SynthNet: Leveraging Synthetic Data for 3D Trajectory Estimation from Monocular Video

Morten Holck Ertner
IT-University of Copenhagen
Copenhagen, Denmark
mert@itu.dk

Sofus Schou Konglevoll
IT-University of Copenhagen
Copenhagen, Denmark
sosk@itu.dk

Magnus Ibh
IT-University of Copenhagen
Copenhagen, Denmark
ibhq@itu.dk

Stella Graßhof
IT-University of Copenhagen
Copenhagen, Denmark
stgr@itu.dk

ABSTRACT

Reconstructing 3D trajectories from video is often cumbersome and expensive, relying on complex or multi-camera setups. This paper proposes SynthNet, an end-to-end pipeline for monocular reconstruction of 3D tennis ball trajectories. The pipeline consists of two parts: Hit and bounce detection and 3D trajectory reconstruction. The hit and bounce detection is performed by a GRU-based model, which segments the videos into individual shots. Next, a fully connected neural network reconstructs the 3D trajectory through a novel physics-based training approach relying on purely synthetic training data. Instability in the training loop caused by relying on Euler-time integration and camera projections is circumvented by our synthetic approach, which directly calculates loss from estimated initial conditions, improving stability and performance.
In experiments, SynthNet is compared to an existing reconstruction baseline on a number of conventional and customized metrics defined to validate our synthetic approach. SynthNet outperforms the baseline based on our own proposed metrics and in a qualitative inspection of the reconstructed 3D trajectories.

CCS CONCEPTS

• **Computing methodologies** → *Activity recognition and understanding*; *Tracking*; **Reconstruction**; *Neural networks*; • **Applied computing** → *Physics*.

KEYWORDS

3D Reconstruction, Machine Learning in Sports, Computer Vision, Ball Tracking, Neural Network, Synthetic Data, Differential Equations

ACM Reference Format:
Morten Holck Ertner, Sofus Schou Konglevoll, Magnus Ibh, and Stella Graßhof. 2024. 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 (MMSports '24), October 28-November 1, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3689061.3689073

MMSports '24, October 28-November 1, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-1198-5/24/10
https://doi.org/10.1145/3689061.3689073

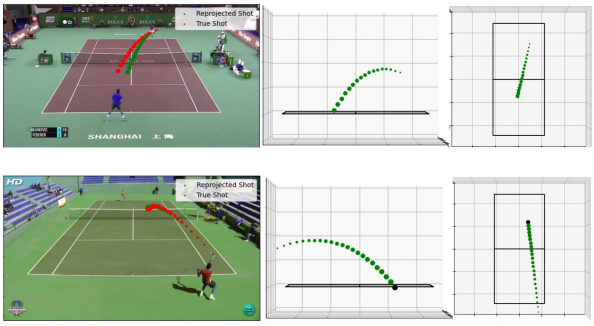

Figure 1: 3D trajectories constructed from monocular video using our end-to-end system SynthNet.

1 INTRODUCTION

In high-level tennis, accurate 3D information on the ball holds great value, both in assisting umpires in calling the game and gathering player and shot information for analytical purposes. Currently, the most popular technology is the Hawk-Eye system[1], which uses multiple cameras and triangulation to capture the ball's 3D position with an error margin of 3.6 millimeters. Such a system is expensive and thus only available for the highest level of tennis players. However, most tennis players are amateurs and often lack access to such a system. Enabling the tracking of the ball's 3D position using a single, everyday camera, such as a phone, would allow the average tennis player to access game statistics previously reserved for professionals.

Estimating 3D ball positions from monocular video has been achieved in other sports like volleyball [2], basketball [1], and badminton [12], but, to our knowledge, has never been achieved in tennis. Thus, the motivation for this paper is to estimate 3D trajectories from monocular video in tennis.
Direct 2D-to-3D lifting of ball coordinates presents a significant challenge since $(2D, 3D)$—pairs of ball coordinates are publicly unavailable. Instead, we segment the game into individual shots and

[1] https://www.hawkeyeinnovations.com

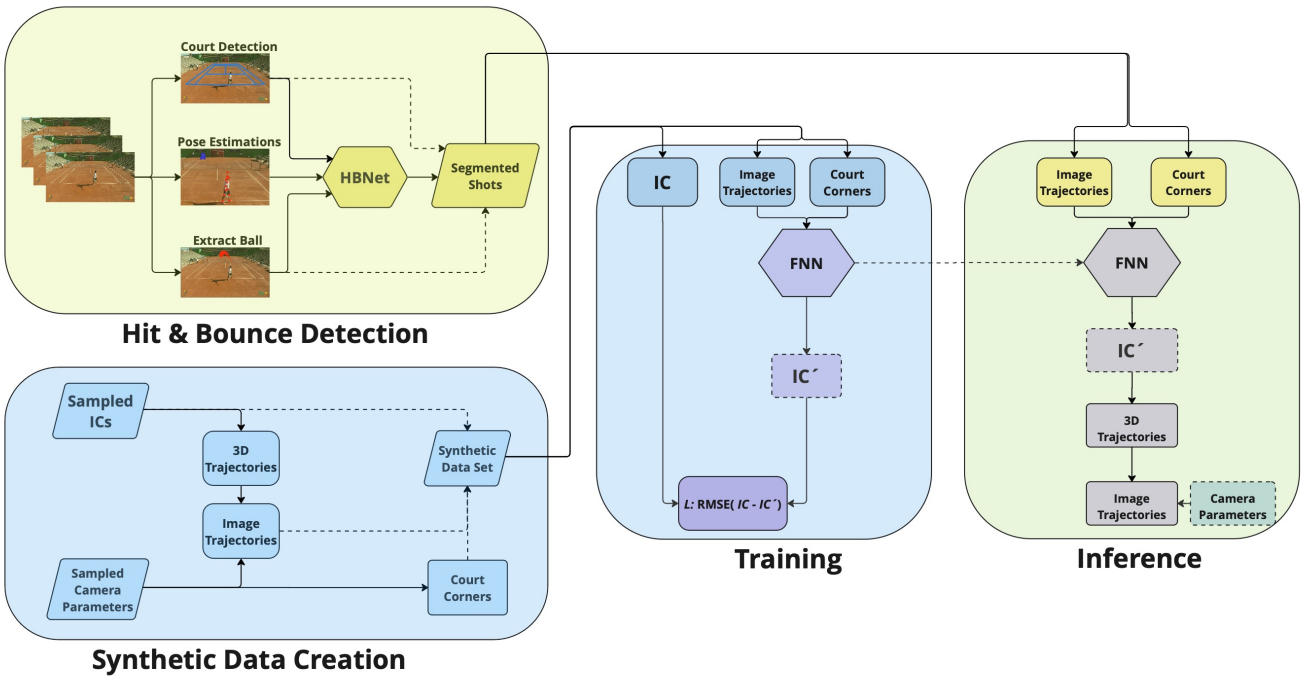

Figure 2: Complete End-to-End Pipeline. *For training*, synthetic data is generated by sampling initial conditions (IC). This data is used to create 3D trajectories and image trajectories along with sampled camera parameters. The feed-forward neural network (FNN) predicts initial conditions based on image trajectories and court corners, and trains using Root Mean Squared Error (RMSE) between true and predicted IC as the loss function. *For inference*, features are extracted from video data and fed into the hit and bounce detection model HBNet. Predictions from HBNet segment the videos into individual shots, which are then used alongside court corners as input to the trained FNN. The FNN predicts a new set of initial conditions, enabling the creation of 3D trajectories that are reprojected onto image coordinates to calculate reprojection error.

use the physical laws of motion to set up differential equations for the trajectories, after which Euler's method is used to estimate the 3D position of the ball, which can be projected to the 2D image coordinates using the camera parameters. Usually, a training loop involving iterative time integration and subsequent 3D-2D camera projection is too unstable to converge. Instead, we propose a feed-forward neural network that predicts the **initial conditions** (**IC**) given the image (2D) trajectories as input, which provides a significantly smoother training routine. This is realized by, prior to training the model, random sampling of initial conditions and subsequent trajectory simulation using Euler's method. Then, only keeping trajectories, passing the net, and landing within the court allows for acquisitions of 3D shot trajectories. Lastly, the image coordinates are retrieved by projection to the image plane using randomly chosen camera parameters found from known positions of the tennis court. A model could also be trained on the synthetic (2D,3D) pairs. However, compressing input image trajectories allows for better generalization of real-world trajectories.

To segment the game into shots, we identify the start and end of each shot by detecting hits and bounces. We use the TrackNet Tennis dataset [7], consisting of videos from 10 professional tennis matches with annotated hits and bounces. From the videos, we extract three features: player poses, ball coordinates, and court corner coordinates, to use as input for our hit and bounce detection model, which we call *HBNet*. We evaluate HBNet on the ground truth labels from the TrackNet dataset and use the predictions to segment the videos into individual shots.

In the trajectory experiments, we evaluate the model on the reprojection error between the true and predicted (estimated with [19]) image trajectories. Additionally, realizing the limitation of using reprojection error as the sole metric, we propose tailored metrics to validate the real-world generalization of our synthetically trained model. The complete *SynthNet* pipeline can be seen in Figure 2.

2 RELATED WORKS

A lot of prior works in the field of computer vision and machine learning in sports have focused on individual components such as court detection [3, 21, 22], ball tracking [7, 11, 20] and pose estimation [8–10]. Previous research used those features to perform action recognition in sports. Huang et al. [5, 6] use the ball trajectories with audio information to detect hits made in a tennis game. Similarly, Shublewska-Paszkowska et al. [18] use 3D tennis movement as input to a graph neural network to predict two actions: forehand

and backhand strokes, while Cai and Tang [24] identifies 12 types of tennis shots using a Long-Short Term Memory model.

When doing 3D tracking of sports balls, the most used and most reliable way is to use a multi-camera setup [4, 15, 23, 25], where the scene is captured from multiple angles and triangulation is used to estimate the balls 3D location. While this can give accurate results, it is less cost-efficient than using a single camera as you need access to multiple cameras and the possibility and permission to set them up. Some research in 3D ball tracking from monocular video has been done in several sports such as volleyball [2], table tennis [17], basketball [1] and badminton [12], all using a similar approach. They extract features from video and use optimization techniques to find the best set of initial conditions for a reconstructed 3D trajectory. This method has several flaws. First of all, using optimization techniques means that each shot is optimized one at a time, resulting in long inference times. Secondly, the reprojection error between the reconstructed 3D trajectory and the image trajectory is used as part of the loss function, which is undesirable as this can lead to unrealistic trajectories. Instead, we propose to use a neural network to predict the initial condition, train this network on synthetic data, and use their ground truth initial condition for loss during training.

3 RECONSTRUCTING 3D TRAJECTORIES

3.1 Problem statement

In this paper, we develop an end-to-end pipeline to reconstruct 3D tennis ball trajectories from monocular video. The process involves extracting the ball, court, and player poses to predict hits and bounces during a match. These predictions segment the video into shots, defined from a hit to a subsequent bounce. We deploy a neural network trained exclusively on synthetic data to estimate initial conditions for ballistic 3D trajectories using 2D image trajectories from these segmented shots. Subsequently, we evaluate the model using reprojection error and additional proposed metrics. Training will be done solely using a synthetic dataset where ground truth initial conditions are known, and evaluation will be conducted on segmented shots using the defined metrics. To generate trajectories from initial conditions, we employ Euler's method for solving differential equations.

3.2 Hit Bounce Detection

To detect the hits and bounces in a match, we use a similar architecture to Liu et. Al [12]. We extract court coordinates with an algorithm proposed by Kosolapov Sergey [16], detect poses with RTMO [13], and track the ball with WASB [19]. The poses are filtered by searching for a pose inside each player's court. If there is no pose in the court, it finds the pose closest to the back line. Court, player poses, and the ball is then used as features for a model that predicts hits, bounces, and nonhits. To get the temporal aspect of the movement before and after a shot, we create a Gated Recurrent Unit (GRU) based architecture, called Hit-Bounce Net (HBNet). HBNet consists of an embedding layer with 32 neurons, 6 gated recurrent layers with 32 units, and 0.2 dropout. Followed by a fully connected layer and a softmax which generates confidence scores. The model takes a snippet of 21 frames at a time and predicts

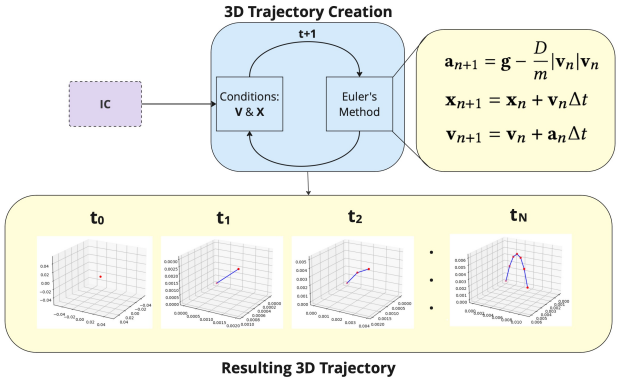

Figure 3: Visual presentation of creating 3D trajectory given a set of initial conditions. The loop runs in N iterations, and with each time step a position and velocity is created.

whether a hit or bounce occurs in the last 9 frames of a snippet. HBNet achieves an accuracy of 86% and a macro F1-score of 0.84. The shots segmented with the predictions from HBNet are called HBNet + WASB.

3.3 Synthetic Learning Procedure

Using the segmented shots from HBNet we can reconstruct 3D trajectories for each shot. We define a shot to be a hit followed by a bounce or another hit. Estimating each shot individually means we can model the trajectory as a projectile under drag:

$$\frac{d^2\mathbf{x}(t)}{dt^2} = \mathbf{g} - \frac{D}{m}|\mathbf{v}(t)|\mathbf{v}(t), \tag{1}$$

where m is the mass of the ball, $\mathbf{x}$ and $\mathbf{v}$ are the 3D position and velocity vectors, g is the gravitational constant, and D the drag coefficient. As this equation has no analytical solution, we use forward Euler's time integration to retrieve a discretized version of the shuttle position with N time steps. At each step, the acceleration is assumed constant for a small enough time step $\Delta t = t_{n+1} - t_n$, where $n \in N$ is the current step. Thus, for each small Δt, we calculate acceleration $\mathbf{a_{n+1}}$, velocity $\mathbf{v_{n+1}}$, and position $\mathbf{x_{n+1}}$, based on the acceleration, velocity, and position of the current time n step as follows:

$$\mathbf{a}_{n+1} = \mathbf{g} - \frac{D}{m}|\mathbf{v}_n|\mathbf{v}_n \tag{2}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{a}_n \Delta t \tag{3}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_n \Delta t. \tag{4}$$

To create a 3D trajectory initial conditions $\mathbf{v}_0 = (\mathbf{v}_{x0}, \mathbf{v}_{y0}, \mathbf{v}_{x0})^T$ and $\mathbf{x}_0 = (x_0, y_0, z_0)^T$ is required. A visualisation of this process can be seen in Figure 3. We center the world coordinates in the middle of the tennis court. The y-axis is oriented along the length of the court, away from the camera. The x-axis is oriented across the width of the court, and the z-axis is oriented vertically, perpendicular to the ground.

To predict a set of initial conditions we use a feedforward neural

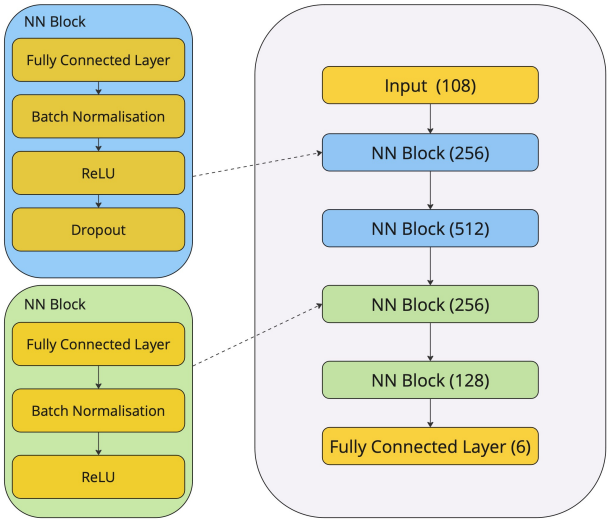

Figure 4: Model Architecture. The module has an input size of 108, 4 NN blocks, and an output size of 6.

network (FNN) consisting of four hidden layers. The model architecture can be seen in Figure 4. The goal of the model is to predict a set of the six initial conditions that create the best-fitting 3D trajectory. We only train the model on synthetic data, where the ground truth initial conditions are known, and we calculate the loss as:

$$L = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(IC_i - \tilde{IC}_i)^2}, \tag{5}$$

where IC is the ground truth initial conditions and $\tilde{IC}$ the predicted initial conditions.

For inference, the FNN uses the extracted 2D ball coordinates and court corners (extracted from the videos) to make predictions of initial conditions. We estimate a 3D trajectory based on the initial conditions and reproject it to image coordinates using a perspective transformation. We find the camera parameters for each video with camera calibrations assuming known world coordinates and corresponding image coordinates. The corresponding coordinates are comprised of the court corners and manually annotated net-pole coordinates. Based on this, we calculate the reprojection error (RE) between the real image trajectory (IT) and predicted image trajectory ($\tilde{IT}$) using root mean squared error (RMSE):

$$RE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(IT_i - \tilde{IT}_i)^2}, \tag{6}$$

3.4 Proposed Evaluation Metrics

We introduce three alternative evaluation metrics based on the 3D landing position of the ball: *Landing Error* (LE), *Tile Accuracy* (T.acc) and *Tile F1-score* (T.F1). To find the 3D landing position of the image trajectories, we use a homography to find the position of the ball when a bounce occurs. The Landing error is the distance

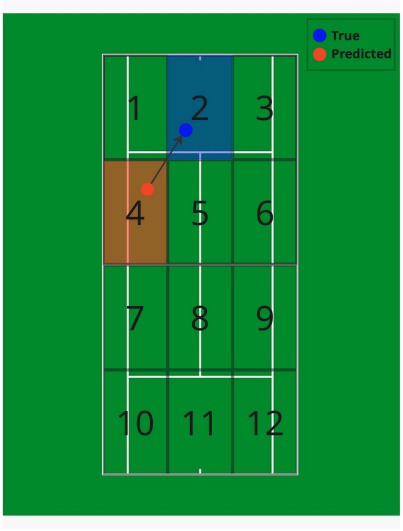

Figure 5: Explanation of Landing error and tile accuracy/F1-score. The predicted landing position, given label 2 is depicted as red dot. The blue dot is the true landing position, given label 4, found using homography on the image trajectory. The landing error is the distance between the two landing positions and tile accuracy/F1-score is calculated based on their given label.

between the real and predicted landing position

$$LE = \sqrt{(x_l - \tilde{x_l})^2 + (y_l - \tilde{y_l})^2}, \tag{7}$$

where (x_l, y_l) is the true landing position and $\tilde{x_l}, \tilde{y_l}$ the predicted position.

To define the tile accuracy and tile F1-score we divide the court into 12 tiles, six on each side of the net, and assign the position a label corresponding to the tile they land in as seen in Figure 5. Using these labels we can calculate accuracy and F1-score.

Lastly, on synthetically created data where we have the true IC and therefore the true 3D trajectory, we can use the *reconstruction error* (RecE) as an evaluation metric. The reconstruction error is the mean Euclidean distance between the predicted and the true 3D trajectory

$$RecE = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i - \tilde{x_i})^2 + (y_i - \tilde{y_i})^2 + (z_i - \tilde{z_i})^2}. \tag{8}$$

4 EXPERIMENTS

4.1 Dataset and Implementation Details

We use the TrackNet tennis dataset [7], which contains 96 rallies spread across 10 different games of both men and women. All videos are statically filmed from an overhead *broadcast* view behind one of the backlines, with a resolution and frame rate of 1280x720 and 30 fps. The videos have annotated hits, bounces, and ball image

coordinates for each frame, along with a visibility score for the ball. Additionally, we annotated some hits and bounces that were missing in the original annotations.

We create additional synthetic data to support the *real* data, by sampling random combinations of initial conditions, and use these conditions to create 3D trajectories. The sampling is done within boundaries such that the trajectory starts within the sidelines and not too far behind the backlines, and such that the velocity is below the highest recorded velocities in tennis.

To use the trajectories as input for our model, we reproject them down onto image coordinates using camera parameters sampled from the *real* data. The court corners corresponding to the camera parameters are used as input. We create 10000 shots, 5000 from each side of the court, all of varying lengths and ending when they hit the ground. The synthetic data allows us to measure both reconstruction and reprojection errors. The real and synthetic data are both divided into train and test, with two games reserved for testing and eight for training. With the synthetic data, we divide based on which game the sampled camera parameters are taken from. The image trajectories are padded, using -1, to have a length of 50 frames to create inputs of equal length, and trajectory and court coordinates are scaled down by the image dimensions. The model is implemented in Pytorch using a dropout of 0.2 and solely trained on the synthetic data for 25 epochs with early stopping.

4.2 Model Evaluation

To evaluate our model, we define three different sets of trajectories:

(1) the ground truth annotations and image ball trajectories from the TrackNet dataset,
(2) the predictions and image ball trajectories from our pipeline, where we utilize WASB to find the ball and our HBNet model to detect the shots (HBNet+WASB), and
(3) the synthetically created trajectories.

Table 1 shows the results of SynthNet on the ground truth, HBNet+WASB, and synthetic test data. The ground truth and HBNet+WASB are only trajectories from the test set, games 1 and 8, and the synthetic trajectories only have trajectories with camera parameters from these two games as well. The results show that SynthNet consistently performs best over all metrics on the synthetic trajectories. This is expected, as these trajectories can be perfectly estimated using our 3D projection method. Using the synthetic data yields a reprojection error of 16.1 pixels, which is more than twice as for the HBNet + WASB and ground truth trajectories. Likewise, we see that the model has half the reconstruction error on the synthetic data compared to both HBNet+WASB and ground truth trajectories.

SynthNet performs equally well on the ground truth trajectories and the HBNet+WASB trajectories in terms of reprojection error, with 41 pixels versus 41.31 pixels, while the landing error differs significantly by a meter. Interestingly, the model has a much higher accuracy, 8%, and F1-score, 0.9, on HBNet+WASB trajectories than on the ground truth trajectories. Additionally, we see a relatively large difference between the mean and median of RE and LE on ground truth and HBNet+WASB trajectories indicating that there are outliers with high RE and LE. The differences between mean and median values on the synthetic data are smaller, showing fewer outliers. Lastly, we can evaluate our model using recreation error on the synthetic data, where we get a mean error of 1.39 meters.

4.3 Effect of Knowing Camera Parameters

To examine the effect of the model having encountered the camera parameters doing training, we test the model on HBNet+WASB trajectories from all the games, i.e. both the test and train sets. Table 2 shows the results. The model is trained using camera parameters from all other games than game 1 and 8, but we don't see that it achieves the worst results on these. On game 1 it is among the best results in all the metrics, while on game 8 it is among the worst results.

Game 4 yields the worst performance based on our proposed metrics. It is noteworthy that game 4 was recorded from a much lower angle than the rest of the games. Based on those observations we conclude that it does not matter as much if the model has encountered the exact camera parameters or angle before if they are not significantly different from previously encountered ones.

4.4 Evaluating 3D Trajectories

Visually inspection of our 3D trajectories (HBNet+WASB) reveals a couple of general tendencies. Firstly, the model seems to have difficulties determining the proper 3D start position of the shot, when the trajectory starts from the side of the court furthest away from the camera. Table 3 shows the absolute error, in meters of the initial position and meters pr. second of the initial velocity, on the synthetic trajectories.

The model is good at determining the correct x-coordinate but has difficulties with especially the y-coordinate of the start position, where it is, on average, 1.6 meters away from the true y-coordinate. We observe the same tendencies on the initial velocity, where it mostly struggles with velocity in the y-direction (along the field). The model is generally better at estimating the initial position than the initial velocity.

Figure 6 shows the image coordinates of the ball found using WASB (red) and the predicted 3D trajectory and its reprojection (green). It shows a reprojection that is somewhat close to the real image trajectory, starting around the same pixels, and landing slightly away from the true position. However, the real 3D shot starts from around the backline, while the predicted shot starts closer to the net. This problem arises, as the perspective makes it more difficult to distinguish and determine distances when it is further away from the camera. Furthermore, when finding the camera parameters, the only reference points that are not on the ground plane are the poles in the middle of the court. This lack of elevated 3D reference points from the rest of the court could make it more difficult to project elevated 3D points onto the image accurately.

Secondly, SynthNet has difficulties with outlier shots such as high-velocity shots, shots from weird angles, shots that have a lot of spin, shots that last only a couple of frames, etc. Figure 7 shows a serve from game 5, with a high velocity, that flies at a strange angle. If we only see the trajectory, it could look like it is flying across instead of along the court, starting close to the net and ending near it as well. As it is a serve, the ball starts high up into the air, which again can be difficult to determine from image coordinates due to the perspective. As a result, the predicted trajectory starts too close

Table 1: Model performance on the three different test sets of trajectories. $\overline{RE}$ denotes the mean RE, $\widetilde{RE}$ the median RE, $\overline{LE}$ the mean LE, and $\widetilde{LE}$ the median LE.

	Image			World			
Image Trajectories	$\overline{RE}$	$\widetilde{RE}$	T.acc	T.F1	$\overline{LE}$	$\widetilde{LE}$	RecE
Ground Truth	41.01 px	29.12 px	46.08%	0.307	2.73 m	2.17 m	-
HBNet + WASB	41.31 px	31.82 px	53.85%	0.398	3.78 m	2.4 m	-
Synthetic	**16.1 px**	**14.65** px	**75.1%**	**0.751**	**1.35 m**	**1.11 m**	1.39 m

Table 2: Results of SynthNet on HBNet+Trajectories trajectories for each game.

	Image	World		
	RE	T.acc	T.F1	LE
game1	35.29 px	64.15%	0.488	3.19 m
game2	35.27 px	56.45%	0.402	**2.25 m**
game3	54.78 px	54.54%	0.294	3.59 m
game4	36.72 px	28.07%	0.187	7.07 m
game5	38.26 px	45.45%	0.202	3.30 m
game6	43.43 px	**66.67%**	0.388	3.40 m
game7	47.76 px	54.00%	0.3540	2.91 m
game8	47.56 px	43.14%	0.320	4.40 m
game9	**34.17 px**	59.52%	0.396	2.61 m
game10	39.56 px	61.54%	**0.5523**	3.02 m
average	41.28 px	53.35%	0.358	3.58 m

Table 3: Average absolute differences and standard deviations on initial conditions on synthetic test data.

IC	Direction	Absolute Error
Position (x_0)	x	0.25 ± 0.23 m
	y	1.63 ± 1.25 m
	z	0.40 ± 0.29 m
Velocity (v_0)	x	0.64 ± 0.53 m/s
	y	3.11 ± 2.67 m/s
	z	0.58 ± 0.41 m/s

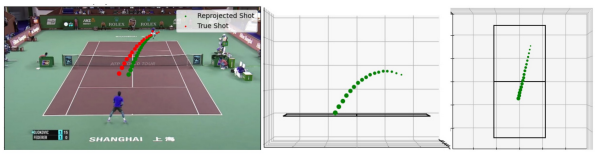

Figure 6: WASB image trajectory and reprojected predicted 3D shot. Left: WASB image trajectory (Red) and reprojected predicted 3D shot(green). Middle: Predicted 3D shot seen from the side. Right: Predicted 3D shot seen from above.

to the net, does not have the correct trajectory, and does not land

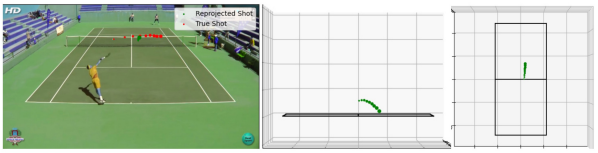

Figure 7: WASB image trajectory and reprojected predicted 3D shot. Left: WASB image trajectory (Red) and reprojected predicted 3D shot(Green). Middle: Predicted 3D shot seen from the side. Right: Predicted 3D shot seen from above.

close to the actual position. In such cases, the model appears to find the least worst trajectory, which is a short trajectory in the middle of the court. This is a general tendency with outlier shots.

Thirdly, SynthNet also tends to create initial conditions such that the trajectories end before they hit the ground. This is of course a problem, as we calculate our proposed metrics under the assumption that the trajectory ends on the ground.

Furthermore, we observe a tendency to create trajectories that are directed more toward the vertical middle line of the court than the actual shot is. This observation is supported by the F1-scores of each of the 12 defined tiles. Figure 8 show SynthNet achieves the best results on the tiles along the vertical middle line. Additionally, we see that there is a slightly better score at the side of the court furthest away from the camera, which might be due to the camera angle, where the ball's position on the court closer to the camera is more difficult to estimate. Interestingly, on tile 7, the model achieves an F1-score of 0.

4.5 Ablation Study

We conducted an ablation study to explore the effect of using the court corners as input along with the image trajectory. The results presented in Table 4 show a clear performance increase in all metrics when using the court corners. We observe that knowing the court size (in image coordinates) yields more accurate trajectories. The court corners are static in the world coordinates. Thus, the changing image position of the corners provides the model with information about camera position and perspective. Additionally, we hypothesize this is because, with the court corners, the model knows the court's limits and, therefore, finds it easier to make shots within these boundaries. We note that the video material is from professional tennis players. These players rarely shoot the ball far outside the court, and the synthetically created trajectories always

Table 4: Ablation study of input features on all HBNet+WASB trajectories.

Input	Image RE	T.acc	World T.F1	LE
Image Trajectory	48.25 px	41.35%	0.289	4.13 m
Image Trajectory + court corners	**41.31 px**	**53.85%**	**0.398**	**3.78 m**

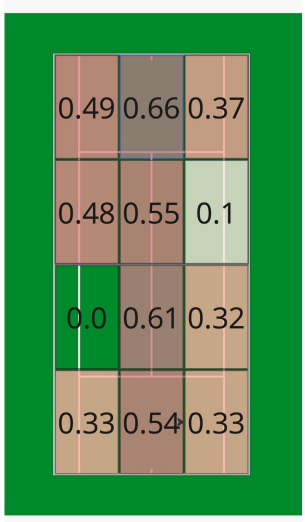

Figure 8: Tile F1-scores of each of the 12 defined tiles on HBNet+WASB trajectories.

land within the court. Therefore, it might be valuable for the model to have additional information on where the trajectory should be within the court.

4.6 SynthNet vs. Baseline

We compare our method of estimating initial conditions with a simplified version of the previously used approach. The problem is formulated as an optimization task minimizing only the reprojection error, which we solve by Powell's method [14]. Small experiments using additional loss terms like start/landing loss were made but yielded no definitive results, thus we chose to use this simplified version. Table 5 shows the results for the optimizer and SynthNet on HBNet+WASB trajectories. On one hand, we see the optimizer generally yields a much lower reprojection error. This is expected based on the design which employs the reprojection error as its loss function and SynthNet does not. On the other hand, SynthNet generally results in better tile accuracy, tile F1-score, and landing error, indicating that it performs better than the previous optimizer. Manual inspection of the predicted trajectories supports the conclusion that SynthNet generally outperforms the simplified Baseline method. Additional practical benefits of SynthNet over the reference method are that our proposed approach is more suitable for real-life scenarios because it does not need the camera parameters of the trajectory which the optimizer relies on. This implies that SynthNet does not require estimates of the camera parameters for the video. It instead extracts the aforementioned features. Additionally, since SynthNet is already trained, running inference is much faster than with the optimizer as it needs to search for the optimal set of initial conditions for every shot individually.

4.7 Limitations

While the unique approach of SynthNet demonstrates encouraging results for monocular 3D reconstruction in tennis, several limitations and challenges persist.

Considering that some players can shoot the ball with a velocity of up to 70 m/s, the ball can travel 2.3 meters in one frame. This implies that the provided frame rate of 30 frames per second (FPS) introduces limitations. When the ball moves faster than the FPS can capture, the bounces and hits are likely not exactly in the annotated frames but somewhere between two frames. In this scenario, the landing error is misleading because

we assume that the ball hits the ground in the annotated bounce. Therefore, the error could be decreased with a higher frame rate.

Another limitation is the metrics to evaluate the models. The only metric that can reliably describe the performance of the 3D-constructed trajectory is the reconstruction error, which we do not have for the real data. Since the reprojection error does not capture important aspects, we introduced new metrics to gauge the performance of our method more accurately. However, these additional metrics only examine the trajectory ending, not its starting position.

Lastly, we observed the situation where the ball bounces on the net poses challenges. This situation is annotated as a non-hit but appears very similar to a bounce. Furthermore, 3D trajectory reconstruction fails hard in these cases because 3D trajectory reconstruction does not consider these cases when modeling with Euler's method from initial conditions. This has a fatal effect on performance and yields unusually high reprojection errors which we choose to filter out in the worst cases.

While bouncing on the net is an uncommon occurrence in real-world scenarios, a model tailored to this domain should be able to handle those edge-cases.

4.8 Future Works

In this study, we choose to estimate hit-to-bounce trajectories. A possible addition is a method that models the bounce-to-hit trajectories. Another future avenue entails fine-tuning the synthetically trained model on real data. Additionally, adding more data with varying camera angles would likely make it more robust to different camera angles and out-of-distribution shots. Alternatively, creating synthetic camera parameters could have prevented the issues with the initial conditions in the y-direction. Lastly, experimenting with incorporating spin in the reconstructed 3D trajectories by adding 3D additional initial conditions to the model output yielded no conclusive positive results. However, this will be a priority in future approaches, as spin is an essential factor in tennis.

Table 5: Optimizer and SynthNet performance on HBNET + WASB trajectories

	Image			World		
Model	$\overline{RE}$	$\widetilde{RE}$	T.acc	T.F1	$\overline{LE}$	$\widetilde{LE}$
Optimizer	**12.20 px**	**6.55 px**	40.71%	0.371	5.11 m	3.09 m
SynthNet	41.31 px	31.82 px	**53.85**%	**0.398**	**3.78 m**	**2.4 m**

5 CONCLUSION

We proposed an end-to-end pipeline that reconstructs 3D tennis ball trajectories from monocular video.

The first part of the pipeline is a GRU-based model, which uses image trajectory, player positions, and court corners to detect hits and bounces in tennis matches. Using these predictions to define shots and their trajectories, we test a neural network to predict initial conditions for a 3D reconstruction of the shots' image trajectory. The method achieves good results in both reprojection error, in our own proposed metrics, and when visually inspecting the created trajectories. However, it has difficulties determining the exact start positions, especially for the player furthest from the camera, as well as struggles with outlier shots.

We compare our results to a simplified established approach from previous research and conclude that our methods achieve better results. Although our method achieves a worse reprojection error, it performs better on landing error, tile accuracy & tile f1-score, and 3D reconstruction error. Furthermore, SynthNet is more efficient and generalizes well, since it does not rely on camera parameters. Thus, we believe our approach is more reliable for the reconstruction of 3D trajectories.

REFERENCES

[1] Vanyi Chao, Ankhzaya Jamsrandorj, Yin May Oo, Kyung-Ryoul Mun, and Jinwook Kim. 2023. 3D Ball Trajectory Reconstruction of a Ballistic Shot from a Monocular Basketball Video. In *IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 1–6. https://doi.org/10.1109/IECON51785.2023.10312079 ISSN: 2577-1647.

[2] Hua-Tsung Chen, Wen-Jiin Tsai, Suh-Yin Lee, and Jen-Yu Yu. 2012. Ball tracking and 3D trajectory approximation with applications to tactics analysis from single-camera volleyball sequences. *Multimedia Tools and Applications* 60, 3 (Oct. 2012), 641–667. https://doi.org/10.1007/s11042-011-0833-y

[3] Dirk Farin, Susanne Krabbe, Peter H. N. De With, and Wolfgang Effelsberg. 2003. Robust camera calibration for sport videos using court models. In *Storage and Retrieval Methods and Applications for Multimedia 2004*, Minerva M. Yeung, Rainer W. Lienhart, and Chung-Sheng Li (Eds.), Vol. 5307. SPIE, San Jose, CA, 80–91. https://doi.org/10.1117/12.526813

[4] Megan Fazio, KS Fisher, and Tori Fujinami. 2018. Tennis ball tracking: 3-D trajectory estimation using smartphone videos. *Department of Electrical Engineering, Stanford University* (2018).

[5] Q Huang, S Cox, F Yan, TE deCampos, D Windridge, J Kittler, and W Christmas. 20110831 - 20110903. Improved Detection of Ball Hit Events in a Tennis Game Using Multimodal Information, In International Conference on Auditory-Visual Speech Processing. *11th International Conference on Auditory-Visual Speech Processing (AVSP)* (20110831 - 20110903).

[6] Qiang Huang, Stephen Cox, Xiangzeng Zhou, and Lei Xie. 2012. Detection of ball hits in a tennis game using audio and visual information. In *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*. IEEE, 1–10.

[7] Yu-Chuan Huang, I.-No Liao, Ching-Hsuan Chen, Tsì-Uí İk, and Wen-Chih Peng. 2019. TrackNet: A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications. http://arxiv.org/abs/1907.03698 arXiv:1907.03698 [cs, stat].

[8] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. 2023. Ultralytics YOLO. Retrieved may 1, 2024 from https://github.com/ultralytics/ultralytics

[9] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. 2023. Ultralytics YOLO. Retrieved may 1, 2024 from https://github.com/ultralytics/ultralytics

[10] S.X. Ju, M.J. Black, and Y. Yacoob. 1996. Cardboard people: a parameterized model of articulated image motion. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*. IEEE Comput. Soc. Press, Killington, VT, USA, 38–44. https://doi.org/10.1109/AFGR.1996.557241

[11] Jacek Komorowski, Grzegorz Kurzejamski, and Grzegorz Sarwas. 2019. DeepBall: Deep Neural-Network Ball Detector. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications. https://doi.org/10.5220/0007348902970304

[12] Paul Liu and Jui-Hsien Wang. 2022. MonoTrack: Shuttle trajectory reconstruction from monocular badminton video. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 3512–3521. https://doi.org/10.1109/CVPRW56347.2022.00395

[13] Peng Lu, Tao Jiang, Yining Li, Xiangtai Li, Kai Chen, and Wenming Yang. 2024. RTMO: Towards High-Performance One-Stage Real-Time Multi-Person Pose Estimation. http://arxiv.org/abs/2312.07526 arXiv:2312.07526 [cs].

[14] M. J. D. Powell. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput. J.* 7, 2 (Jan. 1964), 155–162. https://doi.org/10.1093/comjnl/7.2.155

[15] Jinchang Ren, James Orwell, Graeme A. Jones, and Ming Xu. 2009. Tracking the soccer ball using multiple fixed cameras. *Computer Vision and Image Understanding* 113, 5 (2009), 633–642. https://doi.org/10.1016/j.cviu.2008.01.007 Computer Vision Based Analysis in Sport Environments.

[16] Kosolapov Sergey. 2023. *TennisCourtDetector*. https://github.com/yastrebksv/TennisCourtDetector Accessed: March 2024.

[17] Lejun Shen, Qing Liu, Lin Li, and Haipeng Yue. 2016. 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)*, Paul Chung, Andrea Soltoggio, Christian W. Dawson, Qinggang Meng, and Matthew Pain (Eds.). Vol. 392. Springer International Publishing, Cham, 33–39. https://doi.org/10.1007/978-3-319-24560-7_5 Series Title: Advances in Intelligent Systems and Computing.

[18] Maria Skublewska-Paszkowska, Paweł Powroźnik, and Edyta Łukasik. 2020. Learning Three Dimensional Tennis Shots Using Graph Convolutional Networks. *Sensors* 20 (10 2020), 6094. https://doi.org/10.3390/s20216094

[19] Shuhei Tarashima, Muhammad Abdul Haq, Yushan Wang, and Norio Tagawa. 2023. Widely Applicable Strong Baseline for Sports Ball Detection and Tracking. http://arxiv.org/abs/2311.05237 BMVC2023.

[20] Gabriel Van Zandycke and Christophe De Vleeschouwer. 2019. Real-time CNN-based Segmentation Architecture for Ball Detection in a Single View Setup. In *Proceedings Proceedings of the 2nd International Workshop on Multimedia Content Analysis in Sports*. ACM. https://doi.org/10.1145/3347318.3355517

[21] Fei Wang, Lifeng Sun, Bo Yang, and Shiqiang Yang. 2006. Fast Arc Detection Algorithm for Play Field Registration in Soccer Video Mining. In *2006 IEEE International Conference on Systems, Man and Cybernetics*, Vol. 6. IEEE, 4932–4936. https://doi.org/10.1109/ICSMC.2006.385087 ISSN: 1062-922X.

[22] T. Watanabe, M. Haseyama, and H. Kitajima. 2004. A soccer field tracking method with wire frame model from TV images. In *2004 International Conference on Image Processing, 2004. ICIP '04.*, Vol. 3. IEEE, 1633–1636 Vol. 3. https://doi.org/10.1109/ICIP.2004.1421382 ISSN: 1522-4880.

[23] Qingyu Xiao, Zulfiqar Zaidi, and Matthew Gombolay. 2024. Multi-Camera Asynchronous Ball Localization and Trajectory Prediction with Factor Graphs and Human Poses. http://arxiv.org/abs/2401.17185 arXiv:2401.17185 [cs].

[24] Jia xin Cai and Xin Tang. 2018. RGB Video Based Tennis Action Recognition Using a Deep Historical Long Short-Term Memory. *arXiv: Computer Vision and Pattern Recognition* (2018). https://api.semanticscholar.org/CorpusID:52824593

[25] Fei Yan. 2005. Tennis ball tracking for automatic annotation of broadcast tennis video. *Proceedings of the British Machine Vision Conference* (2005). https://www.semanticscholar.org/paper/Tennis-ball-tracking-for-automatic-annotation-of-Yan/d135b747e99e6a06f5ecac5462b53c1b7bd259e2?p2df