Estimating 3D trajectories from single camera padel videos

Benjamin de Charmoy
Labs
Thinkst Applied Research
Cape Town, South Africa
benjamin@thinkst.com

Abstract — Extracting useful information from single viewpoint videos of padel games relies on relating objects within each frame to a world coordinate system. The estimation of ball and player world coordinates using a single camera is an inherently ill posed problem. The measurement space is 2D in pixel units and the state space is 3D in meters. This paper explores the use of prior knowledge about the scene (court geometry) and object dynamics to produce world coordinate estimates. Additionally, a technique for single image camera calibration is presented that exploits the known scene geometry. Evaluation is done on a new dataset of live broadcast footage from the *Ooredoo Qatar Major 2023* padel tournament. The dataset as well as code is made available to the community as CourtVision.

Index terms—Padel analytics, State estimation, Single image camera calibration, Tracking

I. Introduction

Padel is a racket sport played by teams of two in a 10m by 20m court. The court has glass walls and the ball can be played off and onto the wall. Scoring works similarly to tennis and matches last between one and two hours. During game play, the players and a ball move around in the court. Occasionally players move outside the court to bring the ball back into play. These are valid tactics and rallies continue. This movement is captured by a single broadcast camera at approximately thirty frames per second. A single frame is shown in Figure 1. The information captured is rich and dense. It is, however, not a direct measurement of player and ball dynamics. It is the light radiating from the scene that passes through the camera's optics and falls onto the imaging sensor that is captured. The image formation is well understood [1] and Section II.A describes this process. This information is further transformed by compression and coding schemes and resides on disk.

Unpacking events that took place during game play and relating them to one another is explored in this paper. "Out of band" information such as the court dimensions, player and ball dynamics are exploited to achieve this.

Given these video recordings, can we reconstruct the events that took place with the goal to extract useful player metrics? Player metrics such as: how much time is spent at positions $\{x, y, z\}_{\text{time}}$ during a rally? Or how many meters are covered during a game? Or where on the court are most points won/lost from? The hypothesis is that these metrics are useful for both players and coaches as they can be used to improve training and game play. A second hypothesis is that player and game metrics enhance the viewing experience for spectators. For example, by overlaying player positions on the court during a rally would highlight positional tactics that would otherwise go unnoticed by an enthusiast in the audience.

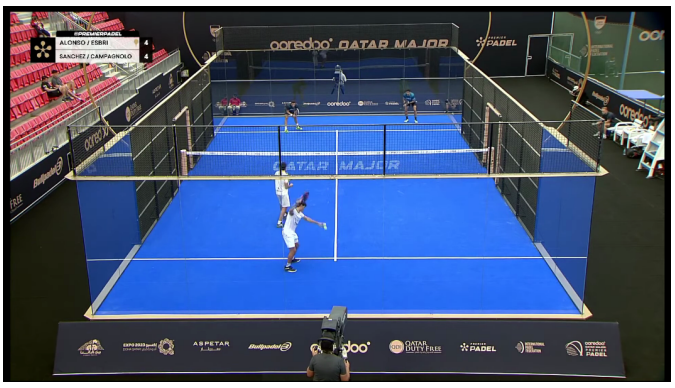

Figure 1: A typical broadcast view of game play.

To unwind and untangle these events from a recording four tasks were identified: single image camera calibration and camera pose estimation; image plane object detection; world coordinate frame object tracking; and data visualizations.

As part of this project a dataset CourtVisionPadelDataset was compiled from videos from the *Ooredoo Qatar Major 2023* tournament. This dataset consists of 6 hours of live broadcast. It's partially segmented to give timestamps of rallies and bounding boxes of the ball. Accompanying the dataset is a Python library, CourtVision, that implements the techniques described in this paper. CourtVision uses *Kornia* [2] a *Pytorch* [3] based computer vision library.

II. Unwinding events

This section details the four tasks aimed at unwinding events from the video recording. The first two tasks are concerned with mapping from the world coordinate frame to the image plane and locating regions of interest in each frame. The third task of tracking objects in the world coordinate frame exploits a physics model of the ball and players. The final task is concerned with visualizing the data.

A. *Single image camera calibration*

Camera calibration is the process of estimating both the intrinsic and extrinsic camera parameters. The intrinsic parameters K (camera matrix) and d (distortion coefficients) together with the pinhole camera model determine how rays are projected onto the sensor. The extrinsics R (rotation matrix) and t (translation vector) relate points in the world coordinate system to the camera coordinate system. Traditionally, camera calibration is done using a calibration rig - typically a flat checker board. The rig is placed at varying orientations in the scene and many images are captured. These images are then used to estimate the cameras' intrinsic parameters. Without access to the camera, calibration was performed by exploiting known scene geometry. Using the Padel court specifications [4] twenty six point correspondences were identified (see Figure 2).

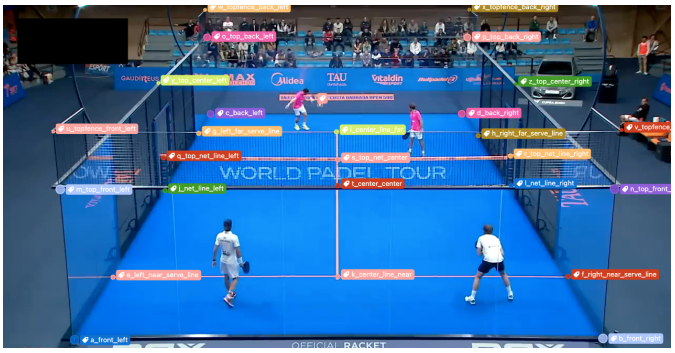

Figure 2: Twenty six points used for camera calibration and camera pose estimation.

Zhangs' [5] ubiquitous calibration approach, assumes camera coordinates all lie on the same plane - a checker board. To utilize this technique sets of co-planar points are needed. The padel court scene has many co-planar points. For example all points on the floor or all points on the front glass. In total eight such planes were used. Since the camera is stationary only a single frame was annotated per clip. The manual annotation process included an unquantified amount of noise. That is the human annotator may have been off by a few pixels. To alleviate this, CourtVisions' implementation added gaussian noise to each annotation and did an exhaustive search for the best K and d. The mean re-projection error was used to determine the best camera matrix and distortion coefficients. Using this method, a re-projection error of $3.50[\text{pixel}]$ in images of resolution 1920×1080 pixels was achieved.

A similar set of point correspondences were used to estimate the pose of the camera relative the court. Estimating camera pose from known point correspondences was done using OpenCV's *solvepnp* implementation. This resulted in a re-projection error of $9.78[\text{pixel}]$. Thus any point in the world coordinate frame within the court volume re-projected onto the image plane with an error of $9.78[\text{pixel}]$.

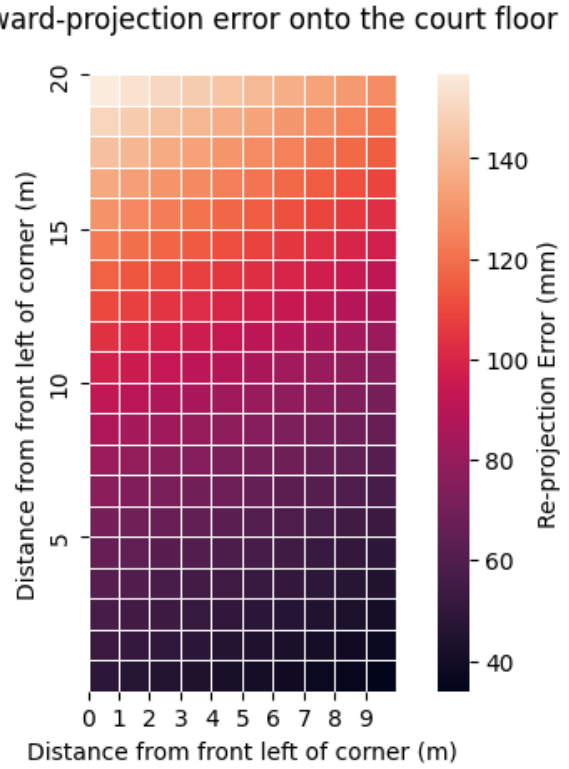

Figure 3: Forward projection error onto the court floor.

To relate this error to the world coordinate system a grid of points on the court floor were projected onto the image plane. Then for each point, a cluster of points with uniform noise of $9.78[\text{pixel}]$ were projected back to the court floor. The error is the distance between the projected point and the ground truth point. The mean forward projection error is $161[\text{mm}]$. This is 1.61% of the court width. The forward projection error is visualized in Figure 3 showing larger errors at the far end of the court. This is an encouraging result shows promise for single camera systems.

B. *Image plane object detection*

The task of object detection (players and the ball) in each frame is a well understood problem. There are a plethora of off-the-shelf models that can be fine-tuned to perform this task. However, the datasets available in the domain of padel is limited. To this end a cyclic pipeline of *train*, *evaluate*,

annotate, *train* was established. Using LabelStudio, a web based annotation tool a dataset of 20 images was annotated. The model was fine tuned and used to predict the ball locations on a further 20 images. These were then checked by a human and corrected. The model was retrained on the total 40 images and the process repeated. This was done until a high quality and diverse dataset was created consisting of 600 images.

For detecting the players a Yolov8 [6] model was used. Ball detection uses a Faster-RCNN [7] model. All detections were passed to the tracker (see Section II.C) which did player re-identification and tracking. The dataset and code used for fine-tuning these models is made available [8].

C. World coordinate frame object tracking

Image plane object detection provides pixel locations of the ball and players per frame. Camera calibration gave a mapping from world coordinates to image plane pixel locations. By combining these two results and exploiting the dynamics of a ball in flight, estimating the world coordinates of the ball is presented. The player tracking is done using the assumption that they remain on the court floor. This assumption this is not strictly needed, however, it is valid for the majority of the time and gives focus to ball tracking - the harder problem. The ball tracking is done using a particle filter.

Related work [9] which uses a single camera to track 3D trajectories of a shuttlecock during badminton games used a number of domain specific constraints. They seeded the tracker with initial positions and velocity constraints. This work took the view to shift prior knowledge into a distribution and apply a Bayesian filtering process. By placing such constraints into prior distributions the proposed tracker aims to easily generalize to other racket sports. The implementation of this Bayesian filtering process is realized as a particle filter.

Particle filters [10] represent possible states of the ball as particles each with an associated weight. The full state is defined by a set of N particles where each particle n is a vector $\vec{X}_{\text{p=n}} = \{x, y, z, \dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z}, w\}$. x is the position $\dot{x}$ is the velocity and $\ddot{x}$ is the acceleration in the $\hat{x}$ direction. Similarly for $y, \dot{y}, \ddot{y}$ and $z, \dot{z}, \ddot{z}$ in the $\hat{y}$ and $\hat{z}$ directions. w is the probability mass at that location in the state space. A measurement (detection) is defined as $\vec{Z}_t = \{u, y, l\}$ at time t where u, v are the pixel coordinates of the detection center and l is the label which is in the set $\{\text{ball}, \text{playerA1}, \text{playerA2}, \text{playerB1}, \text{playerB2}\}$. The particles are initialized at random with x, y, x drawn from a uniform distribution over the cube making up the court. Ball speeds during a serve are approximately $20\,\text{km/h}$ to $60\,\text{km/h}$. Thus, velocity is drawn from a normal distribution with mean $0.0[\text{m/s}]$ and standard deviation $8.3[\text{m/s}]$. The acceleration component is drawn from a normal distribution with mean $0.0\left[\frac{m}{s^2}\right]$ and standard deviation 0.1 with $\ddot{z} = -9.8\left[\frac{m}{s^2}\right]$. The weight is set to $\frac{1}{N}$. The state is updated every $dt = \frac{1}{f}$ seconds where f is the frame rate of the video.

Particle filters are a sequential Monte Carlo method and have two steps; *predict* and *update*. During the *predict* phase a new set of particles are sampled from the previous state according to:

$$p(x_{\text{t+1}} |\ z_{1:t}) = \int p(x_t|x_{\text{t-1}})p(x_{\text{t-1}}|z_{1:\text{t-1}})dx_{\text{t-1}} \qquad (1)$$

The *predict* phase incorporates dynamics of the ball $p(x_t|x_{\text{t-1}})$ which propagates each particle according to the following pseudo code:

```
def predict(state: Tensor, dt: float) -> Tensor:

  noise = torch.randn_like(state) * motion_noise
  # Update position
  state[:, :3] += state[:, 3:6] * dt + noise
  # Update velocity
  state[:, 3:6] += state[:, 6:9] * dt + noise
  # Update acceleration
  state[:, 6:9] += noise
```

During the *update* phase the particles are weighted according to the likelihood of the measurement z_t at time t. $p(z_t|x_t)$ is the likelihood of the measurement given the state. The likelihood is calculated weights updated using the following pseudo code:

```
def update(state:Tensor, obs: Tensor) -> Tensor:
  # predicted measurement given the state
  pred_obs = state[:,:3]@world_to_cam@cam_to_image
 # Error predicted observation and the measurement
  error = torch.norm(pred_obs - obs, dim=1)
  # The likelihood of the error
  likelihood = torch.exp(
    -0.5 * error ** 2 / measurement_noise
  )
  # Update weight using likelihood
  state[:, -1] *= likelihood
  # Normalize the weights
  state[:, -1] /= state[:, -1].sum()
  # Resample the particles
  state = resample(state)
```

The ball tracker outputs a state estimate for each frame. This is the mean of the particles weighted by their probability mass. The mean state is then projected back onto the image plane. Since the tracker holds N particles, the uncertainty of the state can be estimated. The uncertainty is calculated as the standard deviation of the particles weighted by their probability mass. The ball tracking converges to a smooth trajectory when consecutive detections are made. Figure 5 shows the internal representation of the tracker for the ball. At each time step the tracker holds a full posterior distribution of the state given all previous states and observations.

The shape of this distribution changes and is able to quickly adapt when the ball bounces or hits a wall. This allows for consistent tracking despite not having impacts (hits and bounces) included in the model. The mean estimate of the ball position is shown in Figure 4. The tracker is able to handle false detections but too many will cause the tracker to fail. The ball detector is not very robust and false detections are common. Figure 6 shows the trackers' internal state when false detections are present. The tracker is able to recover from these false detections and continue tracking the ball.

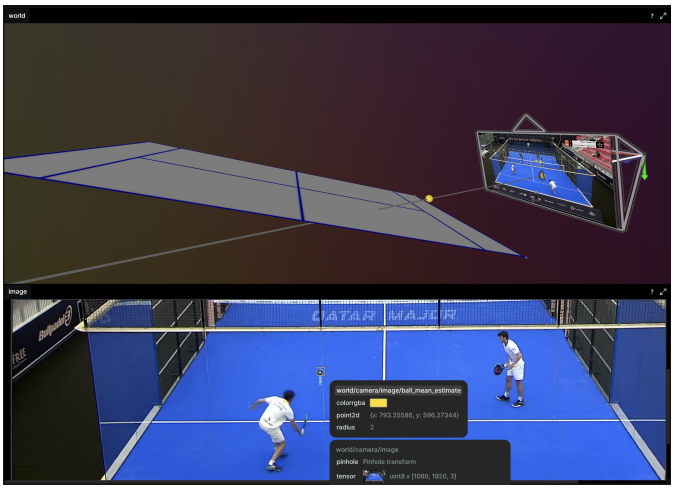

Figure 4: Showing the mean estimate of the ball position.

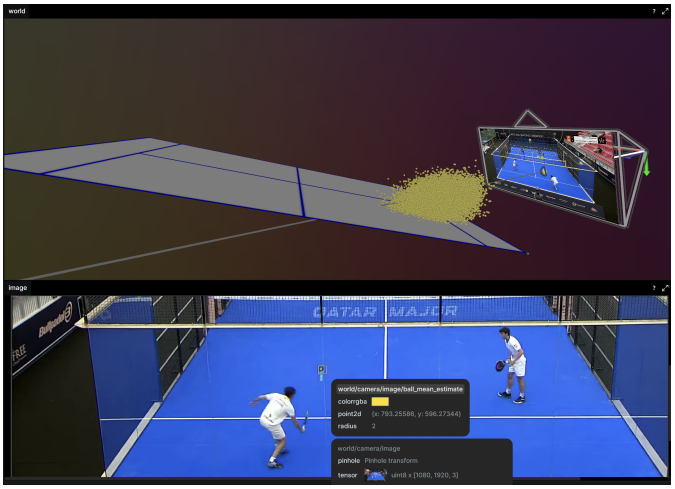

Figure 5: Showing the full state of the tracker for the ball.

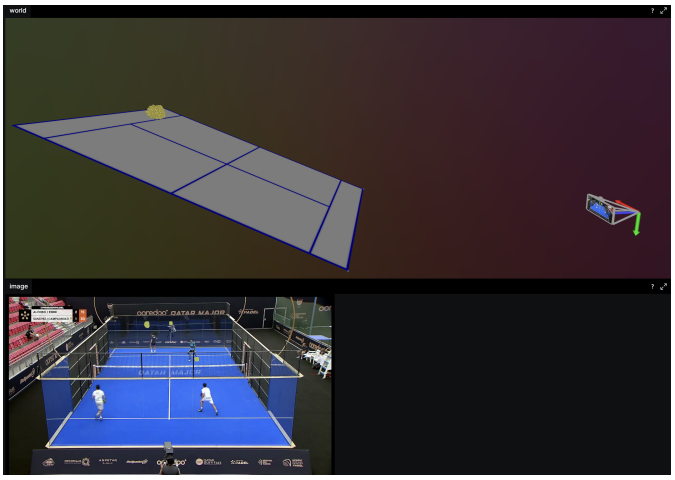

Figure 6: Lower frame shows the image plane detections with false detections on a players foot and the camera operator outside the court. The upper frame shows the trackers' internal state.

D. Data visualizations

Looking at the player positions during a single, and across multiple, rallies during a game is a useful visualization. It gives insight into player tactics and movement. The player positions are projected onto the court floor. Here player The heatmap is generated by summing the time a player is at a position over all frames for a rally in Figure 7 and over multiple rallies in Figure 8. Velocity heatmaps are generated in a similar fashion. The velocity is calculated as the difference in position per time interval. The figures come from rally *635b* and game *0000* of the dataset [8].

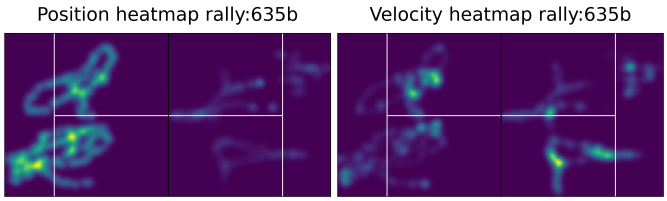

Figure 7: Player positions (left) and speeds (right) during a rally.

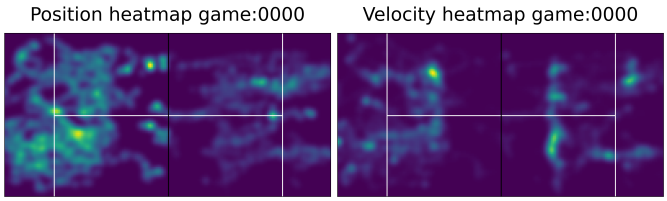

Figure 8: Player positions (left) and speeds (right) during a game.

These are examples of the visualizations that can be generated from the data. The full set of visualizations are made available [8].

III. Conclusion

Single camera systems are a cost effective solution for capturing events during a padel game. This work tackled a number of common vision problems and applied them to the domain of padel. The results are encouraging and show promise for single camera systems. The single image camera calibration technique using multiple sets of co-planar points is a useful technique when access to the camera is not possible. The ball and player detection models were fine tuned on a novel dataset which was released along side this work. The ball and player trackers coupled with the camera calibration and pose estimation enabled world coordinate tracking. These tracks are artifacts of events that took place during the game play and were captured by the camera. Visualizations of these results give insight into player tactics and movement but fall short as a complete picture. The ball trajectory estimates are promising and tracking is consistent across a number of false and / or missed detections. The ball tracker is able to track the ball through bounces and hits without explicitly modeling these events.

Future work should look at quantifying the tracker results which would need 3D ground truth data from a multi camera system. This work does fall short on providing useful information to players and coaches. Further work is needed on visualizations and an interactive tool to explore the data recovered in this work.

IV. Acknowledgments

The author would like to thank Thinkst Applied Research.

References

[1] R. Hartley, and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed., New York, NY, USA: Cambridge University Press, 2003.

[2] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski, "Kornia: an open source differentiable computer vision library for pytorch," 2019. [Online]. Available: http://arxiv.org/abs/1910.02190

[3] A. Paszke, S. Gross, et al., "Automatic differentiation in pytorch."

[4] LTA, "Lta padel court data sheet." [Online]. Available: https://www.lta.org.uk/4ad2a4/siteassets/play/padel/file/lta-padel-court-guidance.pdf

[5] Z. Zhang, "A flexible new technique for camera calibration (technical report)," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, 2002, doi: 10.1109/34.888718.

[6] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[8] B. de Charmoy, "CourtVision," 2023. [Online]. Available: https://benjamindev.github.io/courtvision

[9] P. Liu, and J.-H. W. A. Seattle, "Monotrack: shuttle trajectory reconstruction from monocular badminton video."

[10] A. Doucet, and A. Johansen, "A tutorial on particle filtering and smoothing: fifteen years later," *Handbook Nonlinear Filtering*, no. December, pp. 656–704, 2011, doi: 10.1.1.157.772. [Online]. Available: http://automatica.dei.unipd.it/tl_files/utenti/lucaschenato/Classes/PSC10_11/Tutorial_PF_doucet_johansen.pdf