OPEN ACCESS
OPUS AUGSBURG

Universität Augsburg
Universitätsbibliothek

Towards ball spin and trajectory analysis in table tennis broadcast videos via physically grounded synthetic-to-real transfer

Daniel Kienzle, Robin Schön, Rainer Lienhart, Shin'Ichi Satoh

Angaben zur Veröffentlichung / Publication details:

Kienzle, Daniel, Robin Schön, Rainer Lienhart, and Shin'Ichi Satoh. 2025. "Towards ball spin and trajectory analysis in table tennis broadcast videos via physically grounded synthetic-to-real transfer." In 2025 IEEE/CVF International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 11-12 June 2025, Nashville, TN, USA, 5832–41. Piscataway, NJ: IEEE. https://doi.org/10.1109/CVPRW67362.2025.00582.

Nutzungsbedingungen / Terms of use: **licgercopyright**

Dieses Dokument wird unter folgenden Bedingungen zur Verfügung gestellt: / This document is made available under these conditions:
Deutsches Urheberrecht
Weitere Informationen finden Sie unter: / For more information see:
https://www.uni-augsburg.de/de/organisation/bibliothek/publizieren-zitieren-archivieren/publiz/

Towards Ball Spin and Trajectory Analysis in Table Tennis Broadcast Videos via Physically Grounded Synthetic-to-Real Transfer

Daniel Kienzle[1,2] Robin Schön[1] Rainer Lienhart[1] Shin'Ichi Satoh[2,3]

[1]University of Augsburg, Germany, {firstname.lastname}@uni-a.de

[2] National Institute of Informatics, Japan

[3] University of Tokyo, Japan

Abstract

Analyzing a player's technique in table tennis requires knowledge of the ball's 3D trajectory and spin. While, the spin is not directly observable in standard broadcasting videos, we show that it can be inferred from the ball's trajectory in the video. We present a novel method to infer the initial spin and 3D trajectory from the corresponding 2D trajectory in a video. Without ground truth labels for broadcast videos, we train a neural network solely on synthetic data. Due to the choice of our input data representation, physically correct synthetic training data, and using targeted augmentations, the network naturally generalizes to real data. Notably, these simple techniques are sufficient to achieve generalization. No real data at all is required for training. To the best of our knowledge, we are the first to present a method for spin and trajectory prediction in simple monocular broadcast videos, achieving an accuracy of ***92.0 %*** *in spin classification and a 2D reprojection error of* ***0.19 %*** *of the image diagonal.*

1. Introduction

Computer vision is widely used across various sports to enhance athlete performance, analyze opponents' strategies [29, 37], and assist referees with automated decision-making such as goal-line technology in soccer or line-calling in tennis [21, 44]. Additionally, it provides valuable insights for sports broadcasting.

While player movement analysis is crucial in many sports, table tennis can rely mostly on ball trajectories. Notably, spin plays a key role in understanding gameplay, making its estimation essential for detailed performance analysis. This work especially focuses on predicting ball spin besides its 3D trajectory from standard table tennis broadcast videos.

We propose a learning-based method to estimate the ball's spin and 3D trajectory, enabling a comprehensive analysis of its kinematics. Broadcast video analysis is challenging due to low frame rates, small ball size, and motion blur. Since no public ground truth exists, we train our model exclusively on synthetic data. Instead of using raw visual data, we extract the ball's and table's 2D positions in the video frames, providing an abstract yet effective game-state representation. This representation in addition to physically correct synthetic data and targeted augmentations enables our model to generalize to real data.

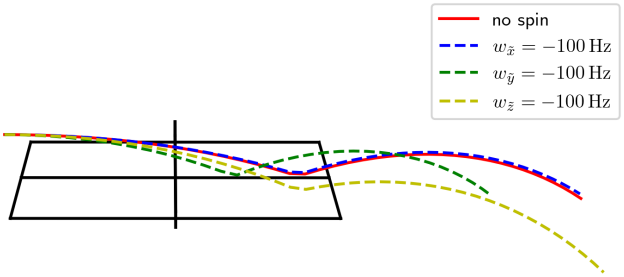


Figure 1. Simulated trajectory of the ball in the image plane under the influence of different spin components $\omega_{\tilde{x}}$, $\omega_{\tilde{y}}$, and $\omega_{\tilde{z}}$.

Our main contributions are:

- **Spin & trajectory estimation**: We introduce a novel approach for analyzing table tennis gameplay by predicting ball spin and 3D trajectory. To our knowledge, this is the first spin estimation method applicable to standard monocular broadcast videos.
- **Synthetic-to-real generalization**: Our model generalizes to real-world data despite being trained solely on synthetic data. Three simple techniques are sufficient to achieve excellent generalization: Using a smart input representation, utilizing a physics-based simulation for the synthetic data, and implementing suitable augmentations.
- **Synthetic train data & simulation pipeline**: We generate a large dataset of synthetic trajectories and publish both the dataset and simulation pipeline.
- **Annotated real-world test data**: We have manually annotated broadcast videos to evaluate real-world general-

ization. We make this dataset publicly available.

2. Related Work

3D Trajectory Estimation Ball trajectory analysis is crucial across various sports, with many methods relying on expensive multi-camera setups for triangulation [31, 34, 36, 45]. While highly accurate, this is costly and the data is often inaccessible. In contrast, we focus on monocular broadcast videos, making our method widely applicable.

Some works estimate a ball's 3D position in individual frames using observed size or height cues [6, 25, 42]. [7] applies this to table tennis but requires extremely high-frame-rate footage. In total, single-frame visual cues are generally insufficient for a precise localization. Instead, we uplift 2D detections over time to infer the full 3D trajectory.

Several methods fit physical models to observed 2D trajectories for 3D estimation in sports like badminton [27], volleyball [22], and basketball [8]. These typically require camera calibration and estimated turning points, which can introduce errors. Rather than performing an optimization for each trajectory, we train a neural network once to directly predict the 3D trajectory in an end-to-end approach, improving stability by performing camera calibration only indirectly and eliminating turning point estimates.

Deep learning enables direct 2D-to-3D mapping. Similar to our work, [13] uses synthetic 2D trajectories for training, predicting the initial 3D position of a tennis ball before applying a physics model. However, their approach overlooks ball spin and complex interactions like bouncing and the Magnus effect. In contrast, our physics simulation provides more realistic training data, and we predict the full 3D trajectory instead of the initial conditions to avoid cumulative errors in motion equation solutions.

Spin Estimation While spin is crucial in table tennis, it is not directly observable in standard broadcasts. To our knowledge, we present the first method for spin estimation from monocular broadcast videos.

Existing approaches often rely on specialized hardware: Event cameras [18, 33], high-speed cameras detecting ball markings [17], or multi-camera setups recognizing logos [48]. While effective, these methods require controlled environments or ball modifications, making them impractical for real-world broadcast analysis. Our method overcomes these limitations by estimating spin from standard video footage without additional equipment.

Physics in Computer Vision Previous works have e.g. used physics-based losses for 3D localization [25], simulated ball trajectories for trajectory completion [2], and physical models to estimate landing positions in golf [32]. In table tennis, [9] applies physics-informed neural networks (PINNs) [35] to multi-camera 3D trajectory data to extract ball motion properties, and [41, 46] predict the future trajectory based on previous 3D measurements. Moreover, [10] trains a robotic player entirely on synthetic data. Our approach leverages the physical simulation environment from [10] to generate synthetic data, enabling our model to generalize to real-world broadcast videos.

3. Problem Description

To effectively analyze a player's technique in table tennis, it is essential to understand the ball's trajectory and the initial spin imparted upon contact with the paddle. This section first outlines the setup of our method and discusses the key design choices. We then introduce appropriate coordinate systems for analyzing both trajectory and spin. Finally, we examine how spin influences the trajectory and identify which spin components can be accurately observed.

3.1. Main Goal

We define the ball's 3D position at each video frame i at time t_i as $\vec{r}(t_i)$. Similarly, we denote the ball's initial spin as $\vec{\omega} = \vec{\omega}(t_0)$. Our model predicts both $\vec{r}(t_i)$ for each t_i and $\vec{\omega}$ once for the full trajectory.

The input to our model consists of the observed 2D pixel coordinates of the ball for each time t_i. We employ an end-to-end approach that does not require additional information such as camera calibration. To facilitate this, we also extract the 2D image coordinates of 13 key table points per frame, since they contain valuable information about the orientation of the camera. Thus, the input to our model comprises the ball's 2D trajectory along with the 2D positions of these table points. Figure 8 in the supplementary material illustrates these key table points in a single frame.

Rather than using raw video frames, our approach relies on a smart data representation based on extracted 2D coordinates of the ball and table points. This is a widely used technique in computer vision, commonly applied in fields such as 3D pose estimation [4, 12, 49], action recognition [11, 23], and sign language translation [15]. Since 2D keypoint extraction is a well-established field [24, 30, 47], we do not further elaborate on this step. In this paper we assume that the 2D keypoint extraction is already performed and focus on the subsequent steps.

A significant advantage of this smart data representation is that it enables our model to focus on the most relevant information while avoiding distractions from irrelevant visual details, which could lead to overfitting and reduced generalization. This way, the model's attention is naturally directed towards the key information, which is crucial for the success of our method. Furthermore, while generating realistic synthetic video data is challenging, it is relatively straightforward to simulate accurate synthetic trajectories and spin using physics-based models. Consequently, this data repre-

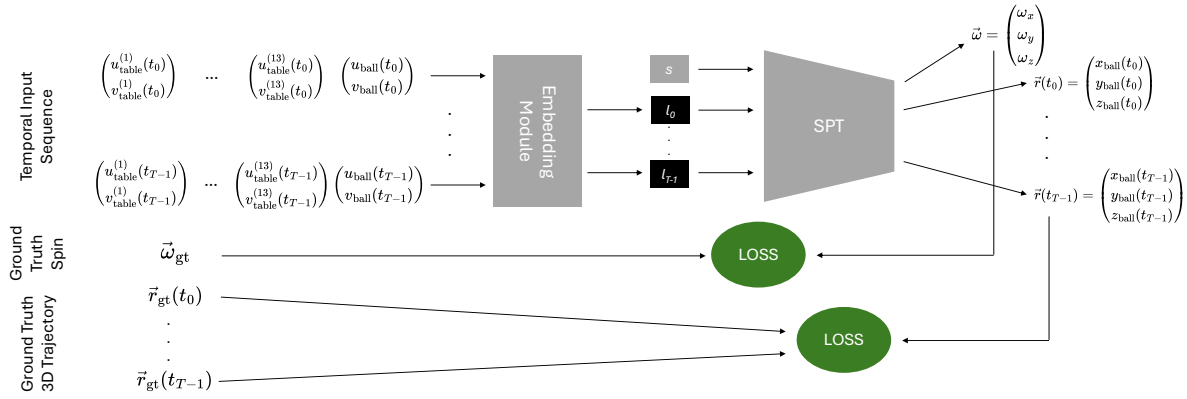


Figure 2. Overview of our pipeline. For each time step t_i, ball coordinates and table keypoints are embedded to generate location tokens l_i. A learnable spin token s is prepended, and the SPT processes the sequence $\{s, l_0, ..., l_{T-1}\}$. The SPT predicts the initial spin $\vec{\omega}$ at t_0 and the sequence of 3D ball positions $\{\vec{r}(t_0), ..., \vec{r}(t_{T-1})\}$. The predicted spin and trajectory are supervised using separate loss terms, ensuring accurate learning of both components.

sentation enables us to solely train the model on synthetic data and still achieve good generalization on real data.

A major challenge when working with broadcast footage is the absence of ground-truth annotations, making it impossible to train a model directly on real data. To address this, we generate a large dataset of physically accurate synthetic trajectories for training. To ensure a smooth transition from synthetic to real data, we employ an advanced physics simulation engine that accurately models the ball's interactions with the table. For this purpose, we use the MuJoCo simulation engine [40], utilizing the same simulation parameters as in [10]. Since [10] demonstrated that synthetic-to-real transfer is feasible in table tennis simulations, we are confident that their setup aligns well with our requirements.

3.2. Predictions and Coordinate Systems

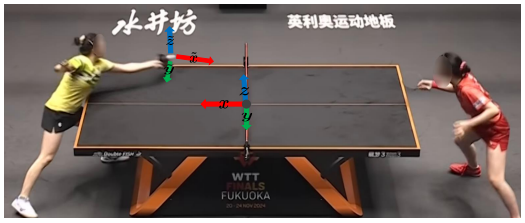


Figure 3. World coordinate system (x, y, z) and ball coordinate system ($\tilde{x}$, $\tilde{y}$, $\tilde{z}$). Both are orthogonal coordinate systems.

Our model predicts two quantities: The 3D locations of the ball $\vec{r}(t_i)$ and the initial spin $\vec{\omega}(t_0)$. To effectively describe these variables, we first define coordinate systems that facilitate both computation and human interpretation. We introduce two coordinate systems: The *world coordinate system* and the *ball coordinate system*, both illustrated in Figure 3.

The **world coordinate system**, remains fixed at the center of the table for each trajectory. The x- and y-axes lie within the table plane, while the z-axis points upwards. The unit vectors are defined as:

$$\vec{e}_x = \begin{pmatrix}1 & 0 & 0\end{pmatrix}^{\mathrm{T}}, \quad \vec{e}_y = \begin{pmatrix}0 & 1 & 0\end{pmatrix}^{\mathrm{T}}, \quad \vec{e}_z = \begin{pmatrix}0 & 0 & 1\end{pmatrix}^{\mathrm{T}}. \quad (1)$$

This system is well-suited for describing and analyzing the ball's trajectory, which is why our model predicts the **3D trajectory** in this coordinate system. However, while spin-related physical effects can be described in this system, interpreting individual spin components in an intuitive manner is difficult. For this reason, we introduce a second coordinate system optimized for human interpretation.

Unlike the world coordinate system, the **ball coordinate system** is adjusted once per trajectory. This system is used to describe the ball's **initial spin** in a more interpretable way. Its coordinate axes, denoted as $\tilde{x}$, $\tilde{y}$ and $\tilde{z}$, are defined as follows:

- The origin is set at the ball's position in the first frame.
- The $\tilde{x}$-axis aligns with the ball's initial movement direction in the x-y plane.
- The $\tilde{z}$-axis remains parallel to the z-axis, pointing upwards.
- The $\tilde{y}$-axis is orthogonal to both and lies in the x-y plane.

Mathematical, the unit vectors are:

$$\begin{aligned}\vec{\Delta} &= \begin{pmatrix}x(t_1) - x(t_0), & y(t_1) - y(t_0), & 0\end{pmatrix}^{\mathrm{T}}, \\ \vec{e}_{\tilde{x}} &= \frac{1}{\left|\vec{\Delta}\right|}\vec{\Delta}, \quad \vec{e}_{\tilde{z}} = \vec{e}_z, \quad \vec{e}_{\tilde{y}} = \vec{e}_{\tilde{z}} \times \vec{e}_{\tilde{x}}\end{aligned} \quad (2)$$

with t_i denoting the i-th timestep and $|\cdot|$ being the euclidean norm of a vector. While the coordinate system is static during a trajectory, its center and orientation is different for each trajectory.

If the trajectory is known, we can easily transform from one coordinate system to the other. For training we use the ground truth 3D trajectory for the transformation and during inference we rely on the 3D trajectory predictions.

The ball coordinate system is chosen such that the main direction of the ball's movement is aligned with the $\tilde{x}$ axis. While there is also some movement in the $\tilde{z}$ direction, there is nearly no movement in the $\tilde{y}$ direction. By adjusting the coordinate axes to the direction of the ball's initial movement, we ensure that the ball can be interpreted in a meaningful way. Especially the rotation around the $\tilde{y}$ axis, denoted as $\omega_{\tilde{y}}$, is of high interest as it describes the top- or backspin of the ball. In the next section, we will discuss the influence of the individual spin components in the ball coordinate system on the ball's trajectory in more detail.

3.3. Observability of Spin Components

Our model estimates the ball's initial spin solely from the observed 2D trajectory. Therefore, if a spin component significantly affects the trajectory, we expect the model to learn it effectively. The spin is characterized as a 3D vector $\vec{\omega}$ with its components describing the rotation around the corresponding axis of the ball coordinate system. It influences the trajectory in two primary ways:

- The **Magnus effect**: This force acts on a spinning ball with velocity $\vec{v}$, given by

$$\vec{F} = k_{\mathrm{M}} \cdot \vec{\omega} \times \vec{v} \tag{3}$$

  where k_{M} is a constant that depends on the ball's surface properties. This force causes a deviation of the ball's trajectory perpendicular to $\vec{v}$ and $\vec{\omega}$ and affects the ball during the entire flight.
- **Friction at Bounce**: When the ball contacts the table, spin-induced friction generates force:

$$\vec{F} = k_{\mathrm{F}} \cdot \omega_{\tilde{x}} \vec{e}_{\tilde{y}} + k_{\mathrm{F}} \cdot \omega_{\tilde{y}} \vec{e}_{\tilde{x}} \tag{4}$$

  where k_{F} is a constant describing the ball's and table's surface properties, and $\vec{e}_{\tilde{x}}$ and $\vec{e}_{\tilde{y}}$ are the unit vectors of the ball coordinate system. This force only appears directly at the bounce and is not present during the flight.

The Magnus effect has the most significant influence on the trajectory, as it affects the ball during the entire flight. As the main movement of the ball is in $\tilde{x}$ direction, only the orthogonal spin components $\omega_{\tilde{y}}$ and $\omega_{\tilde{z}}$ significantly influence the trajectory due to the cross product in Equation 3.

The friction force only appears during the bounce and, thus, has a smaller influence on the trajectory. It is governed by $\omega_{\tilde{x}}$ and $\omega_{\tilde{y}}$.

As a result, $\omega_{\tilde{y}}$ and $\omega_{\tilde{z}}$ strongly impact the trajectory, while $\omega_{\tilde{x}}$ has a minor effect. Therefore, we expect our model to learn the spin components $\omega_{\tilde{y}}$ and $\omega_{\tilde{z}}$ effectively, while the prediction of $\omega_{\tilde{x}}$ will be more challenging.

The spin around the $\tilde{y}$ axis, denoted as $\omega_{\tilde{y}}$, describes the top- or backspin of the ball. This component is easily interpretable and, therefore, of great importance in our analyses.

In Figure 1 we illustrate the effect of the spin components on the ball's trajectory in the image. We simulate the trajectory of the ball, setting the same initial position and velocity for each trajectory, and only varying a single spin component in each case. For each trajectory, we set one spin component to -100 Hz. While the spin components $\omega_{\tilde{y}}$ and $\omega_{\tilde{z}}$ have a significant influence on the trajectory, $\omega_{\tilde{x}}$ has a negligible influence on the trajectory, which supports our previous discussion.

4. Method

We train a neural network to predict the ball's 3D trajectory and initial spin from its 2D trajectory. Each trajectory usually goes from the table tennis shot over a single touchdown on the tennis table to the next touch. The general pipeline of our method is shown in Figure 2.

4.1. Training Objective

Our neural network is trained using a large dataset of simulated synthetic data, enabling **direct supervision with synthetic ground truth**. The loss function for predicting the 3D trajectory is defined as:

$$\mathcal{L}_{\text{trajectory}} = \frac{1}{T} \sum_{i=0}^{T-1} \left\| \vec{r}_{\text{pred}}(t_i) - \vec{r}_{\text{gt}}(t_i) \right\|^2 \tag{5}$$

where $\vec{r}_{\text{pred}}(t_i)$ represents the predicted 3D position of the ball at time t_i, and $\vec{r}_{\text{gt}}(t_i)$ denotes the corresponding ground truth position. The loss is averaged over all time steps T. To evaluate the accuracy of the predicted initial spin, we define the spin loss as:

$$\mathcal{L}_{\text{spin}} = \left\| \vec{\omega}_{\text{pred}} - \vec{\omega}_{\text{gt}} \right\|^2 \tag{6}$$

where $\vec{\omega}_{\text{pred}}$ and $\vec{\omega}_{\text{gt}}$ are the predicted and ground truth initial spins, respectively. Consequently, the total loss is given by

$$\mathcal{L} = \mathcal{L}_{\text{trajectory}} + \mathcal{L}_{\text{spin}} \,. \tag{7}$$

This formulation ensures that both the trajectory prediction and initial spin estimation are optimized simultaneously. Because both losses are in the same order of magnitude, we do not introduce additional weighting factors.

4.2. Base Architecture

Our proposed architecture consists of an *embedding module* and a *Spin Prediction Transformer (SPT)*. Given a sequence of length T, the model takes as input the set of the 2D ball positions and 13 table keypoints at each time t_i.

The embedding module transforms this 2D information into a d-dimensional *location token* $l_i \in \mathbb{R}^d$ for each t_i. Additionally, a learnable *spin token* $s \in \mathbb{R}^d$ is prepended, resulting in the sequence $\{s, l_0, ..., l_{T-1}\} \in \mathbb{R}^{(T+1)\times d}$ that is then processed by the SPT.

After the final transformer layer, a *position head* is applied to each transformed location token individually, predicting the 3D positions $\{\vec{r}(t_0), ..., \vec{r}(t_{T-1})\} \in \mathbb{R}^{T\times 3}$. Similarly, a *spin head* is applied to s, predicting the initial spin $\vec{\omega} \in \mathbb{R}^3$. The overall architecture is illustrated in Figure 2.

Token Embeddings We explore three embedding strategies for computing the location tokens l_i:

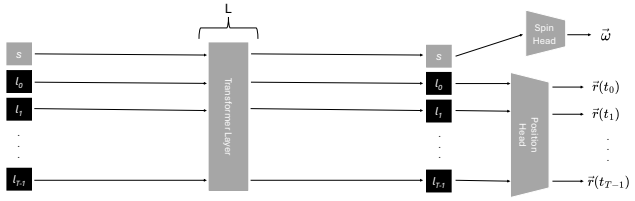
- **Concatenation Method:** Concatenates the 2D coordinates of all 14 points (ball position + 13 table points) into a single vector, followed by an MLP with one hidden layer to obtain the location token l_i.
- **Dynamic Method:** Treats each of the 14 points as a separate token, projects them into a d-dimensional space using an MLP with one hidden layer, and condenses the information via a 4-layer transformer encoder. The transformed ball position token is used as l_i, discarding the other tokens.
- **Context-Free Method:** Uses only the 2D ball position as input. An MLP with one hidden layer projects the ball position into the d-dimensional space to obtain the location token l_i. This method serves as a baseline to check if the information from the table keypoints is needed.

These methods are visualized in Figure 7 of the supplementary material.

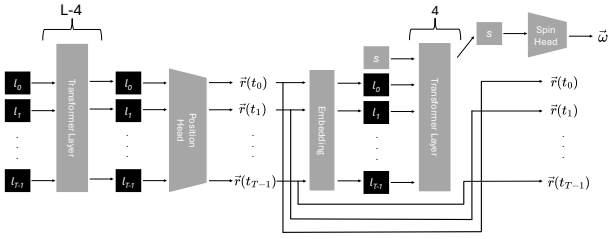
Spin Prediction Transformer The SPT is an encoder-only transformer with L layers. It processes the sequence $\{s, l_0, ..., l_{T-1}\}$ and outputs $T + 1$ tokens. A position head is applied to each transformed location token l_i for predicting the 3D positions $\vec{r}(t_i) \in \mathbb{R}^3$, while the spin head processes the spin token s to predict the initial spin $\vec{\omega} \in \mathbb{R}^3$.

We explore three SPT architectures (see Figure 4):

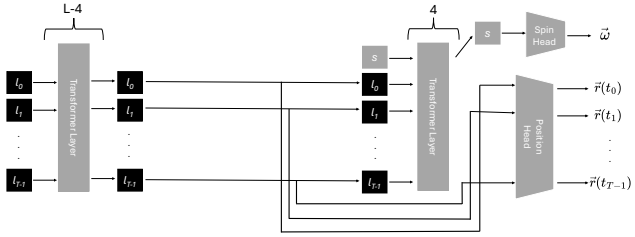
- **Single-Stage Model:** Prepends the spin token to the location tokens before the first transformer layer. It predicts positions and spin jointly by processing all tokens together. However, this approach does not explicitly enforce a dependency between trajectory and spin.
- **Two-Stage Model:** Enforces a physical bottleneck by first predicting 3D positions $\vec{r}(t_i) \in \mathbb{R}^3$ with $L - 4$ transformer layers (no spin token is prepended), then using these positions as the only input for spin prediction in a second stage (4-layer transformer). Since spin and trajectory are directly linked, spin alters the trajectory and trajectory encodes spin information, our model explicitly captures this dependency. While this architecture mimics this physical connection, any noise in the predicted positions directly affects spin accuracy.
- **Connect-Stage Model:** Addresses the rigid bottleneck in the two-stage model by using the transformed tokens $l_i \in \mathbb{R}^d$ before the application of the position head as input of the second stage. This softens the spin's dependency on the trajectory prediction, while still maintaining the physically motivated structure.



(a) Single-Stage Model



(b) Two-Stage Model



(c) Connect-Stage Model

Figure 4. Illustration of the 3 different SPT architectures. In the single-stage model, the trajectory is predicted jointly with the spin. The two-stage model predicts only the 3D positions $\vec{r}(t_i) \in \mathcal{R}^3$ in the first stage and uses these predictions to estimate the spin in the second stage. The connect-stage model uses the transformed tokens $l_i \in \mathcal{R}^d$ as input for the second stage.

Further implementation details While the positions are always predicted in the world coordinate system, we can predict the initial spin either in the world or the ball coordinate system. If predicted in the world coordinate system, it is converted to the ball coordinate system, which is described by Equation 2.

We use Rotary Positional Embeddings (RoPE) [38], which have been recently gaining popularity in the field of language processing [19, 20, 28], to encode the order of the tokens. Instead of adding absolute positional embeddings [43], a rotation is applied to the queries and keys before each attention computation. The strength of the rotation is dependent on the position in the sequence, thus the computed attention scores include positional information.

We implement and test different hyperparameters of the architecture and define various sizes of the model as shown in Table 4 of the supplementary material. We use a model with $L = 16$ and $d = 128$, referred to as *large* model, for all experiments.

5. Data

5.1. Synthetic Data Generation

We generate physically accurate table tennis trajectories using MuJoCo [40], with a particular focus on realistic ball bounces, which significantly impact data quality. Following [10], which trained a human-level robot using synthetic data alone, we adopt the same simulation parameters.

Trajectories are generated by sampling initial position, velocity, and spin, followed by simulating the trajectory. Only valid trajectories are kept. A valid trajectory starts on the left side, bounces once on the opponent's side, and ends on the right, ensuring full visibility in the camera frame. We collect **50,000** valid trajectories, split into **70%** training, **10%** validation, and **20%** test set. To ensure robustness, we vary camera parameters during training while using real-data camera parameters for validation and testing. We illustrate some sampled camera parameters in Figure 10 of the supplementary material.

As ground-truth 3D trajectories and rotations are available in synthetic data, we compute the *spin error* as

$$\Delta\vec{\omega} = \frac{1}{N}\sum_{j=1}^{N} ||\vec{\omega}_{\text{pred},j} - \vec{\omega}_{\text{gt},j}||_2 \tag{8}$$

and the *3D trajectory error* as

$$\Delta\vec{r}_{\text{world}} = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{T_j}\sum_{i=0}^{T_j-1} ||\vec{r}_{\text{pred},j}(t_i) - \vec{r}_{\text{gt},j}(t_i)||_2 \tag{9}$$

where N is the number of trajectories, j indexes the different trajectories, t_i indexes time steps of a trajectory, T_j is the length of trajectory j, and $||\cdot||_2$ is the Euclidean vector norm.

5.2. Real Data

Although training is solely conducted on synthetic data, we assess generalization on real table tennis broadcasts. We manually annotate **50** 2D ball trajectories from six WTTF matches, totaling **1197** annotated frames. Table keypoints are annotated once per trajectory and reused across frames. The videos have a 2560×1440 resolution at 50 Hz.

While obtaining 3D ground truth for real data is infeasible, we manually label spin direction (top-/backspin) by analyzing paddle positions in the image (Figure 9, supplementary material). We evaluate classification accuracy as

$$acc = \frac{1}{N}\sum_{j=1}^{N} \delta_{c_j,\text{sign}(\omega_{\tilde{y},j})} \tag{10}$$

where c_j is the annotated class (1 for topspin, -1 for backspin), $\omega_{\tilde{y},j}$ is the predicted spin, $\text{sign}(\cdot)$ is the sign function, and $\delta_{a,b}$ is 1 if $a = b$, otherwise 0. Similarly, we also compute the macro F1-score.

To convert the estimated $\omega_{\tilde{y}}$ into a binary classification, we set a fixed threshold at $\omega_{\tilde{y}} = 0$, which is physically justified since the sign of $\omega_{\tilde{y}}$ determines the spin direction. However, this strict threshold may overly penalize trajectories with weak spin, where small prediction errors lead to entirely incorrect classifications. In standard binary classification tasks, the ROC-AUC metric [14] avoids reliance on a fixed threshold, making it a more robust performance measure. While accuracy and F1-score remain our primary metrics due to the known threshold, we also evaluate ROC-AUC, as it provides a more nuanced assessment, especially for predictions near zero. As we will see in the experiments, the key threshold values in the ROC-AUC computation are indeed close to zero, further supporting its relevance (see Figure 5b)

For trajectory evaluation, we project predicted 3D paths into 2D image coordinates and compute the *2D reprojection error* $\Delta\vec{r}_{\text{img}}$:

$$\Delta\vec{r}_{\text{img}} = \tfrac{1}{D}\tfrac{1}{N}\sum_{j=1}^{N}\tfrac{1}{T_j}\sum_{i=0}^{T_j-1} ||\mathcal{P}\left(\vec{r}_{\text{pred},j}(t_i)\right) - \vec{r}_{\text{gt2D},j}(t_i)||_2 \tag{11}$$

where $\mathcal{P} \in \mathcal{R}^{2\times3}$ projects 3D coordinates into 2D image positions. The matrix $\mathcal{P}$ is estimated using the table point annotations (see Section C in the supplementary material). Instead of evaluating the absolute pixel error, we divide through the image diagonal length $D = \frac{1}{\sqrt{H^2+W^2}}$ (with $H \times W$ being the video resolution). Consequently, we obtain a relative error that is not dependent on the video resolution and, thus, allows for better human interpretability.

Despite lacking direct 3D annotations, our methodology enables indirect assessment of the generalization ability through classification and projected trajectory comparisons.

5.3. Data Augmentations

By using a smart data representation instead of raw visual data as input and utilizing physically correct synthetic data, we enable the model to generalize to real broadcast. However, three key challenges still introduce inaccuracies:

- **Motion Blur**: The ball exhibits strong motion blur, making precise localization difficult and introducing errors absent in synthetic data.
- **Sudden Trajectory Stops**: In real matches, the opponent's hit often terminates the trajectory suddenly, unlike in synthetic simulations where we do not simulate the opposing player.
- **Annotation Errors**: Manually labeled table points may contain slight inaccuracies, whereas synthetic training data is always precise.

To mitigate these effects, we introduce three augmentations: *motion blur*, *sudden trajectory stop*, and *gaussian blur*. The

implementation of the different augmentations is described in more detail in Section D of the supplementary material.

6. Results

Each model is implemented in PyTorch [3] and trained on a single NVIDIA H100 GPU. We use a fixed learning rate of 10^{-4} and a batch size of 64. Model weights are optimized with ADAM [26], and an Exponential Moving Average (EMA) with a decay of 0.999 [39] is applied. Training runs for 800 epochs, and we select the model with the best $\Delta\vec{\omega}$ score on the synthetic validation set.

We consistently use the large model with rotary positional encodings, predicting spin in the world coordinate system. Additional ablations on these parameters are provided in the supplementary material.

Our primary focus is on real-data performance, as generalization to real-world scenarios is crucial. Synthetic performance is not relevant for practical applications, but is included in all tables. We further discuss the significance of synthetic results in Section 6.1.

6.1. Evaluation of SPT Architectures

We compare the SPT architectures introduced in Section 4.2 without data augmentations, using the concatenation method for token embeddings. Results are shown in Table 1.

The single-stage model performs well for spin prediction on synthetic data but fails to generalize to real data. Its 3D trajectory predictions are also subpar. The two-stage model, in contrast, achieves strong results on real data despite weaker synthetic performance. The connect-stage model slightly lags behind the two-stage model in accuracy and macro F1 score but outperforms in ROC-AUC and 2D reprojection error ($\Delta\vec{r}_{\text{img}}$).

Method	Synthetic		Real			
	$\Delta\vec{\omega}\downarrow$	$\Delta\vec{r}_{\text{world}}\downarrow$	*acc* ↑	$F_1\uparrow$	ROC-AUC ↑	$\Delta\vec{r}_{\text{img}}\downarrow$
single-stage	11.7 Hz	35.2 cm	58.0 %	0.440	0.669	7.67 %
two-stage	56.4 Hz	4.6 cm	**80.0 %**	**0.799**	0.807	0.72 %
connect-stage	31.0 Hz	3.6 cm	74.0 %	0.740	**0.838**	**0.53 %**

Table 1. Comparison of different SPT architectures. The best results on the real data are highlighted in bold.

Both two-stage and connect-stage models generalize well. The two-stage model is preferable for spin classification, while the connect-stage model excels in 3D trajectory prediction. Conclusively, both models are suitable for our task. As the connect-stage model subjectively offers the best balance, we use it for further experiments.

Synthetic results do not reliably predict real-world performance. For instance, the single-stage model excels on synthetic data but fails on real data, whereas the two-stage model shows the opposite trend. This suggests that the physically motivated bottleneck in the two-stage and connect-stage models improves generalization. By constraining the model architecture, we enhance real-data applicability despite training solely on synthetic data. Hence, subsequent experiments focus exclusively on real-data performance.

6.2. Evaluation of Token Embedding Methods

We assess different token embedding methods discussed in Section 4.2 using the connect-stage architecture without data augmentations. Results are presented in Table 2.

The context-free method performs the worst across all metrics. Without table keypoints, it fails to capture sufficient scene context, confirming that the 2D trajectory alone is inadequate for an end-to-end method. In contrast, both dynamic and concatenation methods yield strong results, demonstrating the importance of table keypoints as input.

Method	Synthetic		Real			
	$\Delta\vec{\omega}\downarrow$	$\Delta\vec{r}_{\text{world}}\downarrow$	*acc* ↑	$F_1\uparrow$	ROC-AUC ↑	$\Delta\vec{r}_{\text{img}}\downarrow$
context free	41.5 Hz	54.9 cm	66.0 %	0.649	0.670	4.37 %
dynamic	27.7 Hz	3.3 cm	**84.0 %**	**0.836**	**0.911**	0.56 %
concatenation	31.0 Hz	3.6 cm	74.0 %	0.740	0.838	**0.53 %**

Table 2. Comparison of different embedding methods. The best results on the real data are highlighted in bold.

The dynamic method is best for spin classification, while the concatenation method slightly outperforms in 3D trajectory prediction. We attribute the dynamic method's success to its more complex embedding process, which leverages transformer layers to integrate keypoint and ball position data. However, this increases the parameter count, computational cost and results in sensitivity to hyperparameters as well as less stable training. Given its robustness and efficiency, we select the concatenation method for our further experiments.

6.3. Evaluation of Data Augmentations

We evaluate the three data augmentation techniques described in Section 5.3 using the connect-stage architecture with concatenation embeddings. Results are in Table 3.

All augmentations improve generalization to real data, with sudden end augmentation providing the most substantial gains across all metrics. Motion blur also enhances performance significantly, while Gaussian blur offers only slight improvements. This demonstrates that making synthetic data more realistic significantly boosts real-world performance.

Since each augmentation contributes positively, we combine them in the last row of Table 3. This results in a notable reduction in 2D reprojection error while maintaining strong

Method			Synthetic		Real			
motion blur	sudden end	gaus. blur	$\Delta\vec{\omega}\downarrow$	$\Delta\vec{r}_{\text{world}}\downarrow$	*acc* ↑	$F_1\uparrow$	ROC-AUC ↑	$\Delta\vec{r}_{\text{img}}\downarrow$
×	×	×	31.0 Hz	3.6 cm	74.0 %	0.740	0.838	0.53 %
✓	×	×	44.5 Hz	5.0 cm	88.0 %	0.875	0.969	0.55 %
×	✓	×	31.7 Hz	3.8 cm	**96.0 %**	**0.959**	**0.990**	0.34 %
×	×	✓	42.5 Hz	4.2 cm	80.0 %	0.800	0.898	0.64 %
✓	✓	✓	48.7 Hz	5.5 cm	92.0 %	0.917	**0.990**	**0.19 %**

Table 3. Comparison of different data augmentations. ✓ indicates the application of the augmentation, while × indicates the absence of the augmentation. The best results on the real data are highlighted in bold.

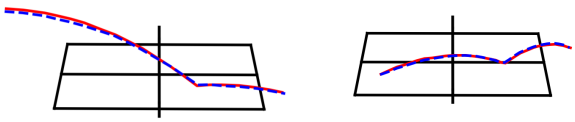


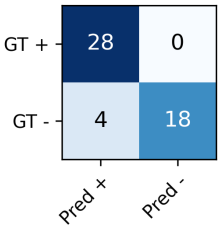
Figure 6. Comparison of reprojection of the predicted trajectory (dashed blue) with the annotated 2D trajectory (solid red). We present two examples on the real dataset.

spin prediction accuracy. This model gives a good trade-off between spin classification and 3D trajectory prediction and, thus, this combination is referred to as the *best model* in the following experiments.

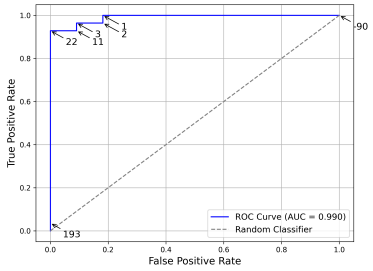
Although the model already performs well on real data without any augmentations, applying them further enhances performance. Our best model nearly achieves perfect results, indicating that augmentations effectively bridge the gap between synthetic and real data. They are the final step to enable generalization from synthetic training alone.

6.4. Detailed Evaluation of Best Model

In this section, we analyze the performance of our best model in greater detail. This model, corresponding to the last row in Table 3, employs the connect-stage architecture with concatenation token embeddings and incorporates all data augmentations during training. Our focus is on evaluating its effectiveness in spin classification and 3D trajectory prediction on real data.



(a) Confusion matrix for spin classification on real data. Topspin is **+** and backspin **-**.



(b) ROC curve for spin classification on real data. The arrows indicate the threshold values in Hz for the predicted $\omega_{\tilde{y}}$.

Figure 5. Confusion matrix and ROC plot for the best model on the real dataset.

Spin Classification Performance Figure 5 presents the confusion matrix and the ROC curve for spin classification on the real test dataset. The confusion matrix reveals that the model distinguishes well between topspin and backspin, though a slight bias towards topspin is noticeable. This is also reflected in the ROC curve, where the threshold values for predicted $\omega_{\tilde{y}}$ are subtly shifted towards topspin.

Despite this bias, the threshold values remain close to the physically correct threshold of 0 Hz, suggesting that the model's predictions are not significantly skewed. Even though we cannot evaluate the exact values of $\omega_{\tilde{y}}$ on the real data, the smooth and consistent behavior observed in the ROC thresholds indicates that the model produces reasonable and stable predictions. Overall, these results confirm the model's reliability in spin classification.

3D Trajectory Prediction Performance In Figure 6, we illustrate 2 predicted trajectories on the real dataset. To compare a predicted trajectory with the corresponding annotated 2D trajectory, we calculate the reprojection of the predicted 3D trajectory onto the 2D image. In both examples, the predicted trajectory closely follows the annotated ground truth, demonstrating strong alignment between the two. This suggests that the model effectively captures the underlying 3D motion of the ball and generalizes well to real-world data.

The high accuracy in trajectory prediction confirms that the model correctly generalizes to real data, showing its suitability for practical applications.

7. Conclusion

In this paper, we introduced a method for predicting the 3D trajectory and initial spin of a table tennis ball in broadcast videos. A key aspect of our approach is that the model is trained exclusively on synthetic data, yet it generalizes remarkably well to real-world footage. This strong generalization is achieved through a combination of a carefully designed data representation, physically grounded synthetic training data, and problem-specific data augmentations. Notably, our results demonstrate that these straightforward, yet effective steps are sufficient to bridge the gap between synthetic and real data. Our method enables detailed analysis of a player's technique using standard monocular RGB videos, making advanced performance evaluation more accessible. It can be applied both by professionals analyzing broadcast footage and by amateurs using a simple smartphone camera, broadening its usability across different levels of expertise.

References

[1] Y.I Abdel-Aziz, H.M. Karara, and Michael Hauck. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. *Photogrammetric Engineering and Remote Sensing*, 81(2):103–107, 2015. 12

[2] Jan Achterhold, Philip Tobuschat, Hao Ma, Dieter Büchler, Michael Muehlebach, and Joerg Stueckler. Black-box vs. gray-box: A case study on learning table tennis ball trajectory prediction with spin and impacts. In *Proceedings of the Learning for Dynamics and Control Conference (L4DC)*, 2023. 2

[3] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, et al. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 929–947, 2024. 7

[4] Tobias Baumgartner and Stefanie Klatt. Monocular 3d human pose estimation for sports broadcasts using partial sports field registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 5109–5118, 2023. 2

[5] C. G. BROYDEN. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970. 12

[6] Marcello Davide Caio, Gabriel Van Zandycke, and Christophe De Vleeschouwer. Context-aware 3d object localization from single calibrated images: A study of basketballs. In *Proceedings of the 6th International Workshop on Multimedia Content Analysis in Sports*, page 49–54, 2023. 2

[7] Jordan Calandre, Renaud Péteri, Laurent Mascarilla, and Benoit 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, 2021. 2

[8] Hua-Tsung Chen, Ming-Chun Tien, 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. *Soft Computing*, 20(3): 204–216, 2009. 2

[9] Zaineb Chiha, Renaud Peteri, and Laurent Mascarilla. Predicting 3d projectile motion in table tennis using computer vision and physics-informed neural network. In *International Conference on Content-Based Multimedia Indexing (CBMI)*, pages 1–7, 2024. 2

[10] David B. D'Ambrosio, Saminda Abeyruwan, Laura Graesser, Atil Iscen, Heni Ben Amor, Alex Bewley, Barney J. Reed, Krista Reymann, Leila Takayama, Yuval Tassa, Krzysztof Choromanski, Erwin Coumans, Deepali Jain, Navdeep Jaitly, Natasha Jaques, Satoshi Kataoka, Yuheng Kuang, Nevena Lazic, Reza Mahjourian, Sherry Moore, Kenneth Oslund, Anish Shankar, Vikas Sindhwani, Vincent Vanhoucke, Grace Vesom, Peng Xu, and Pannag R. Sanketi. Achieving human level competitive robot table tennis. *ArXiv*, abs/2408.03906, 2024. 2, 3, 6

[11] Moritz Einfalt and Rainer Lienhart. Decoupling video and human motion: Towards practical event detection in athlete recordings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020. 2

[12] Moritz Einfalt, Katja Ludwig, and Rainer Lienhart. Uplift and upsample: Efficient 3d human pose estimation with uplifting transformers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023. 2

[13] Morten Holck Ertner, Sofus Schou Konglevoll, Magnus Ibh, and Stella 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*, page 51–58, 2024. 2

[14] Tom Fawcett. Introduction to roc analysis. *Pattern Recognition Letters*, 27:861–874, 2006. 6

[15] Jerome Fink, Pierre Poitier, Maxime André, Loup Meurice, Benoît Frénay, Anthony Cleve, Bruno Dumas, and Laurence Meurant. Sign language-to-text dictionary with lightweight transformer models. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence, IJCAI 2023*, pages 5968–5976, 2023. 2

[16] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 12

[17] Thomas Gossard, Jonas Tebbe, Andreas Ziegler, and Andreas Zell. Spindoe: A ball spin estimation method for table tennis robot. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5744–5750, 2023. 2

[18] Thomas Gossard, Julian Krismer, Andreas Ziegler, Jonas Tebbe, and Andreas Zell. Table tennis ball spin estimation with an event camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3347–3356, 2024. 2

[19] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 Herd of Models. *arXiv e-prints*, art. arXiv:2407.21783, 2024. 5

[20] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need, 2023. 5

[21] Hawk-Eye Innovations. Hawk-eye. https://www.hawkeyeinnovations.com. Accessed: 2025-02-14. 1

[22] Chen Huang, 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 - MTA*, 60, 2012. 2

[23] Magnus Ibh, Stella Grasshof, Dan Witzner, and Pascal Madeleine. Tempose: a new skeleton-based transformer model designed for fine-grained motion recognition in badminton. In *2023 IEEE/CVF Conference on Computer Vision*

and Pattern Recognition Workshops (CVPRW), pages 5199–5208, 2023. 2

[24] Daniel Kienzle, Marco Kantonis, Robin Schön, and Rainer Lienhart. Segformer++: Efficient token-merging strategies for high-resolution semantic segmentation. *IEEE International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2024. 2

[25] Daniel Kienzle, Julian Lorenz, Katja Ludwig, and Rainer Lienhart. Towards learning monocular 3d object localization from 2d labels using the physical laws of motion. *Proceedings of the International Conference on 3D Vision 2024 (3DV)*, 2024. 2

[26] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014. 7

[27] Paul Liu and Jui-Hsien Wang. Monotrack: Shuttle trajectory reconstruction from monocular badminton video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3513–3522, 2022. 2

[28] Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, Yaofeng Sun, Chengqi Deng, Hanwei Xu, Zhenda Xie, and Chong Ruan. Deepseek-vl: Towards real-world vision-language understanding, 2024. 5

[29] Katja Ludwig, Moritz Einfalt, and Rainer Lienhart. Robust estimation of flight parameters for ski jumpers. In *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6, 2020. 1

[30] Katja Ludwig, Daniel Kienzle, and Rainer Lienhart. Recognition of Freely Selected Keypoints on Human Limbs . In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3530–3538, 2022. 2

[31] Andrii Maksai, Xinchao Wang, and Pascal Fua. What players do with the ball: A physically constrained interaction modeling. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 972–981, 2016. 2

[32] William McNally, Jacob Lambeth, and Dustin Brekke. Combining physics and deep learning models to simulate the flight of a golf ball. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 5119–5128, 2023. 2

[33] Takuya Nakabayashi, Kyota Higa, Masahiro Yamaguchi, Ryo Fujiwara, and Hideo Saito. Event-based ball spin estimation in sports. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3367–3375, 2024. 2

[34] Pascaline Parisot and Christophe Vleeschouwer. Consensus-based trajectory estimation for ball detection in calibrated cameras systems. *Journal of Real-Time Image Processing*, 16, 2019. 2

[35] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. 2

[36] Jinchang Ren, James Orwell, Graeme A. Jones, and Ming Xu. Real-time modeling of 3-d soccer ball trajectories from multiple fixed cameras. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(3):350–362, 2008. 2

[37] Takehiro Sawahata, Alessandro Moro, Sarthak Pathak, and Kazunori Umeda. Instance segmentation-based markerless tracking of fencing sword tips. In *2024 IEEE/SICE International Symposium on System Integration (SII)*, pages 472–477, 2024. 1

[38] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), 2024. 5

[39] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1195–1204, 2017. 7

[40] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. 3, 6

[41] Baptiste Toussaint and Maxime Raison. Real-time trajectory prediction of a ping-pong ball using a gru-tae. *Applied Intelligence*, 55(339), 2025. 2

[42] Gabriel Van Zandycke and Christophe De Vleeschouwer. 3d ball localization from a single calibrated image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3472–3480, 2022. 2

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 5

[44] Vieww GmbH. View 4d 2.0. https://vieww.com. Accessed: 2025-02-14. 1

[45] Qingyu Xiao, Zulfiqar Zaidi, and Matthew Gombolay. Multi-camera asynchronous ball localization and trajectory prediction with factor graphs and human poses. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13695–13702, 2024. 2

[46] Qingyu Xiao, Zixuan Wu, and Matthew Gombolay. Learning dynamics of a ball with differentiable factor graph and roto-translational invariant representations, 2025. 2

[47] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. ViTPose++: Vision Transformer for Generic Body Pose Estimation . *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 46(02):1212–1230, 2024. 2

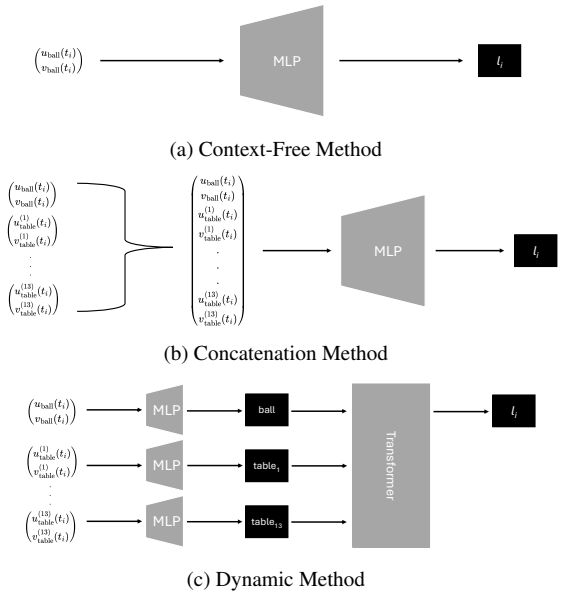
[48] Yifeng Zhang, Yongsheng Zhao, Rong Xiong, Yue Wang, Jianguo Wang, and Jian Chu. Spin observation and trajectory prediction of a ping-pong ball. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4108–4114, 2014. 2

[49] Ce Zheng, Sijie Zhu, Matias Mendieta, Taojiannan Yang, Chen Chen, and Zhengming Ding. 3d human pose estimation with spatial and temporal transformers. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021. 2

Towards Ball Spin and Trajectory Analysis in Table Tennis Broadcast Videos via Physically Grounded Synthetic-to-Real Transfer

Supplementary Material

A. Further Architecture Details



(a) Context-Free Method

(b) Concatenation Method

(c) Dynamic Method

Figure 7. Token embedding methods. Input to the embedding layers are the 2D coordinates of the ball and the 13 table keypoints. The output is the location token l_i. The embedding layer is applied for each time step t_i separately.

Figure 7 provides an overview of the different token embedding strategies utilized in our model. Each method processes the 2D coordinates of the ball along with 13 table keypoints to generate the location token l_i at time t_i. The embedding operation is applied separately for each time step t_i.

- **Context-Free Method:** The context-free method directly embeds the ball coordinates via a multilayer perceptron (MLP) without using any table keypoints.
- **Concatenation Method:** The 2D coordinates of all 14 points are concatenated into a single vector. This vector is then transformed into a location token via an MLP.
- **Dynamic Method:** Instead of direct concatenation, a small transformer encoder processes the table keypoints and condenses their information into the ball position token. This token is then used as location token l_i, the other tokens are discarded.

To evaluate model performance across different architectural complexities, we vary the number of transformer layers L, the number of attention heads H, and the embedding dimension d. Table 4 summarizes the configurations explored.

Size	Layers L	Heads H	Embedding Dimension d	Number of Parameters
Small	8	4	32	0.06×10^6
Base	12	4	64	0.3×10^6
Large	16	4	128	1.6×10^6
Huge	16	8	192	3.2×10^6

Table 4. Transformer architecture variants. The number of trainable parameters is calculated for the model with connect-stage SPT architecture and concatenation token embedding module.

B. Annotation Details

For each trajectory, we annotate the 13 table keypoints in the first frame. Since the camera remains static throughout each video, these annotations are consistently used for all subsequent frames within the trajectory. The annotated keypoints are illustrated in Figure 8. Annotating the spin direction is particularly challenging, as the spin is not directly observable in broadcast footage. To infer the spin type, we analyze the paddle's orientation at the moment of impact.

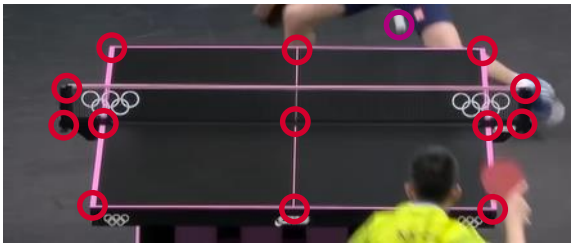


Figure 8. The 13 table keypoints (red circles) and the ball position (purple circle) are highlighted.

- **Topspin**: If the paddle is angled towards the table, the shot is labeled as topspin. This is illustrated in Figure 9a.
- **Backspin**: If the paddle is angled away from the table, the shot is labeled as backspin. This is illustrated in Figure 9b.

This annotation approach provides a practical method for inferring spin direction despite the limitations of broadcast video data.

C. Regressing camera matrices

Although our method operates in an end-to-end manner without requiring camera calibration as input, the intrinsic and extrinsic camera matrices are necessary for computing

(a) Topspin

(b) Backspin

Figure 9. Example frames for the annotation of the spin direction. If the paddle is facing the table (a), the shot is annotated as topspin. If the paddle is facing away from the table (b), the shot is annotated as backspin.

the 2D reprojection error. For each trajectory, we manually annotate the 13 table keypoints once. Since the corresponding 3D world coordinates are known due to standardized table sizes in professional matches, the camera matrices can be estimated by minimizing the reprojection error of these 13 points. However, this regression process is inherently unstable, and even small annotation errors can lead to significant inaccuracies in the estimated camera matrices. To mitigate this issue, we employ the RANSAC algorithm [16] to robustly filter out erroneous annotations. In each RANSAC iteration, we randomly select six non-planar keypoints and compute an initial estimate of the camera matrices using the Direct Linear Transformation (DLT) algorithm [1]. This initial estimate is then refined using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimization algorithm [5]. With the refined matrices, we determine the number of inliers by checking the reprojection error. A point is classified as an inlier if the reprojected 3D point is within 3 pixels of the corresponding 2D annotation. This procedure is repeated 100 times, and the setting with the highest number of inliers is selected. Using all identified inliers, we compute the final camera matrices by first applying the DLT algorithm and subsequently refining the result with the BFGS optimization. The RANSAC-based approach is essential for mitigating the impact of slight errors in the 2D annotations and obtaining accurate camera matrices. We highlight that our model, which does not require camera calibration, is very robust to minor errors in the 2D input, highlighting the benefits of implementing an end-to-end approach.

D. Augmentation Details

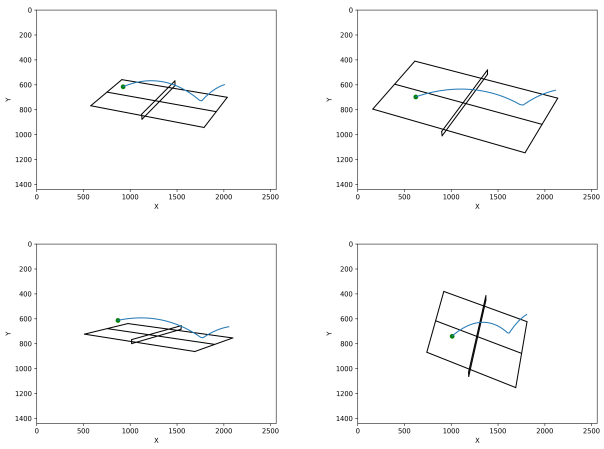


Figure 10. A trajectory sampled from 4 different camera perspectives. During training, we transform the trajectory with such randomly sampled camera parameters to introduce different viewpoints.

During training, we **randomly sample plausible camera parameters** to simulate diverse camera perspectives, increasing the diversity of the model inputs and ensuring that the model generalizes well across different viewpoints. Examples of trajectories visualized from various sampled camera parameters are shown in Figure 10.

In the synthetic dataset, we do not only store the ball's 3D position at each time step but also record intermediate positions. For these intermediate positions, we employ a "virtual" framerate of 500 Hz, which is ten times higher than the actual framerate. This allows us to accurately simulate **motion blur** in a physically consistent manner. Rather than using the exact ball position at a given timestamp, we randomly select a point within a temporal window around each frame. For the experiments in this paper, we define this window such that the selected point has a timestamp within a maximum deviation of $0.4 * \frac{1}{50\,\text{Hz}}$. Motion blur is applied not only to the 2D ball positions but the 3D ground truth positions are shifted accordingly.

The **sudden end** augmentation is designed to simulate scenarios in which the opposing player interferes, leading to an abrupt termination of the trajectory. For each trajectory, we remove a randomly selected number of coordinates at the end, however, we ensure that the trajectory always includes the bounce on the table. This augmentation is applied with a probability of 50% during training, allowing the model to learn from complete trajectories while also adapting to

cases where the ball's motion is unexpectedly interrupted.

The **Gaussian blur** augmentation is implemented by introducing random noise to the 2D ball position and the 2D table keypoints. The noise is sampled from a Gaussian distribution with a standard deviation of 2 pixels in both the x- and y-directions, effectively simulating annotation inaccuracies.

E. Further Experiments

This section presents additional experiments that provide deeper insights into the model's behavior. All experiments are based on the best model, which is described in Section 6.4. It uses the concatenation token embedding method, the connect-stage SPT architecture, and all data augmentations.

E.1. Spin Prediction Coordinate System

Method	Synthetic		Real			
	$\Delta\vec{\omega}$	$\Delta\vec{r}_{\text{world}}$	*acc*	F_1	ROC-AUC	$\Delta\vec{r}_{\text{img}}$
world	48.7 Hz	5.5 cm	92.0 %	0.917	0.990	**0.19 %**
ball	48.3 Hz	5.4 cm	**94.0 %**	**0.938**	**1.000**	0.22 %

Table 5. Comparison of different spin prediction coordinate systems. The best results on the real data are highlighted in bold.

In Section 3.2, we discussed that while the trajectory is analyzed in the world coordinate system, the spin is evaluated in the ball coordinate system. According to Equation 2, the predicted spin can be transformed between coordinate systems. Thus, there are two approaches for predicting the spin:

- The network is trained to predict the spin in the world coordinate system, and the predicted trajectory is used in Equation 2 to transform the spin into the ball coordinate system.
- The network is trained to directly predict the spin in the ball coordinate system, eliminating the need for any transformation.

Since the first approach relies on the predicted trajectory for coordinate transformation, it may introduce additional errors. Conversely, using the same coordinate system for both trajectory and spin could simplify training, as the network does not need to learn the transformation.

Table 5 compares both approaches. Training the network directly in the ball coordinate system results in slightly better performance across all spin-related metrics. However, trajectory prediction benefits from predicting the spin in the world coordinate system. Overall, the differences are minor. Choosing the coordinate system depends on whether trajectory accuracy or spin accuracy is more critical for the specific application. This allows for flexibility in the model design, enabling it to be tailored to the specific requirements of the task at hand.

E.2. Positional Encoding

Method	Synthetic		Real			
	$\Delta\vec{\omega}\downarrow$	$\Delta\vec{r}_{\text{world}}\downarrow$	*acc* ↑	F_1 ↑	ROC-AUC ↑	$\Delta\vec{r}_{\text{img}}\downarrow$
rotary	48.7 Hz	5.5 cm	**92.0 %**	**0.917**	**0.990**	**0.19 %**
added	52.6 Hz	5.8 cm	90.0 %	0.897	0.987	0.26 %

Table 6. Comparison of different positional encodings. The best results on the real data are highlighted in bold.

The standard approach for incorporating positional information in transformers is by adding a fixed sinusoidal positional encoding to the token embeddings. However, our model utilizes a rotary positional encoding, which is commonly used in language models. Table 6 compares both methods. As the rotary positional encoding achieves better performance across all metrics, we conclude that it is more suitable for our task.

E.3. Loss Target

Method		Synthetic		Real			
$\mathcal{L}_{\text{trajectory}}$	$\mathcal{L}_{\text{spin}}$	$\Delta\vec{\omega}\downarrow$	$\Delta\vec{r}_{\text{world}}\downarrow$	*acc* ↑	F_1 ↑	ROC-AUC ↑	$\Delta\vec{r}_{\text{img}}\downarrow$
✓	✓	48.7 Hz	5.5 cm	**92.0 %**	**0.917**	**0.990**	**0.19 %**
×	✓	60.6 Hz	-	78.0 %	0.769	0.890	-
✓	×	-	5.5 cm	-	-	-	0.22 %

Table 7. Comparison of joint prediction with individual models for each task. ✓ indicates which loss function is used during training, while × indicates the absence of the specific loss. The best results on the real data are highlighted in bold.

Our model is designed to jointly predict both trajectory and spin. For training both task, we simply sum the two loss functions in Equation 7. In this section, we examine whether joint prediction is beneficial or if the two tasks should be handled by separate models.

Table 7 compares joint prediction with separate models. The results clearly show that joint prediction outperforms the separate models across all metrics. This suggests that the network extracts useful information from one task that enhances the other. Thus, joint prediction is an effective approach.

E.4. Model Size

Method	Synthetic		Real			
	$\Delta\vec{\omega}\downarrow$	$\Delta\vec{r}_{\text{world}}\downarrow$	*acc* ↑	F_1 ↑	ROC-AUC ↑	$\Delta\vec{r}_{\text{img}}\downarrow$
small	64.9 Hz	10.7 cm	**92.0 %**	**0.917**	0.956	0.29 %
base	51.0 Hz	6.2 cm	90.0 %	0.895	**0.998**	0.25 %
large	48.7 Hz	5.5 cm	**92.0 %**	**0.917**	0.990	0.19 %
huge	48.8 Hz	5.1 cm	86.0 %	0.850	0.971	**0.17 %**

Table 8. Comparison of different model sizes. The best results on the real data are highlighted in bold.

Table 8 compares different model sizes, which are defined in Table 4. All models demonstrate good performance in spin prediction. However, increasing the model size improves trajectory prediction accuracy. Additionally, the spin prediction performance of the largest model is slightly worse than that of the other models, possibly due to overfitting. Therefore, we identify the large model as the best compromise between spin prediction and trajectory prediction.

F. Reproducibility and Open Resources

To facilitate reproducibility and further research, we provide the following resources:

- Synthetic trajectories used for training.
- Annotations for the real-world dataset.
- Trained model weights.
- Source code for both training and inference.

All resources are publicly available at
https://kiedani.github.io/CVPRW2025/.