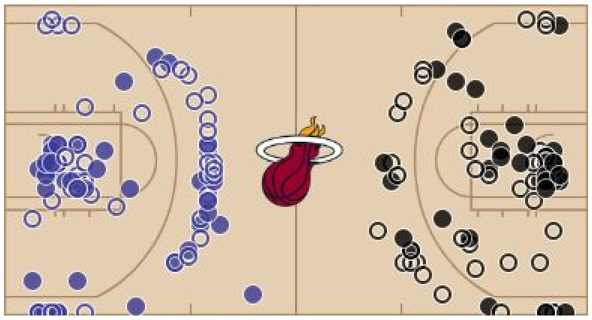
CS231A Project Proposal - Table Tennis Shot Chart

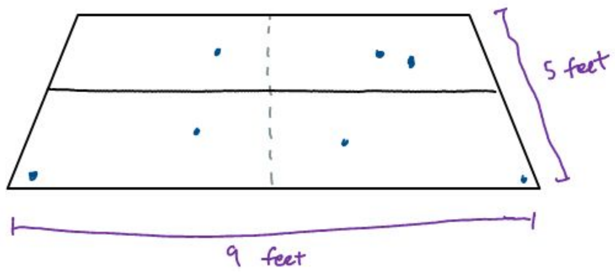
Dillon Koch

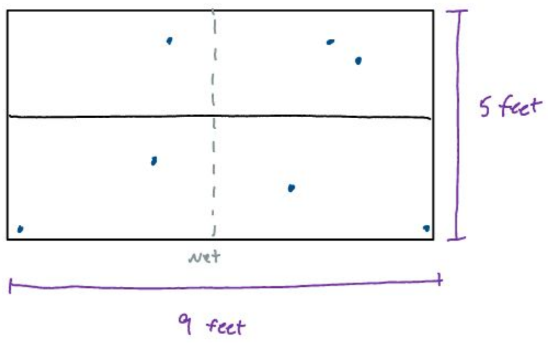
My project idea is to create a shot chart displaying where table tennis balls land during a point by replicating the results from this paper and adding new functionality to map points in the video to the shot chart. Shot charts are more commonly seen in basketball (below), showing where each team made and missed shots during a game. A similar chart can also be applicable to table tennis by showing where each shot lands on the table. This chart can be a useful tool for players to learn the patterns of where their own shots land, as well as where their opponents' shots land.



The researchers from the TTNet paper analyzed videos of ping pong games and trained a deep learning model to track the ball location, another model to detect events (bounce, net hit, non-event), and performed semantic segmentation to find the table and players. My plan is to replicate those results and combine them to create a shot chart. I can record the pixel location of the bottom of the ball each time a bounce is detected, representing the place on the table where the ball landed. I will put a small dot on those pixel locations, indicating that the ball bounced there.

However, the camera's projective transformation causes the table to be shaped like a trapezoid instead of a rectangle. If I just record the pixel locations on the table in the videos, the result will be a trapezoid-shaped shot chart instead of having a bird's eye view like the basketball chart. To remedy this, I plan to use the four corners of the table to perform a perspective transformation that will convert the table shape from trapezoid to rectangle. This same transformation has been successfully applied to scanning documents here and here.





To train the ball detection and event detection models, I will use the OpenTTGames dataset published by the TTNet authors for computer vision research. It includes 5 longer videos of table tennis gameplay and 7 shorter videos for testing. The videos are in HD and recorded at 120 frames per second. All videos come with a json file indicating the frames in which an event occurs, another json file with the two-dimensional position of the ball in each frame, and finally images with the semantic segmentation of the players, table, and scoreboard. These files can be downloaded using a quick python script with wget commands. All the relevant results from their paper can be reproduced using this dataset.

To track the ball during a given frame, the researchers trained a neural network to predict the ball's location given a stack of nine consecutive frames. The model's task is to predict the center of the ball in the last frame of the stack. These frames are downscaled from the original 1920x1080 resolution to 320x128. Once the ball's center is predicted, the researchers cropped another 320x128 image around the predicted center from the original full-resolution frame. This higher resolution image helps the model more accurately predict the ball's location. This process of predicting the ball's center on a downscaled image, then a cropped high resolution image was faster than predicting on the original frame once.

The event detection model is only used for detecting ball bounces and net hits. Similar to the ball detection model, it takes stacks of nine frames as input. It has a similar structure as the ball detection model, but with additional convolutional blocks. This model also uses a sigmoid activation function to output two probabilities, one for a net hit and another for a bounce. Due to the higher frequency of ball bounces compared to net hits, they trained the model with a weighted cross entropy loss. They also added random frames that did not include an event to the training set to help predict the absence of an event.

Lastly, there are multiple options available to detect the table within each video. The TTNet researchers used semantic segmentation to find each pixel belonging to the table. Another possible approach is to use edge detection and contours as mentioned in the document scanning post. Either method would provide a way to detect the lines and corner points of the table in each video.

The primary output of this project will of course be a table tennis shaped chart showing where the ball bounced throughout a rally or game. I'll evaluate this by comparing the chart to the video it's based on to ensure they match. It will also be important to evaluate the ball and event detection models. To evaluate the ball detection model, I'll use the RMSE of the ball's labeled position compared to the predicted position. To evaluate the event detection model, I'll compute the percentage of correct events predicted. The TTNet researchers achieved 1-3 pixels RMSE for ball detection in most settings and about 97% accuracy predicting events.

I have already written code to automate OpenTTGames dataset download. I also plan to have the event detection model trained (by 2/14) and the ball detection model trained (2/28) before the midterm progress report is due (3/1). After that report, I will write code to detect the table's coordinates (3/3), locate where the ball meets the table during frames with a bounce (3/6), and project those points to the shot chart output (3/10) before the project is due (3/18). This leaves time to prepare the GitHub repository, presentation, and final paper before the due date.