A Computer Vision System for 9-Ball Pool

Pranjal Vachaspati
Massachusetts Institute of Technology
pranjal@mit.edu

Abstract

We develop a system to extract semantic data about games of 9-ball pool from online videos. We consider the problems of rectifying video from various camera angles, discarding frames that are focused on players instead of the table, and calculating positions and velocities of balls. We also examine the possibility of applying constraints from physics and game rules to improve results.

1. Introduction

The family of pool games seems especially suited to computer vision analysis, since the visual parameters are highly constrained. Balls of known shape and color move along a background of uniform shape and color. At least in theory, this makes following balls relatively simple. In fact, several systems[4][1] have been created which mount a camera on the user's head or above the pool table and advise the user on the best shot to take.

The goal of our system was slightly different, and presented unique challenges. Instead of providing information in a real-time environment to players, we tried to analyze raw 1080p HD YouTube footage of televised matches, as in [2]. Unlike real-time AR systems, the televised matches include overlays that show the score, instant replays, and strange camera angles that rapidly cut between one another. Thus, aligning the table and discarding frames that don't show the table are particularly important tasks. The system was written in Python using the OpenCV[3] library.

Finally, we considered applying additional constraints to improve the results of the vision algorithms based on the physics and rules of the game.

2. Alignment

A number of camera angles were used by the broadcasters in filming the pool games that we analyzed. A few examples are seen in Figure 1. Ultimately, we decided to use all the perspectives that showed the entire board, but not the ones that zoomed in on just a few balls.

Figure 2. The pocket images were too low-resolution and generic to be useful for SURF keypoint matching

2.1. Image Keypoints

Our intial strategy was to use image keypoints using the SURF feature detector to get the locations of the pockets and the orientation of the board. This, however, proved to be more difficult than expected. The pocket images used for training the detector are seen in Figure 2. Unfortunately, even with 1080p video, these images did not contain enough information to be useful. We found that the SURF keypoints detected in these images showed up at too many places in the test image, and that the keypoints often did not appear in the perspective-transformed boards, as seen in the matching in Figure 3.

Since this model failed because of the symmetry of the board and the specific constraints of the problem, we decided to find methods that exploited these properties instead of being vulnerable to them.

2.2. Color Countours

We quickly noticed, as did the creators of the other systems we examined [1][4], that the pool table is a very uniform color, up to a lighting gradient. We also found that the hue component of the HSL color space was independent of the lighting conditions, making filtering this a very reliable way to determine the pool area. We also found that while the luminosity of the table varied to some degree, it was

Figure 1. The broadcast games use a variety of camera angles.

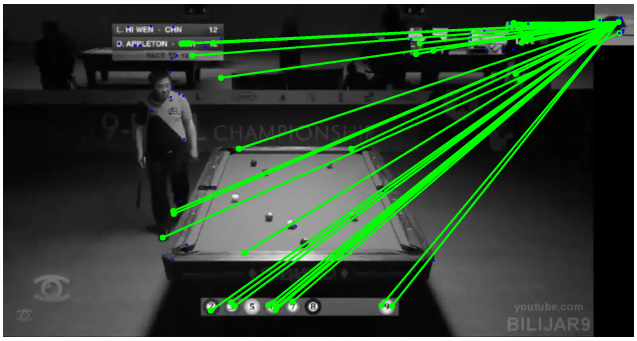

Figure 3. The SURF keypoints in the pockets did not reliably find pockets in the scene

Figure 4. Filtering on hue and luminosity isolates the board quite well for some images.

more brightly lit than much of the background. The first step was therefore to mask out sections of the image that did not have the right hue or had too low luminosity. This works very well on the example image in Figure 4. In general, this is the case; however, occasionally a few pixels of some other object will show up, and some parts of the board are always excluded by this analysis, especially if they are occluded by a ball or player.

Choosing the largest connected component in this image always gave the correct board image. To eliminate the effect of occluding players, we took the convex hull of the component, as seen in Figure 5

However, this still does not provide enough information to align the picture. Because the corners of the pool table have pockets, the corners of this image are very rounded and running a corner detector on the image gives poor results, either aligning the image poorly or choosing multiple corners on the same physical corner. We run a Hough transform on the convex hull image to resolve a set of lines from it, and cluster these into estimates of the top, bottom, left, and right sides of the table (Figure 6). We then use the average of each group as the sides of the table, and input the intersections into a homography that aligns the table, giving a result like Figure 7.

We discard bad images at several points in this process. First, we discard when the largest connected component with the correct hue and luminosity is too small. This eliminates most images of people. Then, we eliminate images

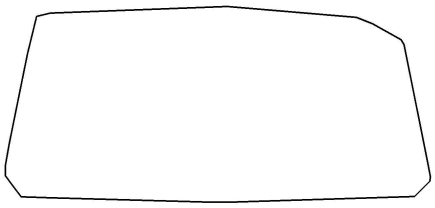

Figure 5. Taking the convex hull of an image reduces the effect of occlusions

where all four corners of the aligned rectangle are not on the screen. This eliminates most of the "zoomed-in" shots, like the third image in Figure 1.

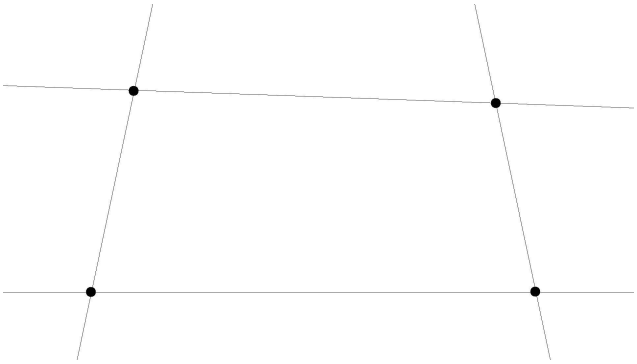

Figure 6. A Hough transform finds four sides of the table

Figure 7. A simple homography finally aligns the image

This approach worked quite well. Of a random sample of 50 images, all 32 images of the pool table were detected as such. 17 of the 18 images of people or zoomed in shots were discarded, and one zoomed-in shot was accidentally aligned (Figure 8). However, it should be fairly easy to detect many of these in the future by ensuring that most of the pixels in the output image have the correct hue.

A larger problem is misalignments. This manifests as the long axis of the pool table being transformed onto the short axis of the image. Currently, our system is set up to only handle the most common case, and as a result, four of the 32 aligned images were misaligned in this way. This is a slightly harder problem to fix, because the only way to find the longer side is to know some physical dimension of the table. The easiest dimension to detect would seem to be the distance between pockets; however, the difficulty of detecting pockets remains. This is a serious issue, and needs to be resolved for this system to be useful.

3. Tracking

Once extraneous regions of the frame are removed and the table is aligned, tracking the balls becomes easy. Again, we worked in HSL space. We used the same mask used for alignment to mask out the table and only retain the balls and edge of the table. Empirically, we found that the sum of the hue and luminosity provided a good value for a Canny detector and Hough transform to detect circles. We identified

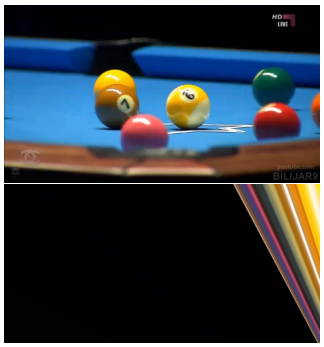

Figure 8. A small portion of zoomed in shots are mistakenly aligned. However, detecting these should be fairly easy.

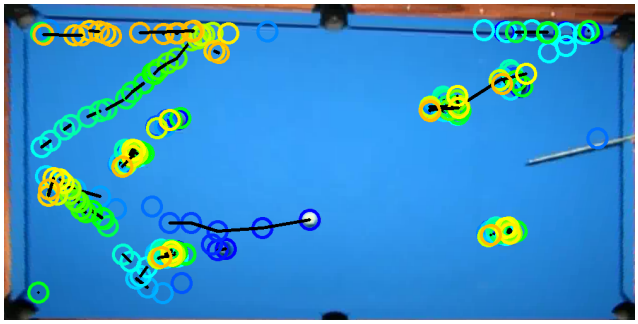

Figure 9. The system is able to recover the ball paths with some accuracy. Time is represented by circle color; balls identified in the same frame have the same color. Black lines identify the measured paths of individual balls. This section includes the collision of the white cue ball with the blue ball in the bottom left. The blue ball bounces off the bottom edge and comes to rest near the left edge, while the cue ball deflects upwards after hitting the blue ball, bounces off the left edge, and comes to rest near the top edge.

balls by taking the mean value of the hue and luminosity inside the detected circle. This was a fairly noisy process, and many balls were detected as identical that were actually distinct. It may end up being more correct to simply use RGB values to identify the balls. An example run at 10fps over four seconds is given in Figure 9.

Some jitter in the alignment is clearly present, since the non-moving balls appear to move quite a bit. We discuss methods to mitigate this in Section 5.

4. Performance

While the system is reasonably performant, it does not run in real time on 1920x1080 video, which is the ultimate goal. On a recent dual core Intel Core i5 system, a single frame takes roughly 0.75 seconds to align and 0.75 seconds to find the balls. Most of the work in alignment goes towards finding the corners in the Hough transform, and most of the work in finding the balls goes towards the Hough circle transform, which is much slower than the line transform. In addition to improving these bottlenecks, the code

could be parallelized by running separate frames in individual threads to get a close to linear speedup.

5. Further Work

While we get reasonable results, especially for table alignment, there remains a significant gap between the necessary performance of the system and its current effectiveness, especially for ball tracking.

5.1. Applying Physics Constraints

We conjecture that the greatest enhancement to our system can come from analyzing the physics of the game and assigning a higher prior probability to ball locations that would result naturally from previous ball velocities. Currently, the only way this is implemented is by eliminating percieved ball motions with unphysical velocities. However, most of the infrastructure to add more effective filtering on this is in place, and should be applied. In fact, once this feature exists, the system could become a powerful tool by simulating the effects of possible shots, and determining how well a player is performing compared to the best he could be performing.

In addition, these physics constraints could be used to improve the alignment process. Since the cue ball always has to be the first ball to move, and balls cannot move without first being hit, we can assume that most of the balls remain stationary. We can propagate this information back into the alignment procedure and line those balls up with the same balls from the previous frame with an additional homography.

6. Conclusion

We found strong evidence that broadcast videos of 9-ball pool can be well analyzed by a computer vision system. We successfully aligned images of pool tables and discard images of other things, and with some success tracked the motion of balls across frames. Computer vision is used with great success in some sports, like the line cameras in tennis, and would be an excellent addition to the sport of 9-ball pool.

References

[1] J. Alderman, M. Kapoor, Y. Nakhimovsky, and C. Cope. Bankshot: A visualization aid for the game of pool.

[2] Billjar9. `http://www.youtube.com/watch?v=YLXYI_t5Ydk`.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] T. Jebara, C. Eyster, J. Weaver, T. Starner, and A. Pentland. Stochasticks: Augmenting the billiards experience with probabilistic vision and wearable computers. In *Proceedings of the 1st IEEE International Symposium on Wearable Computers*, ISWC '97, pages 138–, Washington, DC, USA, 1997. IEEE Computer Society.