This CVPR Workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

TT3D: Table Tennis 3D Reconstruction

Thomas Gossard * Andreas Ziegler Andreas Zell
Cognitive Systems - Univeristy of Tuebingen
Sand 1 Tuebingen 72076 Germany
thomas.gossard@uni-tuebingen.de

Abstract

Sports analysis requires processing large amounts of data, which is time-consuming and costly. Advancements in neural networks have significantly alleviated this burden, enabling highly accurate ball tracking in sports broadcasts. However, relying solely on 2D ball tracking is limiting, as it depends on the camera's viewpoint and falls short of supporting comprehensive game analysis. To address this limitation, we propose a novel approach for reconstructing precise 3D ball trajectories from online table tennis match recordings. Our method leverages the underlying physics of the ball's motion to identify the bounce state that minimizes the reprojection error of the ball's flying trajectory, hence ensuring an accurate and reliable 3D reconstruction. A key advantage of our approach is its ability to infer ball spin without relying on human pose estimation or racket tracking, which are often unreliable or unavailable in broadcast footage. We developed an automated camera calibration method capable of reliably tracking camera movements. Additionally, we adapted an existing 3D pose estimation model, which lacks depth motion capture, to accurately track player movements. Together, these contributions enable the full 3D reconstruction of a table tennis rally. Project page: https://cogsys-tuebingen.github.io/tt3d/

1. Introduction

Accurately capturing the state of a game is fundamental for effective sports analytics. Traditionally, this task required extensive manual annotation, making it labor-intensive and time-consuming. However, recent advancements in machine learning have significantly automated this process, providing powerful tools to analyze gameplay and extract meaningful insights [25]. Learning-based methods [49] allow for the precise extraction of player poses, while advanced object detectors enable effective tracking of the ball and other key objects [16]. By leveraging these automated techniques, raw data can be systematically processed to evaluate player performance, uncover tactical patterns, and refine strategic decision-making, ultimately offering a competitive edge. This processed data can have multiple uses. For example, motion-captured recordings of table tennis games were used to find the important shot characteristics necessary to win a rally [24]. Wu et al. also showed that it is possible to predict the ball bouncing position from the serve stroke motion [44]. Using such data can furthermore enable a policy to learn sport skills like tennis strokes [48]. Finally, precise game state estimation has the potential to enable automated officiating, reducing human error in refereeing and improving fairness in sports [42].

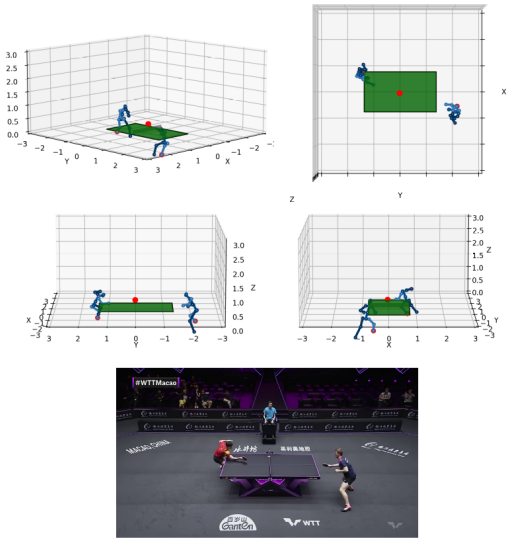


Figure 1. (Top) 3D Reconstruction of table tennis game. We show the reconstruction from different points of view with the players and the ball (red dot). (Bottom) Corresponding frame from real footage.

Despite these advancements, a key challenge remains: most game recordings are captured using monocular cam-

*This research was funded by Sony AI.

eras, leading to a loss of depth information. While 2D ball tracking provides valuable insights, it remains inherently limited by perspective distortions and camera positioning. For a more robust and comprehensive analysis, spatial data must be reconstructed in 3D, as it eliminates viewpoint dependencies and allows for a more accurate representation of ball motion and interactions. Therefore, the primary objective of this paper is to achieve 3D reconstruction of sports games, with a specific focus on the challenging task of reconstructing ball trajectories for table tennis.

Table tennis presents unique challenges due to its highly dynamic nature. For comparison, tennis averages 1.3 hits per second [1], while badminton ranges from 1.26 to 1.76 hits per second [2, 20]. In contrast, table tennis can reach up to 2 hits per second [2]. Additionally, ball spin plays a crucial role in shaping its trajectory. The Magnus effect causes the ball to curve in mid-air, while its spin significantly influences its bounce trajectory, altering its direction upon impact. These factors make the ball's 3D trajectory particularly complex, as the reconstruction must accurately account for aerodynamic forces and bounce dynamics compared to badminton. But these dynamics are also what will help infer the 3D motion of the ball from just the 2D observation. Finally, we have to deal with a lot of ball occlusion due to the players, depending on the point of view.

In this work, we propose a complete pipeline to reconstruct the 3D state of the table tennis game: 3D player pose and 3D ball trajectory. For this purpose, we developed an automated camera calibration method capable of tracking both the camera's pose and focal length. It achieves this by locating the table corners using a segmentation mask of the table surface. We also propose a novel method to reconstruct 3D ball trajectories from monocular table tennis match recordings. Our approach leverages the physics of ball motion to reconstruct the ball's 3D trajectory from the table bounce position, achieving robust and accurate predictions even under challenging real-world conditions. Integrating 3D human pose estimation finally allows us for a complete 3D reconstruction of table tennis matches.

2. Related Work

Many 3D reconstruction methods have been developed for different sports such as badminton [21], football [23], tennis [12], basketball [8, 47], table tennis [7, 32, 36] or volleyball [9]. Reconstructing the 3D state of a ball sport involves three fundamental components: camera calibration, ball detection and tracking, and 2D-to-3D lifting. Each step plays a critical role in ensuring accurate 3D trajectory reconstruction.

2.1. Single View Camera Calibration

Camera calibration is essential to determine the camera matrix $\boldsymbol{K}$ and the camera pose, which is defined by the rotation matrix $\boldsymbol{R}$ and the translation vector $\boldsymbol{T}$. These parameters enable the transformation between 3D world coordinates and 2D image coordinates using the pinhole camera model. Specifically, given a 3D world point $\boldsymbol{X_w} = [X, Y, Z]^T$ and its corresponding image point $\boldsymbol{x} = [u, v, 1]^T$, the relationship is defined by the projection matrix $\boldsymbol{P}$, expressed as

$$\boldsymbol{x} = \boldsymbol{P}\boldsymbol{X_w} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{T}]\boldsymbol{X_w}. \tag{1}$$

Typically, camera calibration is conducted using a moving calibration pattern, such as a checkerboard, an asymmetric circle grid, or Aprilgrid [26]. Broadcast footage usually does not require camera calibration, and information about the camera's placement or focal length is generally not provided in such recordings. To address this, objects of known geometry in the scene are used for calibration.

In racket sports like badminton and tennis, the intersections of court lines and net poles are commonly used [21, 45]. This process relies on a combination of computer vision techniques, including color filtering, Canny edge detection, Hough line transforms, and sport- or Point Of View (POV)-specific heuristics. Currently, no automatic calibration methods exist for table tennis. This is due to the limited number of field features compared to sports like tennis or badminton, the smaller court size, variations in field color schemes, diverse recording POVs, and frequent occlusions. As a result, many approaches resort to manually annotating features on the images for the calibration process [7, 12, 32, 36, 42].

2.2. Ball Tracking

Once the camera is calibrated, the next step is ball detection and tracking. Traditional computer vision methods, including color filtering, background subtraction, and blob detection, have been widely used for this task [5, 14, 36, 38, 42]. More recently, deep learning-based approaches, such as fine-tuned general object detectors like Detectron2 [7] and YOLOv4 [18], have gained popularity in sports applications. However, in scenarios where a single ball is present, a common approach is to generate a heatmap representing its most probable location using models such as DeepBall [13], TracketNet and its variants [10, 13, 29, 34], Monotrack [21], WASB [37], and BlurBall [4]. Among these, BlurBall not only estimates the ball's position but also provides motion blur information, offering valuable insight into velocity. This is particularly beneficial in table tennis, where fewer frames are available to capture ball trajectories compared to other racket sports. In our pipeline, we adopt BlurBall as the primary ball detector.

2.3. 3D Reconstruction

With the 2D ball positions available, we now need to reconstruct the 3D data from it. The first step is to segment the whole rally into individual ball trajectories, i.e.

between two player strikes. Previous methods either use heuristics on the change in velocity [7, 32] or Gated Recurrent Unit (GRU) networks to detect frames where the ball bounces or is hit by players [12, 21], using inputs such as field coordinates, ball positions in image space, and player pose. With the individual trajectory isolated, we can infer the 3D trajectory.

Calandre et al. [7] propose the use of the observed diameter of the ball to infer its 3D position, with depth estimated using the camera's intrinsic parameters. The resulting 3D positions are then projected onto a 2D plane fitted to all observed ball locations from the trajectory. This approach assumes that no sidespin is applied to the ball. Unfortunately, this method is impractical for broadcast recordings, as the ball's small apparent diameter (lower than 10 pixels) and the sensitivity of depth regression to measurement errors make accurate 3D position estimation unfeasible. Nonetheless, humans are still able to infer the ball's 3D trajectory to some extent. This ability stems from our understanding of the physics governing the ball's flight and bounce, which constrains the predicted 3D trajectory only to plausible paths. The flight path of the ball can be modeled using an Ordinary Differential Equation (ODE), where the trajectory is predicted from an initial position and velocity. By projecting the predicted trajectory onto the image plane using the pinhole camera model, the ball's initial state can be optimized to minimize the reprojection error. Liu et al. employed this approach for badminton [21]. However, the optimization problem is non-convex and prone to local minima, requiring good initialization or additional constraints to guide convergence toward the correct solution. Liu et al. addressed this issue in badminton by constraining the initial and final 3D positions of the shuttlecock, ensuring they were close to the player's hand as estimated by a pose detection model. A similar approach was tested in table tennis [32] but the additional constraints were set as the ball's first and second bouncing position for the serve. Indeed, the 3D position of the bouncing ball can be inferred as a ray-plane intersection. However, the defined ODE completely ignores drag and the Magnus effect and this method was only tested for serves. An alternative to optimization-based methods is treating the problem as a regression task. In tennis, SynthNet [12] utilizes a synthetic dataset of ball trajectories to train a Multi-Layer Perceptron (MLP) that predicts the ball's initial position and velocity based on its 2D trajectory before the bounce and the court's pose.

3. Camera Calibration

Camera calibration requires matching 2D image features to their corresponding 3D world points. In table tennis, the table's standardized dimensions make it the only viable calibration target. The world frame is thus defined relative to the table, as illustrated in Fig. 3d.

A closed-form solution for estimating camera parameters can be obtained using Direct Linear Transform (DLT) if at least six non-coplanar points are available. For the table, potential calibration features include its corners (4 points), center line (2 points), and net poles (2 points). However, these features are often obstructed by players or not detected, making the use of DLT impractical. To simplify the problem, we assume no lens distortion ($\boldsymbol{d} = 0$), the optical center aligns with the image center ($c_x = w/2, c_y = h/2$), and the focal lengths are equal in both directions ($f_x = f_y$). Under these assumptions, the problem falls into the Perspective-n-Point (PnP) domain with unknown focal length (PnPf). Only 3.5 points are required to estimate the camera's focal length f and extrinsic parameters $\boldsymbol{R}$ and $\boldsymbol{T}$ [43]. As such, we can calibrate the camera with only 4 points (which can be coplanar). Since it is unlikely for an entire table edge to be obstructed, we chose the table corners—defined as the intersections of the table edges—as our calibration points.

In the following sections, we discuss the various components of the calibration process. Section 3.1 details the camera calibration approach, assuming we already have access to the necessary keypoints. Next, we introduce our table tennis table segmentation model in Section 3.2. Finally, in Sec. 3.3, we describe how the table mask is used to extract the features necessary for calibration.

3.1. Calibration

Existing PnPf methods are highly complex and depend on advanced algebraic solvers, such as Gröbner solvers [17, 27, 43, 50, 51]. Zheng et al. [50] showed that, with a good initialization of the camera pose and focal length, minimizing the reprojection error can achieve accuracy comparable to state-of-the-art PnPf methods. Given this insight, we opted to frame the PnPf problem as an optimization problem, resulting in a much simpler approach. This is feasible because the camera is always positioned within a specific distance from the table and must cover the entire field, significantly constraining the range of possible focal lengths. We describe the optimization procedure in Algorithm 1. We begin by using PnP to estimate the camera pose ($\boldsymbol{R}$ and $\boldsymbol{T}$) with the currently estimated focal length f. Next, we refine f by minimizing the reprojection error. These two steps are repeated iteratively until the focal length converges. As mentioned earlier, proper initialization is crucial. Manual annotation of features using DLT yielded focal lengths ranging from 1000 to 3000. Therefore, we initialize the focal length at 1500. The advantage of our approach is the flexibility. It can work with just 4 coplanar points.

To validate our approach, we conducted tests using synthetically generated points for random table positions, orientations, and focal lengths to assess the calibration accuracy of the calibration. As shown in Figure 2, the table's ro-

Algorithm 1 Estimate Focal Length By Minimizing Reprojection Error

Require: Set of 3D points $\mathbf{X} = \{\boldsymbol{X}_i\}$, corresponding 2D image points $\mathbf{x} = \{\boldsymbol{x}_i\}$, initial focal length f_0, convergence threshold ϵ, maximum iterations N

Ensure: Optimized focal length f^*

Initialize $f \leftarrow f_0$
Set iteration counter $k \leftarrow 0$
repeat
    Solve the PnP problem with f to estimate $\mathbf{R}, \mathbf{T}$
    Compute the total reprojection error:

$$E(f) = \sum_i ||\boldsymbol{x}_i - \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{T}]\boldsymbol{X}_i||$$

    Update f by minimizing $E(f)$ using a line search
    Increment iteration counter $k \leftarrow k + 1$
until $|E(f_{\text{prev}}) - E(f)| < \epsilon$ or $k \geq N$
Set $f^* \leftarrow f$
return f^*

tation is accurately estimated, with an average error of only 3 degrees. However, the translation vector and focal length show significant errors. This limitation arises from using only four planar points, which results in a strong correlation between focal length and depth estimation. This correlation makes it challenging to estimate these parameters independently. Despite this limitation, it doesn't impact the downstream reconstruction task, as errors in one parameter are often compensated for by corresponding adjustments in the other. The method demonstrates strong performance in estimating the XY components, achieving a Mean Relative Absolute Error (MRAE) of just 5%, indicating its particular effectiveness in the plane of the table.

The described method can be applied to individual frames. However, when processing a video with potential camera movement and zooming, integrating a Kalman Filter can help refine the raw observations, providing smoother and more accurate estimates.

3.2. Segmentation Model

To extract the features necessary for the camera calibration, we need to accurately identify the table. We propose to do so with a segmentation model. Promptable segmentation models like SAM [30] can identify table regions, even with players occluding parts of the table. However, they struggle with net segmentation and often include the table's vertical sides in the mask. The same goes for depth models like DepthAnything V2 [46] which do not perfectly allow to identify the table's surface.

To address our specific requirements, we developed a custom table segmentation model. We began by assembling a dataset of 1,200 images sourced from online videos and manually labeled them. We then fine-tuned pre-trained models for table tennis table segmentation. Given the task's relative simplicity, we focused on lightweight models with no more than 2 million parameters. After extensive testing, we selected a U-Net++ architecture [52] with a pre-trained RegNetY encoder [28] (2M parameters) for its optimal balance between inference speed and accuracy. The model was trained using the Dice loss [33]. Our final model achieves a mean Intersection over Union (mIoU) of 0.92 on the test set and runs at 70 frames per second, making it suitable for real-time applications.

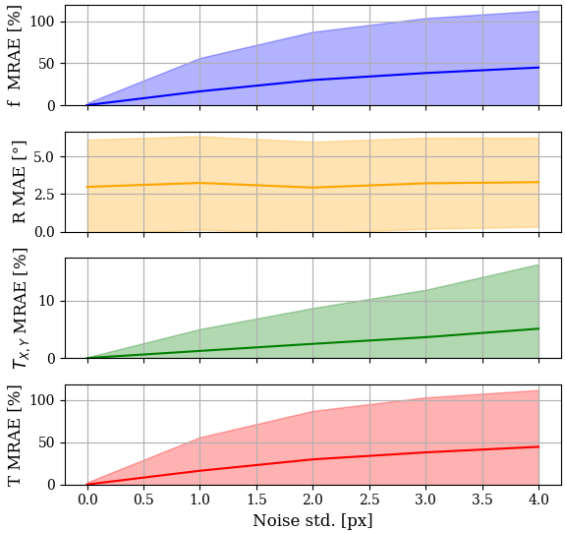
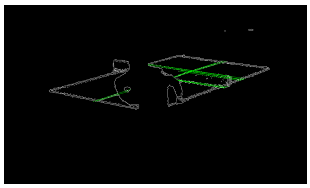


Figure 2. Mean Relative Absolute Error (MRAE) and Mean Absolute Error (MAE) for camera parameters estimated with Algorithm 1 for 1000 samples with noise. The table position was randomly sampled within the camera frame coordinates, ranging from [-2, -2, 5] to [2, 2, 20] (m), and the focal length was randomly sampled between 500 and 3000.

3.3. Feature extraction

We first try to extract the table corners. We start by extracting the table contours from the segmentation mask using the method proposed by Suzuki et al. [35] (Figure 3a). Smaller contours are filtered out, keeping only the two largest ones. If the mask is not convex, additional internal contours might appear, which are also removed. Using the probabilistic Hough transform [22], we identify and group similar lines to detect the table edges. If exactly four lines remain, their intersections are calculated and sorted clockwise to determine the table's four corners (Figure 3c). These corner points are then used for an initial camera calibration using the method described in Sec. 3.1. To lift the corner ordering ambiguity, we perform this calibration twice with different corner orderings and select the result with minimal

(a) Mask (yellow) and contours (red)



(b) Canny edge within table mask (white) and Hough lines (green)

(c) Table edges (blue lines), midline (green), and detected features (numbered points)

(d) Calibrated camera with reprojected features.Red, Green and blue axes are respectively X, Y and Z.

Figure 3. Calibration pipeline. (a) Mask with contour, (b) Midline detection (c) Detected corners and midline, (d) Final calibrated camera.

reprojection error and a plausible transformation (e.g., the camera isn't unrealistically far, typically under 10m).

This initial calibration can then be leveraged to detect other features such as the midline. We detect the midline using a combination of Canny edge detection and the Hough line transform within the table mask as shown in Figure 3b. The detected lines are filtered based on the known directions of the table edges, and the midline points are defined by the intersection between the midline and the back edge of the table. The midline points are not necessary for calibration but will improve the calibration accuracy.

4. Trajectory Segmentation

We use BlurBall [4] to detect the ball for a whole rally. To analyze individual ball trajectories, we segment the rally using piecewise polynomial segmentation. Since a ball's flight trajectory can be well approximated by a second-degree polynomial [41], this approach naturally aligns with the motion characteristics of the ball. These trajectory segments can be effectively identified through segmentation and solved efficiently using dynamic programming, as described in [11]. The solution is one that minimizes

$$J = \sum_{t}^{T} \left(|u_t - P_u(t)| + |v_t - P_v(t)| \right) + \lambda K, \qquad (2)$$

where (u_t, v_t) are the ball pixel coordinates at time t, P_u and P_v are respectively the piecewise polynomes fitted to u and v, λ is a penalty for creating new polynomes and K is the number of segments in the piecewise polynomes.

This approach was selected for its robustness, as it is independent of the POV and, unlike previous methods using GRU networks [12, 21], does not rely on the players' pose. However, we observed that the number of frames between the table bounce and racket strike can be quite low. Attacks require the ball to be hit as early as possible which then leads to obstruction by the player. The number of balls observed in between can be too small to be detected as a new segment. To solve this, we leverage the additional blur information provided by BlurBall [4]. The blur is linked to the ball's velocity and thus to the derivative of the polynomials.

$$B_{error} = \left| \theta_t - \arctan\left(\frac{P_v(t)'}{P_u(t)'}\right) \right| \qquad (3)$$

where θ_t is the observed motion blur. We add this blur error B_{error} to the cost function J.

The piecewise polynomial segmentation offers greater generalizability compared to heuristic approaches, such as detecting changes in the ball's velocity, as used in [7, 32].

Once these polynomials are determined, bounce positions and times can be precisely calculated at the intersection points of consecutive polynomial segments. This is extremely important as most recordings run at 25 fps. This would lead to an uncertainty in the bounce time of 40 ms (time between successive frames for 25 fps) and a bounce position error that can go up to 40 cm for a ball with a velocity of 10 m/s. As explained in Section 5.2, we require a very accurate bounce position for our reconstruction to work. This method offers higher accuracy than previous approaches, which are inherently limited by the video framerate. An example of the segmentation is shown in Figure 4.

To differentiate between racket strikes and table bounces, we employ a heuristic approach. Specifically, we assume that a change in direction along the table's longitudinal axis indicates a racket strike.

4.1. Bouncing Position

Throughout the ball's trajectory, there exists a discrete moment where its exact 3D position can be determined: the bounce. At this instant, the ball is in direct contact with the table, whose pose is known from the camera calibration. This constraint restricts the possible ball positions to a 3D plane defined by its normal vector $\boldsymbol{n}$ and a point $\boldsymbol{X_{plane}}$. Furthermore, trajectory segmentation provides the precise image coordinates of the ball at the bounce point, allowing us to compute the corresponding ray direction as

$$\boldsymbol{d} = \boldsymbol{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \qquad (4)$$

The intersection of this ray with the plane yields the 3D bounce position of the ball

$$\boldsymbol{X_{bounce}} = \frac{\boldsymbol{X_{plane}} \cdot \boldsymbol{n}}{\boldsymbol{d} \cdot \boldsymbol{n}} \boldsymbol{d}. \qquad (5)$$

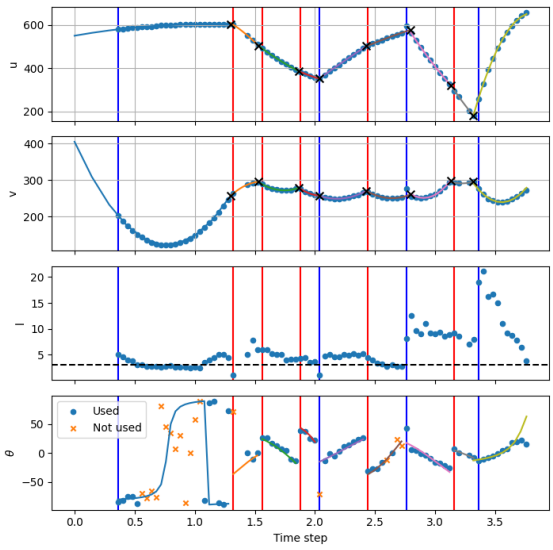


Figure 4. Segmentation of a rally. The dots represent the detected ball positions, while the vertical lines delineate different segments. Using heuristics, table bounces are marked in red and racket bounces in blue. The black crosses indicate the precise bounce coordinates, determined by the intersection of the fitted second-degree polynomials.

However, the computed intersection lies on the court plane, whereas the actual contact point is slightly offset due to the ball's radius. This offset is particularly significant when the camera's viewpoint is low and close to the court plane, where perspective effects amplify small vertical errors. We deal with this by shifting $\boldsymbol{X_{plane}}$ upwards by the radius of the ball with regard to the table.

5. 3D Reconstruction

Using the camera calibration and the segmented 2D ball trajectory, we reconstruct the 3D ball trajectory. To achieve this, we leverage the ball dynamics described in Section 5.1. These dynamics are used to optimize the ball's bounce velocity and spin, minimizing the reprojection error as detailed in Section 5.2.

5.1. Physics Model

The ball dynamics are divided into two categories: the airborne motion of the ball and its behavior during bounces. The airborne ball trajectory is defined by the following ODE [40]:

$$m\dot{\boldsymbol{v}} = \underbrace{k_D||\boldsymbol{v}||\boldsymbol{v}}_{\text{Drag}} + \underbrace{k_M\boldsymbol{\omega}\times\boldsymbol{v}}_{\text{Magnus}} + m\boldsymbol{g} \tag{6}$$

where $m = 2.7 \times 10^{-3}\,\mathrm{kg}$ is the mass of the ball, $\boldsymbol{v}$ and $\boldsymbol{\omega}$ are respectively the ball velocity and spin, $k_D = 3.8 \times 10^{-4}\,\mathrm{kg}\cdot\mathrm{s}^{-4}$ is the drag coefficient, $k_M = 4.86 \times 10^{-6}\,\mathrm{kg}\cdot\mathrm{s}^{-4}\cdot\mathrm{m}^{-1}$ is the Magnus coefficient and $\boldsymbol{g}$ is the gravity vector. The ODE in Equation (6) lacks an analytical solution due to the quadratic drag term and the Magnus force's dependence on spin and velocity. However, it can be solved as an initial value problem using Runge-Kutta or collocation methods.

The bounce is a discrete event that is not described by an ODE. It is modeled using a Coulomb friction model [6] with:

$$\begin{aligned}\boldsymbol{v}^+ &= \boldsymbol{A}\boldsymbol{v}^- + \boldsymbol{B}\boldsymbol{\omega}^- \\ \boldsymbol{\omega}^+ &= \boldsymbol{C}\boldsymbol{v}^- + \boldsymbol{D}\boldsymbol{\omega}^-.\end{aligned} \tag{7}$$

where $\boldsymbol{v}^-$, $\boldsymbol{\omega}^-$ and $\boldsymbol{v}^+$, $\boldsymbol{\omega}^+$ are respectively the ball velocity and spin before and after the bounce. The dynamic matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ will be different depending on whether the ball is rolling or sliding. The bounce type can be distinguished with the coefficient α.

$$\alpha = \frac{\mu\,(1+k_{COR})\,|v_z^-|}{\sqrt{\left(v_x^- - \omega_y^- r\right)^2 + \left(v_y^- + \omega_x r\right)^2}} \tag{8}$$

where $k_{COR} = 0.85$ is the Coefficient Of Restitution (COR), $\mu = 0.3$ is the friction coefficient and $r = 0.02\,\mathrm{m}$ is the radius of the ball.

If $\alpha \geq 0.4$, then the ball is rolling and we use the following matrices:

$$\begin{aligned}\boldsymbol{A} &= \begin{bmatrix} 1-\alpha & 0 & 0 \\ 0 & 1-\alpha & 0 \\ 0 & 0 & -k_{COR} \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 0 & \alpha r & 0 \\ -\alpha r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \boldsymbol{C} &= \begin{bmatrix} 0 & -\frac{3\alpha}{2r} & 0 \\ \frac{3\alpha}{2r} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \boldsymbol{D} = \begin{bmatrix} 1-\frac{3}{2}\alpha & 0 & 0 \\ 0 & 1-\frac{3}{2}\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}.\end{aligned} \tag{9}$$

If $\alpha < 0.4$, then the ball is sliding and we use the following matrices:

$$\begin{aligned}\boldsymbol{A} &= \begin{bmatrix} 0.6 & 0 & 0 \\ 0 & 0.6 & 0 \\ 0 & 0 & -k_{COR} \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 0 & \alpha r & 0 \\ -\alpha r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \boldsymbol{C} &= \begin{bmatrix} 0 & -0.6/r & 0 \\ 0.6/r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \boldsymbol{D} = \begin{bmatrix} 0.4 & 0 & 0 \\ 0 & 0.4 & 0 \\ 0 & 0 & 1 \end{bmatrix}.\end{aligned} \tag{10}$$

Wang et al. [41] showed that, though the ball spin can be approximated by the curve in the trajectory due to the Mangus effect, it can be further refined from the change in direction after the bounce leading to accurate spin estimation. Though we do not have access to the 3D ball trajectory, we can observe the ball's curve and the change in direction after the bounce. Since we take into account the spin in the physical model, we are able to infer the spin of the ball from

only the 2D ball trajectory. Pre-existing methods that infer spin from monocular recordings usually rely on the player stroke movement [19, 31].

5.2. Reconstruction

From the trajectory segmentation, we extracted the exact image position and timestamp of the ball's bounce on the table. By computing the intersection of the image ray corresponding to the bounce position with the 3D table plane, we determine the precise 3D bounce position of the ball, $\boldsymbol{p}_0$. Unlike previous methods [12, 21, 32], which initialize their models at the start of the trajectory, we use the bounce as the initial state. This approach eliminates the need to optimize for the ball's position and instead focuses on optimizing its velocity and spin immediately before the bounce, represented as $\boldsymbol{v}^-$ and $\boldsymbol{\omega}^-$. Using a bounce model, we compute the ball's velocity $\boldsymbol{v}^+$ and spin $\boldsymbol{\omega}^+$ just after the bounce. We then estimate the 3D positions of the ball before and after the bounce, denoted as $\hat{\boldsymbol{X}}_{\boldsymbol{k}}$. By optimizing $[\boldsymbol{v}^-, \boldsymbol{\omega}^-]$ for the reprojection error, we reconstruct the 3D trajectory of the ball. We use IPopt [39] in Casadi [3] to optimize our loss function, defined as:

$$\mathcal{L}_{ball} = \sum_{k=0}^{n} ||\boldsymbol{x}_{\boldsymbol{k}} - \hat{\boldsymbol{x}_{\boldsymbol{k}}}||_2^2 = \sum_{k=0}^{n} ||\boldsymbol{x}_{\boldsymbol{k}} - \boldsymbol{P}\hat{\boldsymbol{X}}_{\boldsymbol{k}}||_2^2 \quad (11)$$

where $\boldsymbol{x}_{\boldsymbol{k}}$ represents the observed ball positions. The ball spin is initialized to $\boldsymbol{\omega}^- = [0, 0, 0]^T$ (rad/s) and the ball velocity to $\boldsymbol{v}^- = [0, \pm 5, -3]^T$ (m/s).

6. 3D Pose Estimation

So far, our primary focus has been on reconstructing the ball's 3D trajectory. While valuable on its own, this information becomes significantly more insightful when combined with player motion and stroke estimation. By linking the ball's trajectory to the player's movements and actions, we can gain a deeper understanding of gameplay dynamics, strategy, and shot execution.

To achieve this, we first track the players and estimate their 2D pose using RTMPose [15]. We then infer their 3D pose with MotionBERT [53]. However, since the estimated 3D pose is expressed in camera coordinates, it must be transformed into the world frame. Camera calibration provides the necessary rotation, but estimating the translation vector $\boldsymbol{T}$ and scale factor s remains a challenge.

We estimate $\boldsymbol{T}$ by minimizing the reprojection error between the detected 2D keypoints and the projected 3D pose in the world frame:

$$\mathcal{L}_{\text{pose}} = \sum_{k=1}^{17} \|\boldsymbol{q}_k - s\boldsymbol{K}(\boldsymbol{Q}_k + \boldsymbol{T})\|_2^2 + \lambda \mathcal{L}_{\text{floor}} \quad (12)$$

where $\boldsymbol{q}_k$ represents the observed 2D joint positions, $\boldsymbol{Q}_k$ denotes the inferred 3D joint positions in the camera frame,

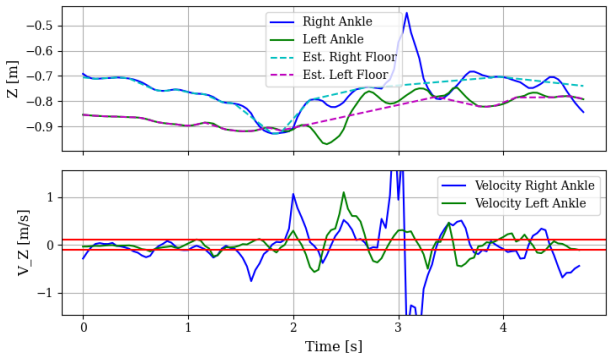


Figure 5. Floor contact estimation based on the vertical velocity of the players' ankles.

and s is the scale factor of the projection and $\lambda = 10$ is a weight coefficient. $\mathcal{L}_{floor}$ is an additional constraint to ensure that the player's feet touch the ground and that the depth information is correctly estimated. Indeed, although the 3D pose model accurately captures lateral motion, it struggles with depth estimation relative to the camera. This issue is particularly pronounced in side-view recordings, where precise depth perception is critical for reliable player tracking. We thus define $\mathcal{L}_{floor}$ as

$$\mathcal{L}_{\text{floor}} = \mathbb{1}_{\text{contact}}(t) \left(\|\boldsymbol{q}_{\text{left ankle}} - 0.1\|_2^2 + \|\boldsymbol{q}_{\text{right ankle}} - 0.1\|_2^2 \right), \quad (13)$$

where we approximate the ankle height as 0.1 m when the feet are touching the floor. We define floor contact as periods when the vertical velocity of the ankles remains below a predefined threshold, indicating contact. By constraining translation estimation to these stable frames, we obtain more accurate and consistent pose estimates. Jumps would otherwise cause the estimated pose to shift backward. For frames where the feet are not in contact with the floor, we interpolate the translation linearly to ensure smooth motion across the sequence. An example of floor contact estimation is shown in Figure 5.

7. Experiments

While qualitative assessment of our pipeline can be performed through visual inspection of the reconstructions, obtaining a quantitative evaluation is considerably more challenging due to the lack of video recordings with calibrated cameras, synchronized human motion capture and 3D ball tracking. Therefore, we evaluate the individual components of our pipeline to the best of our ability.

7.1. Camera Calibration

We first evaluated the accuracy of table corner detection. For this, we manually labeled the exact positions of the ta-

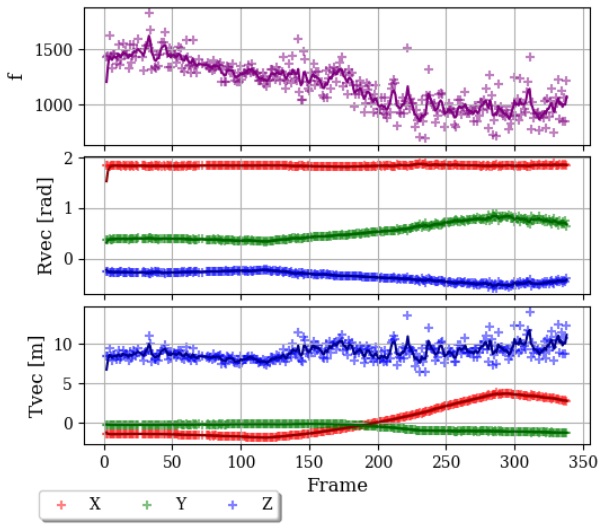


Figure 6. Continuous camera calibration from an online video. The darker lines are the filtered observations using a Kalman filter.

ble corners in 40 images distinct from the training set We achieved a Mean Absolute Error (MAE) of 2.39 $\pm$ 1.47 px.

We performed camera calibration on a video with camera movement and zooming [1]. The results, shown in Figure 6, demonstrate that the estimated parameters are temporally consistent. Additionally, slight variations in f closely align with changes in the Z component of $\boldsymbol{T}$, as their estimations are interdependent.

In the supplementary material, we also include multiple examples of successful camera calibration as well as examples of failure cases, which we go into more detail in Section 7.3.

7.2. 3D Ball Trajectory Reconstruction Benchmark

We recorded a table tennis match between two club-level players using a multi-camera system capable of tracking the ball's 3D position at 200 Hz. However, the system requires orange balls, which BlurBall was not trained to detect, rendering the ball detector inapplicable. Despite this limitation, the recorded 3D trajectories are accurate and can be used to generate simulated 2D observations at 25 fps using the pinhole camera model. To achieve this, we created 130 ball trajectories from side, oblique, and back views, introducing noise to the 2D ball position ($\sigma_{p_b} = 2$ px) and to the blur estimation ($\sigma_\theta = 6°$, $\sigma_l = 1$ px). The standard deviations were set according to the error observed in [4]. The evaluation results are presented in Table 1. We can observe that our rally reconstruction is robust to noise. This is because, with enough observations, the constraints imposed by the physics filter out the noise. While we can't directly validate the estimated bounce spin due to lack of ground truth, the low reconstruction error suggests it is reasonably accurate—since spinless trajectories would differ significantly, and topspin was applied during recording as instructed.

[1]First rally of https://www.youtube.com/watch?v=nd40lIYtQmA

Table 1. 3D Reconstruction errors using our method

POV	No Noise		Noise	
	Succ. [%]	MAE [cm]	Succ. [%]	MAE [cm]
Side	97.3	8.9	97.3	12.4
Oblique	91.5	14.5	89.9	17.1
Back	92.3	22.3	86.9	29.8

7.3. Limitations

While our camera calibration method is effective in most cases, it has certain limitations. Failures primarily occur due to player-induced occlusions, where body contours may be misinterpreted as table edges. This issue is particularly pronounced in back-view recordings, where the player frequently obstructs a significant portion of the table. However, since players move dynamically throughout a rally, clear views of the table naturally emerge, providing reliable opportunities for calibration. Additionally, if the camera's elevation angle is too low, the calibration process may become less reliable. In such cases, the segmentation model doesn't perform as well because of the thin shape of the mask and the intersection of detected table edges becomes highly sensitive to minor errors, increasing the likelihood of calibration failure.

While the ball detection method remains effective with moving cameras, the trajectory segmentation does not, as the ball's observed motion no longer follows a polynomial trajectory. Furthermore, our segmentation approach relies on second-degree polynomials, requiring the ball to be visible for at least three frames after a bounce. In some cases, this condition is not met due to occlusions caused by players or when the ball is hit back early after bouncing, limiting the effectiveness of the trajectory segmentation process.

8. Conclusion

We developed a method capable of reconstructing the 3D trajectory of a table tennis ball from an online recording. Moreover, the camera calibration is performed automatically compared to previous approaches. With our approach, we can make use of the vast amount of table tennis match recordings available on the internet. This can help get better player statistics or build a table tennis foundation model that could for example predict the next likely stroke or real-time win probability. While our method has been tested exclusively on table tennis, it could be easily adapted for other racket sports like tennis or pickleball.

ACKNOWLEDGMENT

Thank you to Dieter Buechler and Jan Schneider for their help with the recording setup and to Giorgio Becherini and Chuyu Yang for participating in the recordings.

References

[1] How To Experience More Time In Tennis | Feel Tennis. https://www.feeltennis.net/experience-more-time/. 2

[2] J. A. The fastest sport? Table Tennis VS Badminton, 2017. 2

[3] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 2018. 7

[4] Anonymous. Blurball: ball detection with blur estimation, 2025. 2, 5, 8

[5] M. Archana and M. Kalaisevi Geetha. Object Detection and Tracking Based on Trajectory in Broadcast Tennis Video. *Procedia Computer Science*, 58:225–232, 2015. 2

[6] H. Bao, X. Chen, Z. Wang, M. Pan, and F. Meng. Bouncing model for the table tennis trajectory prediction and the strategy of hitting the ball. In *2012 IEEE International Conference on Mechatronics and Automation*, pages 2002–2006, 2012. 6

[7] J. Calandre, R. Péteri, L. Mascarilla, and B. Tremblais. Extraction and analysis of 3D kinematic parameters of Table Tennis ball from a single camera. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 9468–9475, Milan, Italy, 2021. IEEE. 2, 3, 5

[8] Hua-Tsung Chen, Ming-Chun Tien, Yi-Wen Chen, Yi-Wen Chen, Wen-Jiin Tsai, and Suh-Yin Lee. Physics-based ball tracking and 3D trajectory reconstruction with applications to shooting location estimation in basketball video. *Journal of Visual Communication and Image Representation*, 20(3): 204–216, 2009. 2

[9] Hua-Tsung Chen, Wen-Jiin Tsai, Wen-Jiin Tsai, Suh-Yin Lee, and Jen-Yu Yu. Ball tracking and 3D trajectory approximation with applications to tactics analysis from single-camera volleyball sequences. *Multimedia Tools and Applications*, 60(3):641–667, 2012. 2

[10] Yu-Jou Chen and Yu-Shuen Wang. Tracknetv3: Enhancing shuttlecock tracking with augmentations and trajectory rectification. In *Proceedings of the 5th ACM International Conference on Multimedia in Asia*, New York, NY, USA, 2024. Association for Computing Machinery. 2

[11] J. Duan, Q. Wang, and Y. Wang. HOPS: A Fast Algorithm for Segmenting Piecewise Polynomials of Arbitrary Orders. *IEEE Access*, 9:155977–155987, 2021. 5

[12] M.H. Ertner, S. S. Konglevoll, M. Ibh, and S. Graßhof. SynthNet: Leveraging Synthetic Data for 3D Trajectory Estimation from Monocular Video. In *Proceedings of the 7th ACM International Workshop on Multimedia Content Analysis in Sports*, pages 51–58, New York, NY, USA, 2024. Association for Computing Machinery. 2, 3, 5, 7

[13] Y. Huang, I. Liao, C. Chen, T. İk, and W. Peng. TrackNet: A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications. In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–8, 2019. 2

[14] C. Hung. A Study of Automatic and Real-Time Table Tennis Fault Serve Detection System. *Sports*, 6(4):158, 2018. 2

[15] Tao Jiang, Peng Lu, Li Zhang, Ningsheng Ma, Rui Han, Chengqi Lyu, Yining Li, and Kai Chen. Rtmpose: Real-time multi-person pose estimation based on mmpose. *arXiv preprint arXiv:2303.07399*, 2023. 7

[16] Paresh R. Kamble, Avinash G. Keskar, Avinash G. Keskar, and Kishor M. Bhurchandi. Ball tracking in sports: A survey. *Artificial Intelligence Review*, 52(3):1655–1705, 2019. 1

[17] Ekaterina Kanaeva, Lev Gurevich, and Alexander Vakhitov. Camera pose and focal length estimation using regularized distance constraints. In *BMVC*, pages 162–1, 2015. 3

[18] K. Kulkarni, R. Jamadagni, J. Paul, and S. Shenoy. Table Tennis Stroke Detection and Recognition Using Ball Trajectory Data. *SSRN Electronic Journal*, 2022. 2

[19] Kaustubh Milind Kulkarni and Sucheth Shenoy. Table tennis stroke recognition using two-dimensional human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4576–4584, 2021. 7

[20] Guillaume Laffaye, Michael Phomsoupha, and Frédéric Dor. Changes in the Game Characteristics of a Badminton Match: A Longitudinal Study through the Olympic Game Finals Analysis in Men's Singles. *Journal of Sports Science & Medicine*, 14(3):584–590, 2015. 2

[21] P. Liu and J. Wang. MonoTrack: Shuttle trajectory reconstruction from monocular badminton video. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3512–3521, New Orleans, LA, USA, 2022. IEEE. 2, 3, 5, 7

[22] J. Matas, C. Galambos, and J. Kittler. Robust Detection of Lines Using the Progressive Probabilistic Hough Transform. *Computer Vision and Image Understanding*, 78(1):119–137, 2000. 4

[23] Juergen Metzler and Frank Pagel. 3D Trajectory Reconstruction of the Soccer Ball for Single Static Camera Systems. In *International Conference on Machine Vision Applications*, Kyoto, Japan, 2013-05-20/2013-05-23. 2

[24] Katharina Muelling, Abdeslam Boularias, Betty Mohler, Bernhard Schölkopf, and Jan Peters. Learning strategies in table tennis using inverse reinforcement learning. *Biological Cybernetics*, 108(5):603–619, 2014. 1

[25] B. T. Naik, M. F. Hashmi, and N.D. Bokde. A Comprehensive Review of Computer Vision in Sports: Open Issues, Future Trends and Research Directions. *Applied Sciences*, 12(9):4429, 2022. 1

[26] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407, 2011. 2

[27] Adrian Penate-Sanchez, Juan Andrade-Cetto, and Francesc Moreno-Noguer. Exhaustive linearization for robust camera pose and focal length estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(10):2387–2400, 2013. 3

[28] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing Network Design Spaces. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10425–10433, 2020. 4

[29] Arjun Raj, Lei Wang, and Tom Gedeon. TrackNetV4: Enhancing Fast Sports Object Tracking with Motion Attention Maps, 2024. 2

[30] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos, 2024. 4

[31] Soichiro Sato and Masaki Aono. Leveraging human pose estimation model for stroke classification in table tennis. In *Working Notes Proceedings of the MediaEval 2020 Workshop, Online, 14-15 December 2020*. CEUR-WS.org, 2020. 7

[32] L. Shen, Q. Liu, L. Li, and H. Yue. 3D reconstruction of ball trajectory from a single camera in the ball game. In *Proceedings of the 10th International Symposium on Computer Science in Sports (ISCSS)*, pages 33–39, Cham, 2016. Springer International Publishing. 2, 3, 5, 7

[33] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248, Cham, 2017. Springer International Publishing. 4

[34] N. Sun, Y. Lin, S. Chuang, T. Hsu, D. Yu, H. Chung, and T. İk. TrackNetV2: Efficient Shuttlecock Tracking Network. In *2020 International Conference on Pervasive Artificial Intelligence (ICPAI)*, pages 86–91, 2020. 2

[35] Satoshi Suzuki and KeiichiA be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985. 4

[36] S. Tamaki and H. Saito. Reconstruction of 3D Trajectories for Performance Analysis in Table Tennis. In *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1019–1026, 2013. 2

[37] S. Tarashima, M. Haq, Y. Wang, and N. Tagawa. Widely applicable strong baseline for sports ball detection and tracking. In *34th British Machine Vision Conference 2023, BMVC 2023, Aberdeen, UK, November 20-24, 2023*. BMVA, 2023. 2

[38] J. Tebbe, Y. Gao, M. Sastre-Rienietz, and A. Zell. A table tennis robot system using an industrial kuka robot arm. In *Pattern Recognition: 40th German Conference, GCPR 2018, Stuttgart, Germany, October 9-12, 2018, Proceedings 40*, pages 33–45. Springer, 2019. 2

[39] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006. 7

[40] Ping Wang, Qian Zhang, Yinli Jin, and Feng Ru. Studies and simulations on the flight trajectories of spinning table tennis ball via high-speed camera vision tracking system. *Proceedings of the Institution of Mechanical Engineers, Part P*, 233 (2):210–226, 2019. 6

[41] Yuxin Wang, Zhiyong Sun, Yongle Luo, Haibo Zhang, Wen Zhang, Kun Dong, Qiyu He, Qiang Zhang, Erkang Cheng, and Bo Song. A Novel Trajectory-Based Ball Spin Estimation Method for Table Tennis Robot. *IEEE Transactions on Industrial Electronics*, pages 1–11, 2023. 5, 6

[42] P. Wong, H. Myint, L. Dooley, and A. Hopgood. A multi-view automatic table tennis umpiring framework. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, page 175433712311714, 2023. 1, 2

[43] Changchang Wu. P3.5P: Pose estimation with unknown focal length. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2440–2448, 2015. 3

[44] Erwin Wu, Florian Perteneder, and Hideki Koike. Real-time Table Tennis Forecasting System based on Long Short-term Pose Prediction Network. In *SIGGRAPH Asia 2019 Posters*, pages 1–2, New York, NY, USA, 2019. Association for Computing Machinery. 1

[45] Xinguo Yu, Xinguo Yu, Nan Jiang, Nianjuan Jiang, Loong-Fah Cheong, Loong-Fah Cheong, Hon Wai Leong, Wai Leong, Xiaogang Yan, and Xin Yan. Automatic camera calibration of broadcast tennis video with applications to 3D virtual content insertion and ball detection and tracking. *Computer Vision and Image Understanding*, 113(5): 643–652, 2009. 2

[46] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv:2406.09414*, 2024. 4

[47] Gabriel Van Zandycke and Christophe De Vleeschouwer. 3D Ball Localization From A Single Calibrated Image. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3471–3479. IEEE Computer Society, 2022. 2

[48] Haotian Zhang, Ye Yuan, Viktor Makoviychuk, Yunrong Guo, Sanja Fidler, Xue Bin Peng, and Kayvon Fatahalian. Learning physically simulated tennis skills from broadcast videos. *ACM Trans. Graph.* 1

[49] Ce Zheng, Wenhan Wu, Chen Chen, Taojiannan Yang, Sijie Zhu, Ju Shen, Nasser Kehtarnavaz, and Mubarak Shah. Deep Learning-based Human Pose Estimation: A Survey. *ACM Comput. Surv.*, 56(1):11:1–11:37, 2023. 1

[50] Yinqiang Zheng and Laurent Kneip. A Direct Least-Squares Solution to the PnP Problem with Unknown Focal Length. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1790–1798, Las Vegas, NV, USA, 2016. IEEE. 3

[51] Yinqiang Zheng, Shigeki Sugimoto, Imari Sato, and Masatoshi Okutomi. A General and Simple Method for Camera Pose and Focal Length Determination. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 430–437, Columbus, OH, USA, 2014. IEEE. 3

[52] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. UNet++: A Nested U-

Net Architecture for Medical Image Segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 3–11, Cham, 2018. Springer International Publishing. 4

[53] Wentao Zhu, Xiaoxuan Ma, Zhaoyang Liu, Libin Liu, Wayne Wu, and Yizhou Wang. Motionbert: A unified perspective on learning human motion representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 7