such images have been annotated with bounding boxes and segmentation masks, and will be used to evaluate the performance of the computer vision system developed. As classes of interests, the following categories and labels have been considered: background (label id = 0); white cue ball (label id = 1); black “8-ball” (label id = 2); balls with solid colors (label id = 3); balls with stripes (label id = 4); playing field (label id = 5).

[1] https://docs.opencv.org/3.4/d9/df8/group__tracking.html

Performance measurement

For measuring the system performance, you should have a look and understand the following metrics:

- The mean Average Precision (mAP) - https://learnopencv.com/mean-average-precision-map-object-detection-model-evaluation-metric/
- The mean Intersection over Union (mIoU) - https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2

Such metrics shall be used to evaluate the billiard analysis system as follows:

- For ball localization, the mean Average Precision (mAP) calculated at IoU threshold 0.5;
- For balls and playing field segmentation, the mean Intersection over Union (mIoU) metric, that is the average of the IoU computed for each class (background, white ball, black ball, solid color, striped and playing field).

The metrics mentioned above need to compare the output of your system against the ground truth, namely what is considered "the truth" for each output image. Ground truth annotations are already included on the dataset provided for the final evaluation of your system, with respect to the first and the last frame of each video clip. **Please note that the ground truth cannot be used as an input of your algorithm**.

The ground truth is stored in a set of files, one file for each input image. The information representing the ground truth (e.g., the rectangle enclosing the ball or the pixels belonging to each ball in the image) is expressed in such files based on the following standard:

- For ball detection: every ball is found inside a rectangle defined by 5 parameters [x, y, width, height, ball category ID], where (x,y) are the top-left corner coordinates and width and height are the bounding box main dimensions; the 5^{th} parameter is the label ID representing the type of the ball; such parameters are listed in a row, one ball per row;
- For ball and playing field segmentation: a grayscale mask where each pixel is assigned the corresponding category ID (background, white ball, black ball, solid color, striped and playing field).

Feel free to add more test images and videos, taken from the internet or acquired using your camera. In case you use additional images for the development of your system, you must include those images and related annotations (or a link to them) in your final delivery. If you use additional images, you are free to define your own standard (which you should describe in the report). If you agree with other groups to share the ground truth collection, be sure to share the standard. The organization of the ground truth collection is completely free; if you wish, you can use the dedicated section in the moodle forum.

Project delivery

The project must be developed in C++ with the OpenCV library **only**. **The project cannot be developed using deep learning; in particular you are not allowed to rely on state-of-the-art object detector or people detector based on deep learning**.

You need to deliver your project including:

- All the source code (C++), where each source file must contain the name of the main group member developer - **one author per file is allowed**; you should check that **the code compiles on the Virtual Lab**, that is considered the official building environment;
- CMake configuration files (the use of CMake is mandatory);
- A report (no page limit) presenting your approach and **the performance measurement on the dataset provided and linked above**. You shall report **the metrics and the output images** for every element in the dataset (see details below);
- The **output videos** of your system for all the clip videos in the provided dataset, showing the 2D top-view superimposed on the input video as shown in Figure 1.

In particular, **you should include in your report** both quantitative and qualitative results of the performance of your system for each video clip in the provided benchmark dataset:

- for localization and segmentation tasks, report the output images (bounding boxes and segmentation masks) of your system on each pair of “frame first” and “frame last” images, together with all the requested metrics values (mAP, mIoU) computed with respect to the provided ground truth;
- for 2D visualization and ball trajectory, report the image of the final 2D top-view obtained for each video clip representing the ball position in the last frame and their trajectories from the start of the video.

When delivering your project, you should **clearly identify** the contribution of each member in terms of ideas, implementation, tests and performance measurement. You can organize the work as you prefer: you are not forced to assign one specific step to each group member. Please also include **the number of working hours** per person in the report. This is needed for a monitoring on our side of the effort requested – the evaluation will not depend at all on the number of working hours, but on the quality of the result. **Any requested detail that is missing will result in a penalty.**

It is not allowed to include code that was not written by any of the group members - this includes ChatGPT & similar tools.