Pong-ball Tracking: A Low Cost Approach

Batu Inal, Charles Sie, Carl Kershaw
EECS 442, Fall 2014

In this paper, we present a low cost approach to tracking projectile motion, more specifically a ping-pong ball thrown into the canonical Solo Cup, observed by handheld camera. This project represents the first step towards integrating computers into this social environment. Experimental results are promising, we were able to correctly predict make/miss 83% of the time; however additional robustness over varied conditions are necessary.

I. Introduction

Computers influence our social lives every day, but one dimension of our everyday interaction computers have yet to influence is partying. (Beer) Pong is a pivotal part in many peoples party experience that computers fail to perceive. The pong experience could be enhanced by improving the computer influence to the person playing it using interactive feedback.

The work explored in this project is a critical first step in bringing computers into the pong fold; tracking a ping pong ball and predicting whether it will land in a solo cup is the basis of pong spectating. Imagine watching a game of pong and there is an application to track ball movement to predict the result of a shot before anyone sees it. The basic idea of this work can be expanded to improve player's performance by giving suggestion to improve player's accuracy and precision.

II.a Previous work

Previous work in this scope is very limited; the majority of trajectory estimation involves additional sensors and financial investment.

If we wanted a perfect solution to this problem, any number of commercial motion capture systems exist that could track specially made pong balls at 100-1000 fps, allowing for a fit and prediction very quickly. Unfortunately, these systems are significantly more expensive than a handheld camera, while being more intrusive setups. Both of these factors would intrude on the social atmosphere eliminating the desire to play pong.

II.b Contributions

Our initial attempt in this field was using a point-shoot handheld camera that has dual lenses and synchronized CCD sensors to give stereo pairs. However, sufficiently accurate triangulation using the two cameras was not possible. The baseline of the two CCDs was very small relative to the distance to the pong-ball, leading to extremely noisy disparity measurements. We abandoned the stereo approach to estimate the trajectory and depth due to our limited time and available equipment. Thus, we decided to work with 2-D approach of estimating the trajectory. In this approach, the information on the depth of field is impossible. Thus, we are only analyzing the movement of the ball in 2 dimensions.

The trajectory of the ball in 2 dimensions can be modeled using the projectile motion equation. This method is often used to predict the trajectory of an object with known velocity and direction. The motion of the moving pong ball can be divided into two parts, horizontal and vertical motion. The figure 1 below visualizes the projectile motion in two direction of motion and the combination of both motions would yield the final projectile motion.

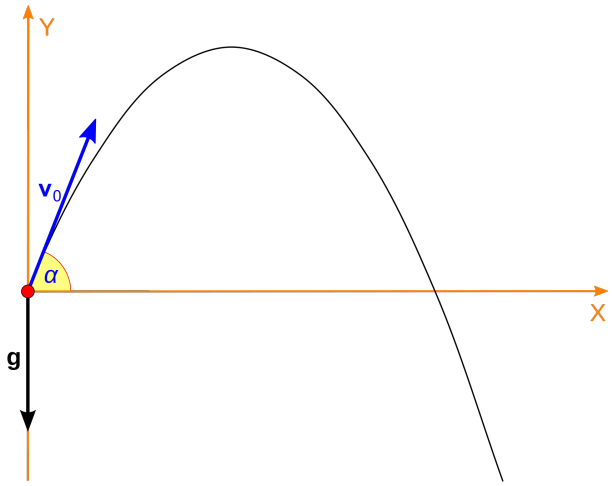

Figure 1: A simplification of projectile model in 2-D

In the model, the motion is calculated under an ideal condition. This means that the wind resistance that might slow down the ball during the projectile is omitted. The predicted final position of the ball would be farther than the actual final position of the ball. Thus, another additional information of the wind resistance is needed to give a better approximation on the trajectory. The method used in this project tries to simply estimate the final position of the ball using all information gathered.

Our current method is significant primarily because of it's simplicity; our system is a single handheld camera, providing low resolution, low frame rate video. This

approach utilizes a novel method to utilize motion blur; by extracting several pose estimates from every frame, our system can fit trajectories earlier, and with higher confidence. Additionally, we leverage knowledge about the target (the Solo Cup) to increase the accuracy of predictions.

III.a Technical Summary

Our project can be separated into three general parts, ball detection, ball tracking/estimation, and cup detection. Detection utilizes prior knowledge about the ball and cup colors, as well as their geometries. The primary challenges stem from the significant motion blur incurred by the fast moving ball, and the unobservable nature of the white top edge of the solo cup in most environments. Both of these challenges drove us to extract more information from the scene, ultimately improving our solution. Various models and fitting methods for ball trajectories were explored, but the preferred method is a least squares fit over a fourth order polynomial that omits the third order term.

Our methods are comprised of several options that offer comparable results; much of this comparison is moved to the Experiments section to allow for a sane process description.

III.b Technical Details

Below is a walkthrough of our trajectory estimation and tracking process, which occurs for every frame.

1. We first convert the image to HSV space, and filter based on the known color of the pong ball.
2. Using connected components analysis/naive blob detection; we filter for detected size, as the conversion to HSV introduces a significant amount of high frequency noise.
3. Using the selected area of matching pixels, we extract ball centers, either by:
    a. taking the centroid of the matched points (yielding one point per frame), or
    b. by convolving the matched region with a Gaussian of a sigma appropriate for the ball at the given resolution, and looking for points within some threshold of the observed maximum.
4. Once sufficient ball centers have been observed, we fit a model to the data
5. Using this model and the cup estimate detailed below, we estimate how close the ball will come to entering the cup, yielding a Make/Miss verdict

After tracking these judgments through the video, and comparing to the actual results, we can ascertain system accuracy.

Below is a walkthrough of our cup pose estimation.

1. Similar to above, convert the queried frame to HSV, and filtered using known "Red Solo Cup" values.
2. Connected components analysis is again used to filter by size, however this time we retain the bounding box.
3. Using this bounding box, we compare the observed aspect ratio to the known ratio of a Solo Cup, to estimate our viewing angle.
4. Using this viewing angle and the cup's current width in pixels, we adjust the estimated top-corners of the cup in pixel space to more closely align with the white lip of the cup.

IV. Experiments

Ball Detection

To detect the ball in a frame, we converted the frame from RGB to HSV, and then filtered for points with the correct hue, and high saturation. After we have this binary image of ball color/not ball, we group connected pixels into blobs. These blobs are then filtered to remove elements that are too big (other orange objects) or too small (pixel noise, from HSV conversion). We then take the largest remaining blob to the ball (if there is one). We developed three methods for extracting ball locations from the matched blob: First and more simply, taking the centroid of the matched points provides one point per frame that is fairly resilient to noise. Second, we return all matching ball points, a method that returns many points per frame, at the expense of additional noise. Finally, the best method we use is convolving the matched points with a Gaussian such that ball centers have a high response, while other points on the ball do not. Figure 2 shows all three methods, showing fitting noise increases, as more points are included.

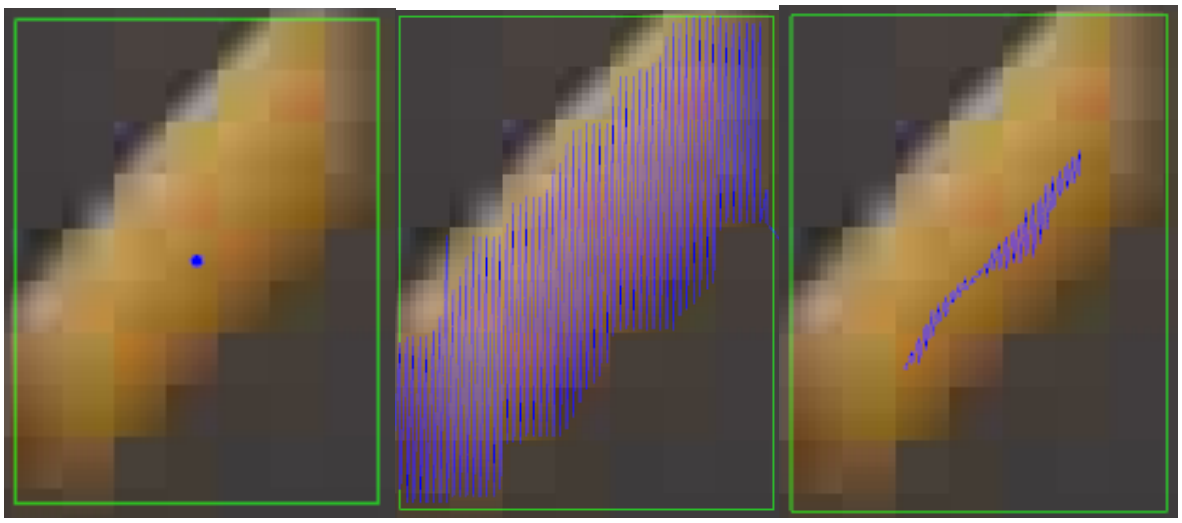

Figure 2: A comparison of ball detection methods.

Trajectory Fitting

We used either least squares or RANSAC to fit the center points found above to models we explored. Traditional least squares did not lend itself to the all-matching-points method of ball detection, as the additional points just increased noise. To combat this, we implemented RANSAC over the points, which also proved unstable in the early fitting stages, see figure 3.

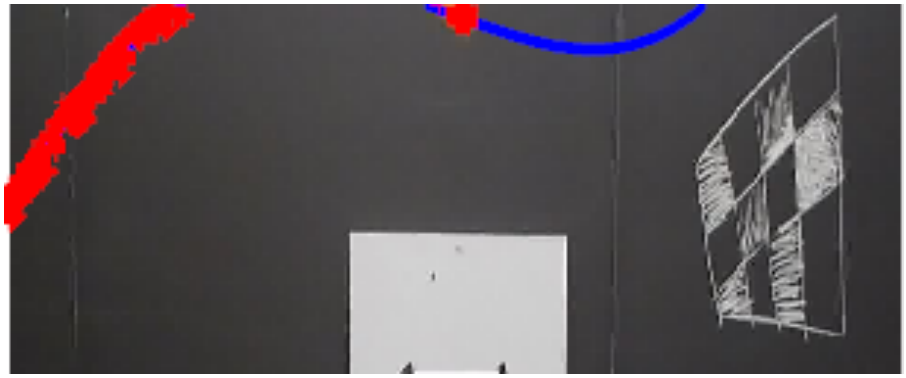

Figure 3: An acute RANSAC failure.

Trajectory Model

We used naive polynomial models to fit the ball's trajectory; to validate each model, we fit to the entire trajectory of the ball, examining the residual error. Simple trajectories should be second-order, however pong balls experience significant drag, so higher order terms are necessary; depending on which model we use for drag, either a third or fourth order term should be added, so we examined both, as well as a full fourth order polynomial (including the third order

term). Figure 4 is an overview of the fit quality, in this particular shot, the ball hits inside the cup.

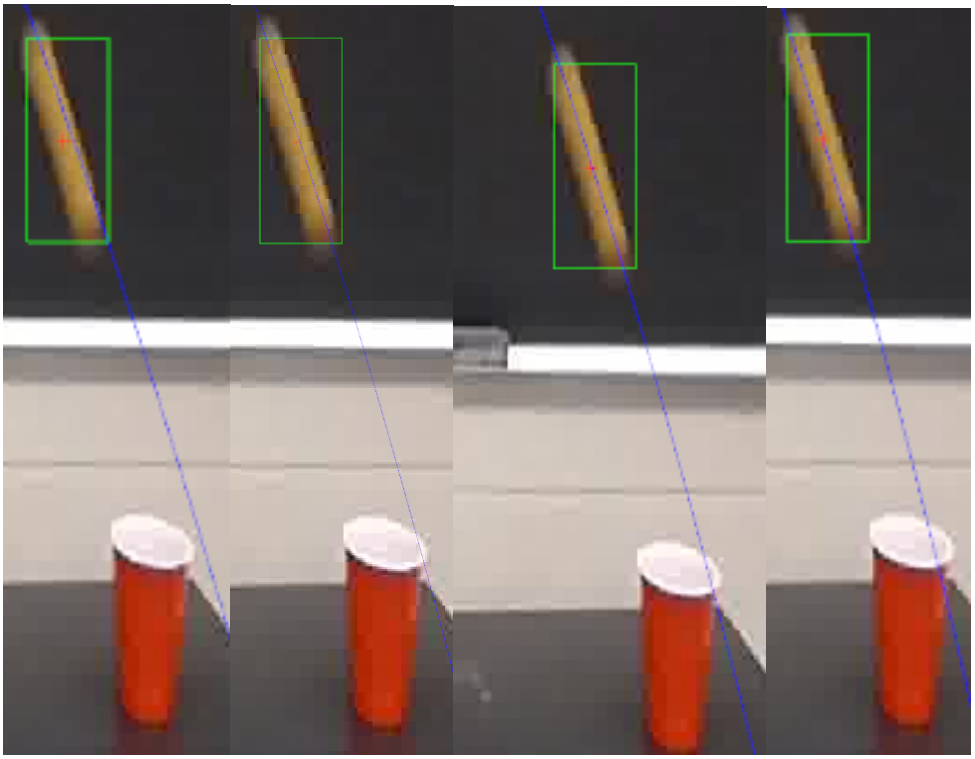

Figure 4: Overview of fitting methods, in the order x2, x3, x4, x3 & x4

Table one gives an overview of fitting accuracies over several detection methods and motion models.

Model:	x^2	x^3	x^4	$x^3 + x^4$
Centroid:	1004	142	72.8	4.6
Multicenters (Normed):	72	12	8.6	5.4

Table 1: R^2 fitting accuracy over models and location methods.

With both methods, the $x^3 + x^4$ model is the best fit, with x^4 outperforming x^3.

Cup Detection

Cup detection is fairly similar to ball detection, but we take the top corners of the bounding box as the cup edge. Unfortunately, the white section of the cup is often missed, so we developed an adjustment based on the observed aspect ratio. Figure 5 is a prime example, green is the naive bounding box, and while blue is the

adjusted cup edge. This difference of a few pixels is often the difference between a cup making it or not.

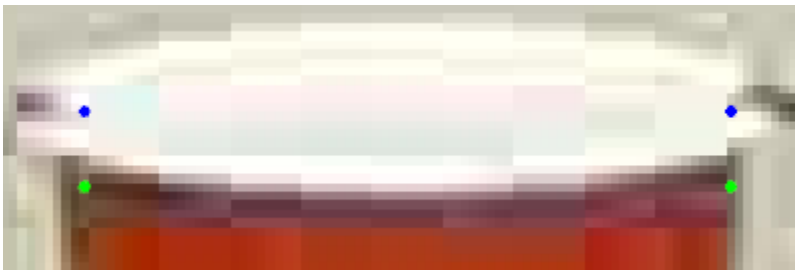

Figure 5: Adjusted cup edges.

From further research we realized that another approach we could use, to be more precise in cup detection, was “Object Detection Using Haar Cascades”. This is a fairly common method proposed by Paul Viola and Michael Jones in 2001. It is a machine learning based approach where lots of positives and negatives are thrown in the data set to train the classifier. Once the classifier has been trained from these images, unique features particular to the object can be extracted. Using Haar features just like our convolution kernel we subtract the sum of the pixels under the white rectangle from the sum of the pixels under the black rectangle(please see figure 6 for example of some Haar-like features). Even in windows of size 24 x 24, this method yields around 160,000 features.

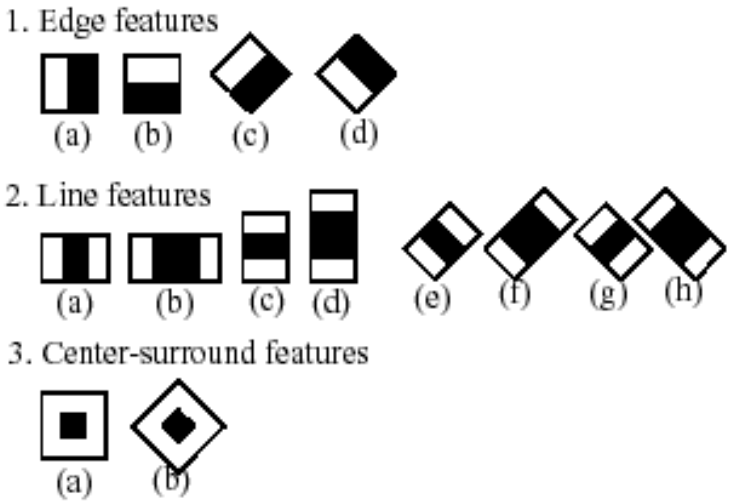

Figure 6: Some Haar-Like Features

However a downside to this approach is that some features may not be as distinguishable as others. Figure 7 is a primary example of such a scenario where the region of the eye is darker than the region of the nose and cheeks so is distinguishable however the area around the cheeks is nearly uniform therefore it is impossible to use this technique on such locations. Although we did not implement it, during the examination of the current literature we did find that a technique called “Adaboost” made this possible. Eventually we applied every feature on all training sets and found the best threshold, which would classify the positives and negatives and weight them accordingly. Eventually selecting features with minimum error rate.

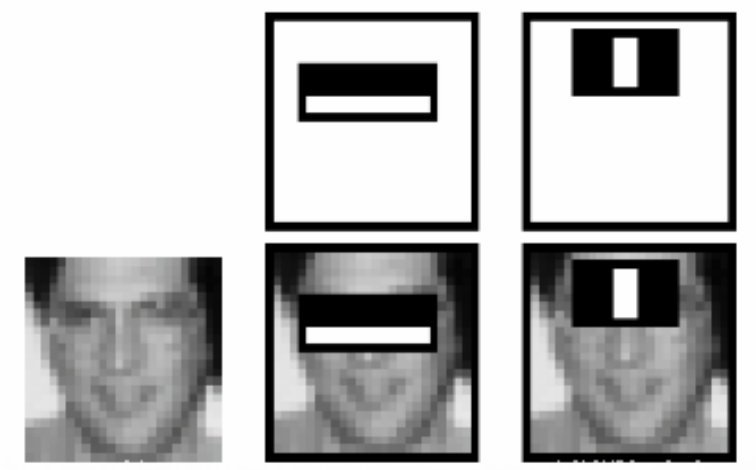

Figure 7: Haar-Like Features applied on a face

After large amounts of effort, we realized that we did not have sufficient time to get the algorithm to work the way we envisioned it. Therefore we consulted using openCV for this application however as such algorithms required serious computing power, they were all implemented in faster languages like C. Since our main code functioned in Matlab we wanted to revert the code to Matlab however as our main target for the project was to track project motion, not have the best cup detection algorithm, we decided to go with our preliminary approach which was simple yet fairly robust. If we had more time, we believe that “Object Detection using Haar Cascades” could make our system much more robust and reliable; as our current system is incapable of determining scenarios where the ball hits the side of the cup then goes in.

Prediction

All of the above components combine to allow our system to predict whether the ball will go in. We are able to correctly predict make/miss 83% of the time, with all confused shots being near misses, within 1 ball radius. most shots only take 11 frames, or a third of a second. Our predictions are rendered by the 6th or 7th frame. We found that giving humans the first half of frames for a shot and asking them to predict hit/miss yielded comparable results; many pong ball shots are hard to judge clearly, as the thresholds to make it are so small. One major factor that affects fitting accuracy is curvature of the shot; if we observe the apex of the trajectory, we get much better results, in general observing more curvature better constrains the higher-order terms.

V. Conclusions

Working with a moving object and trying to capture the motion with a simple point-shoot or cellphones camera is a challenging problem to tackle. The 3 dimensional in the real work environment does not translate directly to the 2 dimensional space in the camera world. Working on a single frame limit the ability to predict the final location of the object. Despite the limitation, the method least square-fitting algorithm used in this project were able to estimate the trajectory of the object and give a pretty accurate estimation of the final ball location.

Vi. References

1. http://www.hindawi.com/journals/ijcgt/2014/463489/
2. http://note.sonots.com/SciSoftware/haartraining.html
3. http://www.physicsclassroom.com/Class/vectors/u3l2a.cfm