

Appendix: Where Is The Ball: 3D Ball Trajectory Estimation From 2D Monocular Tracking

A. Overview

In this Appendix, we present:

- Section B: Simulation details.
- Section C: Dataset details.
- Section D: Implementation details and network architectures.
- Section E: Runtime.
- Section F: Additional results.
- Section G: Failure cases.

B. Simulation details

We used Unity (v2019.3.2f1) with PhysX engine (v3.3) to simulate ball trajectories. The ground plane was created using a box collider object, and its center position was set to the origin. We used a sphere collider object with the property “rigid” for the ball. The camera parameters were manually set based on real-world parameters (estimated from three real datasets). For all simulations, we take into account the ball’s size, weight, and a plausible range of ball speeds (by varying the applied force). Other factors, such as ball spin, aerodynamics, and court type (e.g., grass), are not considered in the current setup but could be incorporated in future work to enhance realism of the simulation, as they influence friction and bounce behavior.

B.1. Mocap and IPL

To simulate a bouncing ball with multiple trajectories, we applied an impulse force at the beginning and let the ball bounce until its velocity dropped below a threshold, indicating that it had nearly stopped moving, before applying a new force. These forces had random magnitudes, and their directions were randomly generated so that projectile motions and rolling motions on the ground occurred with an equal chance. Note that projectile motions are generated using forces with positive y components, assuming y+ points upward, and rolling motions using forces with zero y component. The end-of-trajectory flag was only set true at the time step right before each force was being applied. The simulation of Mocap and IPL in Unity Game Engine is shown in Figure 8.

B.2. Tennis

To simulate tennis shots, we built upon an open-source tennis game [5] and made the gameplay between two computer-bot players for the ease of data collection. Each bot has 2 actions: hit and receive. The hitter bot will randomly pick a location on the opponent’s side for the ball to

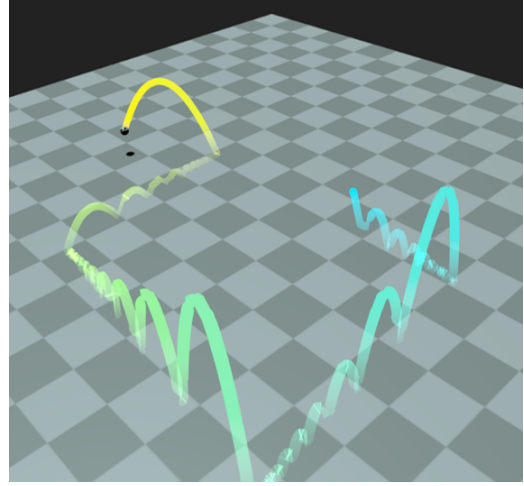


Figure 8. Unity Game Engine for Mocap, IPL and Simpler synthetic datasets.

land and make a hit with random angles between 10-20 degrees (creating a lob shot or a flat shot). Then, the receiver bot will receive the ball behind the landing position with an offset of 5-7 meters away and subsequently becomes the hitter, and vice versa. The end-of-trajectory flag was only set true at the time step right before the bots make a hit. Additionally, the net was created using a box collider object for filtering out trajectories that are not passing over the net. The tennis simulation in Unity Game Engine is shown in Figure 9.

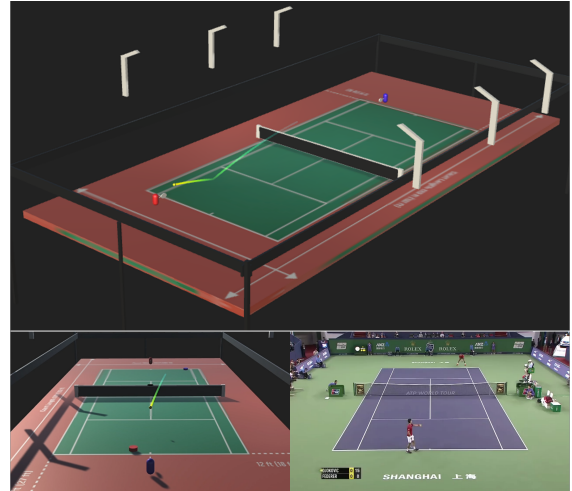


Figure 9. Unity Game Engine for Tennis.

B.3. Synthetic Single-Launch Trajectory Dataset

For comparison with Mocanu et al. [31] and Shen et al. [47], we matched their input assumptions (single force / single trajectory) and simulated single-trajectory sequences by

launching the ball from the origin into a random direction in the first quadrant (0-90 degrees). Other configurations are similar to those used in Mocap and IPL.

C. Dataset details

In this section, we explain details of our synthetic datasets and real datasets. For synthetic datasets, the train, validation and test sets consist of 5000, 1500 and 500 sequences.

C.1. Synthetic Mocap

In this dataset, the sequence length varies between 85-2,873 time steps with an average of 460 time steps. The space for the ball to travel is about $11.11 \times 10.92\text{m}^2$ with the maximum height of 1.58m. Each sequence in the train set contains two consecutive trajectories, whereas in the validation and test sets, each sequence contains 1-7 consecutive trajectories.

C.2. Synthetic IPL

In this dataset, the sequence length varies between 37-949 time steps with an average of 104 time steps. The space for the ball to travel is about $30 \times 75\text{m}^2$ with the maximum height of 10.42m. Similar to Synthetic Mocap, each sequence in the train set contains two consecutive trajectories, while in the validation and test sets, 1-7 consecutive trajectories.

C.3. Synthetic Tracknet (Tennis)

In this dataset, the sequence length varies between 64-822 time steps with an average of 122 time steps. The space for the ball to travel is about $18.11 \times 37.12\text{m}^2$ with the maximum height of 3.87m. All sequences in train, validation, and test sets contain 3 strokes.

C.4. Synthetic Single-Launch Trajectory Dataset

This synthetic dataset for state-of-the-art comparison with Mocanu et al. [31] and Shen et al.[47] has 300, 100 and 100 sequences for train, validation and test set. This dataset has the minimum and maximum sequence lengths of 46 and 106 time steps. The average sequence length of trajectories is 74 time steps. The space for the ball to travel is about $4.48 \times 4.43\text{m}^2$ with the maximum height of 0.77m.

C.5. Real IPL

IPL soccer ball detection dataset [17] contains short video streams of a real soccer match from 6 synchronized cameras at 25fps. The 2D ball tracking sequences are provided, and we followed the camera pose estimation pipeline in [39] to estimate the 3D ball positions used as ground truth. In this dataset, the 2D track points are sometimes missing for a few frames, e.g., due to occlusion. We describe our method to fill in these missing data in section D.1. Then, we used the

completed trajectories as input to our model. There is a total of 9 remaining sequences that were successfully calibrated from the above process and satisfied our assumption that the ball starts and ends on the ground. The minimum and maximum sequence lengths are 18 and 147 and the average is 68 time steps. The ball travels within a space of size $30.60 \times 47.07\text{m}^2$ with the maximum height of 1.66m.



Figure 10. Motion capture studio. The top left is a ping-pong ball attached with IR reflective materials. The right image is our motion capture studio used to collect data. The bottom left is one of the eight IR cameras used in the studio.

C.6. Real Mocap

This dataset captures the bouncing motion of a ping pong ball in a motion capture studio shown in Figure 10. This system uses 8 synchronized IR cameras to track IR reflective stickers that were attached on the ping-pong ball with a 40mm diameter. The camera frame rate was set to 50fps. The Mocap system provided the 3D positions of the ping-pong ball with a 2D tracking sequence from each camera with known parameters. We used all cameras with their 2D tracking sequences as input to our method for evaluation. We generated bouncing motions by throwing the ball upward within this space and kept re-throwing whenever the ball stopped moving from that last spot. This dataset contains 344 different trajectories, and the maximum number of consecutive trajectories is 3. The minimum and maximum sequence lengths are 150 and 907 and the average is 301 time steps. The space for the ball to travel is about $6.21 \times 3.74\text{m}^2$ and the maximum height is 1.49m.

C.7. Real Tracknet (Tennis)

This dataset [20] contains 81 video clips of 10 tennis matches captured from a 30FPS broadcast camera. The 2D ball tracking annotations are also provided. We qualitatively evaluate our performance on 118 trajectories from 13 clips in one match. The minimum and maximum sequence lengths are 18 and 288, and the average is 92 time steps. Each sequence has a varying number of strokes between 1 to 10 (with an average of 3 strokes) and the tennis ball bounces 9 times at most (with an average of 4 bounces).

D. Implementation details / Network architectures

For training, we set $(\lambda_\epsilon, \lambda_{3D}, \lambda_B) = (10, 1, 10)$ and trained our networks for 1,400 epochs using Adam optimizer [25] with a constant learning rate of 0.001 and a batch size of 256. We trained our LSTMs with backpropagation through time. Note that our trained pipeline can still predict output sequences of arbitrary lengths. We also randomly add a Gaussian noise to each 2D input location (u_t, v_t) to simulate noisy 2D tracking from a tracking algorithm or human labels. Results for different levels of noise are reported in the main paper in Table 5.

Next, we explain the network architectures of:

1. EoT prediction network (LSTM^ϵ) in Table 5.
2. Height prediction network ($\text{LSTM}^{f,b}$ and $\text{LSTM}^{\text{height}}$) in Table 6, 7.
3. Refinement network ($\text{LSTM}^{\text{refine}}$) in Table 8.

Note that in these tables, B is the batch size, L is the sequence length, and all LeakyReLUs use 0.01 slope.

Table 5. Network architecture of the EoT prediction network (LSTM^ϵ).

Layer	Activation	Output size
Input	-	$B \times L \times 4$
BiLSTM.0	-	$B \times L \times 2 \times 64$
BiLSTM.1	-	$B \times L \times 2 \times 64$
+ output of BiLSTM.0 (residual) BiLSTM.2	-	$B \times L \times 2 \times 64$
Concat	-	$B \times L \times 128$
FC.0	Leaky ReLU	$B \times L \times 32$
FC.1	Leaky ReLU	$B \times L \times 32$
FC.2	Leaky ReLU	$B \times L \times 32$
FC.3	Sigmoid	$B \times L \times 1$

Table 6. Network architecture of the $\text{LSTM}^{f,b}$ in height prediction network. Note that we use the same architecture for both the forward and backward directions.

Layer	Activation	Output size
Input	-	$B \times L \times 6$
LSTM.0	-	$B \times L \times 1 \times 64$
LSTM.1	-	$B \times L \times 1 \times 64$
LSTM.2	-	$B \times L \times 1 \times 64$
Concat	-	$B \times L \times 64$
FC.0	Leaky ReLU	$B \times L \times 32$
FC.1	Leaky ReLU	$B \times L \times 32$
FC.2	Leaky ReLU	$B \times L \times 32$
FC.3	-	$B \times L \times 1$

D.1. Filling in missing track points (IPL dataset)

In IPL dataset [17], there are missing data points in some time steps in the 2D tracking sequences. We solve this problem with an additional pre-processing step that fills in the

Table 7. Network architecture of the $\text{LSTM}^{\text{height}}$ in height prediction network.

Layer	Activation	Output size
Input	-	$B \times L \times 5$
BiLSTM.0	-	$B \times L \times 2 \times 64$
BiLSTM.1	-	$B \times L \times 2 \times 64$
+ output of BiLSTM.0 (residual) BiLSTM.2	-	$B \times L \times 2 \times 64$
Concat	-	$B \times L \times 128$
FC.0	Leaky ReLU	$B \times L \times 32$
FC.1	Leaky ReLU	$B \times L \times 32$
FC.2	Leaky ReLU	$B \times L \times 32$
FC.3	-	$B \times L \times 1$

Table 8. Network architecture of the refinement network.

Layer	Activation	Output size
Input	-	$B \times L \times 7$
BiLSTM.0	-	$B \times L \times 2 \times 64$
BiLSTM.1	-	$B \times L \times 2 \times 64$
+ output of BiLSTM.0 (residual) BiLSTM.2	-	$B \times L \times 2 \times 64$
Concat	-	$B \times L \times 128$
FC.0	Leaky ReLU	$B \times L \times 32$
FC.1	Leaky ReLU	$B \times L \times 32$
FC.2	Leaky ReLU	$B \times L \times 32$
FC.3	-	$B \times L \times 3$

missing points before using the completed sequence as input to our main pipeline and other competing techniques. In particular, we trained an auto-regressive network also based on LSTMs that takes as input the temporal difference of 2D pixel coordinates $(\Delta u_t, \Delta v_t)$ and predicts the difference for the next time step $(\Delta u_{t+1}, \Delta v_{t+1})$, following [18]. This network consists of 2 independent directional-LSTMs that auto-regress the sequence in the forward and backward directions shown in Table 9. The resulting two predicted sequences are combined with linear ramp weighting similar to Eq. 3 in the main paper, to output the final 2D tracking sequence. Note that if a tracking data point is available for the current time step, we simply use it. We trained this network with the teacher forcing technique [49].

Table 9. Network architecture of the auto-regressive model for interpolating missing data points. Note that we used the same architecture for both forward and backward directions.

Layer	Activation	Output size
Input	-	$B \times L \times 2$
LSTM.0	-	$B \times L \times 64$
LSTM.1	-	$B \times L \times 64$
+ output of LSTM.0 (residual) LSTM.2	-	$B \times L \times 64$
+ output of LSTM.0 and LSTM.1 (residual) LSTM.3	-	$B \times L \times 64$
FC.0	Leaky ReLU	$B \times L \times 64$
FC.1	Leaky ReLU	$B \times L \times 32$
FC.2	Leaky ReLU	$B \times L \times 16$
FC.3	Leaky ReLU	$B \times L \times 8$
FC.4	Leaky ReLU	$B \times L \times 4$
FC.5	-	$B \times L \times 2$

E. Runtime

We measured runtime on the test set of Simpler Synthetic dataset (Appendix C.4), which contains 100 trajectories (7,463 timesteps in total). We tested our method and other competing techniques on 100 trajectories for 100 times (10,000 sequences) on a desktop with AMD Ryzen 9 3900X and a single NVIDIA 2080 super. Our method took an average of 1.01 ± 0.11 ms per frame, which is about $8.6\times$ faster than the other learning-based Mocanu et al. [31] (8.7 ± 1 ms). The physics-based method, Shen et al. [47], only requires optimization and is the fastest with an average runtime of 0.012 ± 0.003 ms per frame.

F. Additional results

In this section, we provide an additional prior work comparison on two real datasets, additional error metrics, as well as additional qualitative results for three real and three synthetic datasets.

F.1. Comparison with prior work on Real Mocap and Real IPL

We compare our method to the same state-of-the-art methods [31, 47] used in Section 4.2 of the main paper, but each test example in this experiment contains multiple trajectories due to multiple acting forces (e.g., tennis hits). Note again that these prior methods are not designed for multiple trajectories, but we include this experiment for completeness. We performed a fair comparison using a single-launch trajectory test set in Section 4.2.

Table 10 reports distance NRMSEs on the test sets of Real Mocap and IPL datasets. Our method achieves significantly better NRMSEs with performance gaps of up to 75.4 in Mocap and 13.6 in IPL, but this is expected as these test scenarios violate their assumptions.

F.2. Results using other NRMSE variants

Table 11 reports different variants of NRMSEs, which are $\text{RMSEs} \times 100\%$ divided by the trajectory height, area’s length, area’s width, or the distance to camera, following [11]. Here the length and width are the field dimensions (e.g., tennis court ($23.27 \times 10.97m^2$) or soccer pitch ($105 \times 69.5m^2$)). We report NRMSEs for *distance*, based on the L2 distance on the xyz coordinates, and *height*, based on the distance along the y coordinate only. Since our method may exhibit errors relative to the size of the playing area, these metrics are important for assessing our performance for different applications or different world scales. For example, when visualizing the soccer ball in the entire soccer field, errors with respect to the area’s length or width may be appropriate.

For Real Mocap, we achieve a 0.48% distance NRMSE

with respect to both the area’s length and width. For IPL, the errors are 1.13% and 1.71% with respect to the soccer pitch’s dimensions, or 1.13% with respect to the camera distance, which is about 106m away from the soccer pitch.

F.3. Other quantitative metrics

We show quantitative results from all experiments and ablation studies in RMSEs (in centimeters) in Table 12-16. Additionally, we report the statistics of ground penetration in the predicted trajectories on Real Tracknet in Table 17.

F.4. Qualitative results

We present additional qualitative results on synthetic datasets of Mocap, IPL, and Tracknet in Figure 11, and on their real counterparts separately in Figures 12-14. Lastly, a comparison with the state of the art is shown in Figure 15.

G. Failure cases

We observed that our method performs worse on unusual trajectories that are substantially different from the simulated trajectories. Some rare trajectories in tennis include volley shots (the player returns the ball before it bounces off the ground), or when the player strikes near the net, while in soccer, when the player chests the ball. We show these failure cases on Mocap, Tracknet (Tennis) and IPL datasets in Figure 16, 17 and 18.

Table 10. Comparison with prior work on Real Mocap and Real IPL. The numbers are NRMSEs. Note that each test example in these datasets contains multiple trajectories, which are outside prior work’s assumptions.

Method	Real	
	Mocap	IPL
Mocanu et al.[31]	15.92	14.33
Shen et al. [47]	76.09	5.03
Ours	0.68	0.74

Table 11. We report NRMSEs with respect to the trajectory height, area’s length, area’s width, and distance to camera, following Calendre et al. [11]. *Each row shaded in gray shows the denominators (meter) used to compute each normalized RMSE.

Metric	Synthetic						Real			
	Mocap		Tennis		IPL		Mocap		IPL	
	Distance	Height	Distance	Height	Distance	Height	Distance	Height	Distance	Height
RMSE (cm)	1.33	0.48	8.48	2.25	3.43	0.80	3.83	2.15	119.15	26.04
Trajectory height (m)	1.58		3.87		10.42		1.49		1.66	
%NRMSE	0.84	0.30	2.19	0.58	0.33	0.08	2.59	1.45	71.77	15.68
Area's length (m)	10.92		23.77		75.00		8.00		105.00	
%NRMSE	0.12	0.04	0.36	0.09	0.05	0.01	0.48	0.27	1.13	0.25
Area's width (m)	11.11		10.97		30.00		8.00		69.50	
%NRMSE	0.12	0.04	0.77	0.21	0.11	0.03	0.48	0.27	1.71	0.37
Distance to camera (m)	6.37		32.84		105.66		6.37		105.66	
%NRMSE	0.21	0.08	0.26	0.07	0.03	0.01	0.60	0.34	1.13	0.25
%NRMSE (RMSE / (max - min))	0.09	-	0.15	-	0.02	-	1.01	-	1.03	-

Table 12. Ablation study on input/output parameterization. We evaluate our full pipeline with different types of input / output parameterization. The numbers are RMSEs of distance and height measured in centimeter.

Parameterization		Synthetic						Real			
		Mocap		Tennis		IPL		Mocap		IPL	
Input	Output (before refine.)	Distance	Height	Distance	Height	Distance	Height	Distance	Height	Distance	Height
Pixel	xyz	165.19	9.49	18.43	5.12	117.25	27.44	91.63	28.21	240.81	44.88
	height	2.82	0.72	19.29	3.96	13.78	2.98	52.12	24.77	265.79	50.62
Pixel + E	xyz	96.13	75.14	34.96	11.57	81.39	17.63	160.22	28.02	221.26	39.41
	height	3.54	1.11	16.78	3.46	8.25	1.95	52.36	25.07	273.175	53.17
$\mathbf{p}_{\text{ground}} + (\varphi_{az}, \theta_{el})$	xyz	8.33	6.03	13.52	3.22	73.7	16.06	51.95	11.92	225.50	44.65
	height	4.11	1.29	13.51	2.68	4.53	1.21	28.90	13.67	248.31	47.21
$\mathbf{p}_{\text{ground}} + (\varphi_{az}^{\sin, \cos}, \theta_{el}^{\sin, \cos})$	xyz	2.25	1.18	14.06	3.45	127.58	29.86	21.89	6.04	271.52	55.25
	height	13.48	5.47	12.33	2.45	5.78	1.46	6.77	3.50	130.27	27.28
$\mathbf{p}_{\text{ground}} + \mathbf{p}_{\text{vertical}}$ (Ours)	xyz	2.23	1.12	13.02	3.12	83.02	22.59	5.13	2.57	296.55	56.96
	height (Ours)	1.33	0.48	8.48	2.25	3.43	0.80	3.83	2.15	119.15	26.04

Table 13. Ablation study on LSTM components. The numbers are RMSEs of distance and height measured in centimeter.

LSTM ^[*] components				Synthetic						Real			
				Mocap		Tennis		IPL		Mocap		IPL	
ε (EoT) f, b height refine				Distance	Height	Distance	Height	Distance	Height	Distance	Height	Distance	Height
-	✓	✓	✓	19.40	0.93	18.53	3.63	3.75	0.95	6.16	3.16	228.28	44.52
✓	-	✓	✓	3.58	1.05	13.33	2.61	4.30	1.04	4.10	3.35	290.21	56.01
✓	✓	-	✓	53.77	30.89	11.92	2.46	182.01	46.23	6.83	3.11	284.33	56.22
✓	✓	✓	-	2.80	0.92	11.89	2.37	8.66	1.93	4.71	3.86	272.84	54.21
-	✓	✓	-	4.96	1.91	20.73	4.17	14.54	3.39	4.92	2.63	374.85	73.09
✓	-	✓	-	3.57	1.04	14.07	2.85	47.87	2.08	6.68	3.58	315.66	60.85
✓	✓	-	-	125.4	53.33	32.01	7.07	555.15	128.91	29.79	16.17	258.76	52.13
✓	✓	✓	✓	1.33	0.48	8.48	2.25	3.43	0.80	3.83	2.15	119.15	26.04

Table 14. Ablation study on loss terms. We train our full pipeline with each loss term removed and report distance NRMSEs.

Loss	Synthetic			Real	
	Mocap	Tennis	IPL	Mocap	IPL
no \mathcal{L}_ε	0.15	0.25	0.06	0.84	1.24
no \mathcal{L}_B	0.09	0.27	0.05	0.87	1.34
no $\mathcal{L}_\varepsilon, \mathcal{L}_B$	0.23	0.29	0.08	0.98	3.16
Ours (all terms)	0.05	0.09	0.01	0.68	0.74

Table 15. Ablation study on loss terms. We train our full pipeline with each loss term removed. The numbers are RMSEs of distance and height measured in centimeter.

Loss	Synthetic						Real			
	Mocap		Tennis		IPL		Mocap		IPL	
	Distance	Height	Distance	Height	Distance	Height	Distance	Height	Distance	Height
no \mathcal{L}_ε	7.05	4.86	16.39	3.23	7.80	1.87	4.02	2.26	179.98	36.21
no \mathcal{L}_B	5.74	4.64	16.73	3.36	8.94	2.34	4.16	2.65	214.28	41.86
no $\mathcal{L}_\varepsilon, \mathcal{L}_B$	9.03	3.4	17.46	3.5	15.61	3.38	4.2	2.45	439.63	86.29
Ours (all terms)	1.33	0.48	8.48	2.25	3.43	0.80	3.83	2.15	119.15	26.04

Table 16. Comparison with prior work on Synthetic Mocap. The numbers are RMSEs of distance and height measured in centimeter for varying levels of noise in the input 2D trajectory.

Method	Synthetic Mocap											
	No noise		± 5 pixels		± 10 pixels		± 15 pixels		± 20 pixels		± 25 pixels	
	Distance	Height	Distance	Height	Distance	Height	Distance	Height	Distance	Height	Distance	Height
Mocanu et al. [31]	8.58	6.44	8.62	6.45	8.72	6.50	8.91	7.11	9.21	7.45	9.41	7.72
Shen et al. [47]	1.83	1.37	2.10	1.50	2.80	1.83	3.75	2.34	4.86	2.96	5.65	3.37
Ours	0.60	0.30	0.65	0.32	0.69	0.34	0.97	0.36	1.30	0.38	1.66	0.45

Table 17. Qualitative analysis on Real Tracknet (Tennis). We report the statistics of points mistakenly predicted below ground at different penetration distances.

Metric	Real Tracknet (Tennis)					
	0-2.5 cm.	2.5-5 cm.	5 - 7.5 cm.	7.5 - 10 cm.	10 - 25 cm.	25 - 50 cm.
#(predicted points below ground) (N=181)	49	51	29	17	34	1
as a percentage of #(ground contact points) (N=236)	20.76%	21.61%	12.29%	7.20%	14.41%	0.42%
as a percentage of #(all points) (N=10,844)	0.45%	0.47%	0.27%	0.16%	0.31%	0.01%

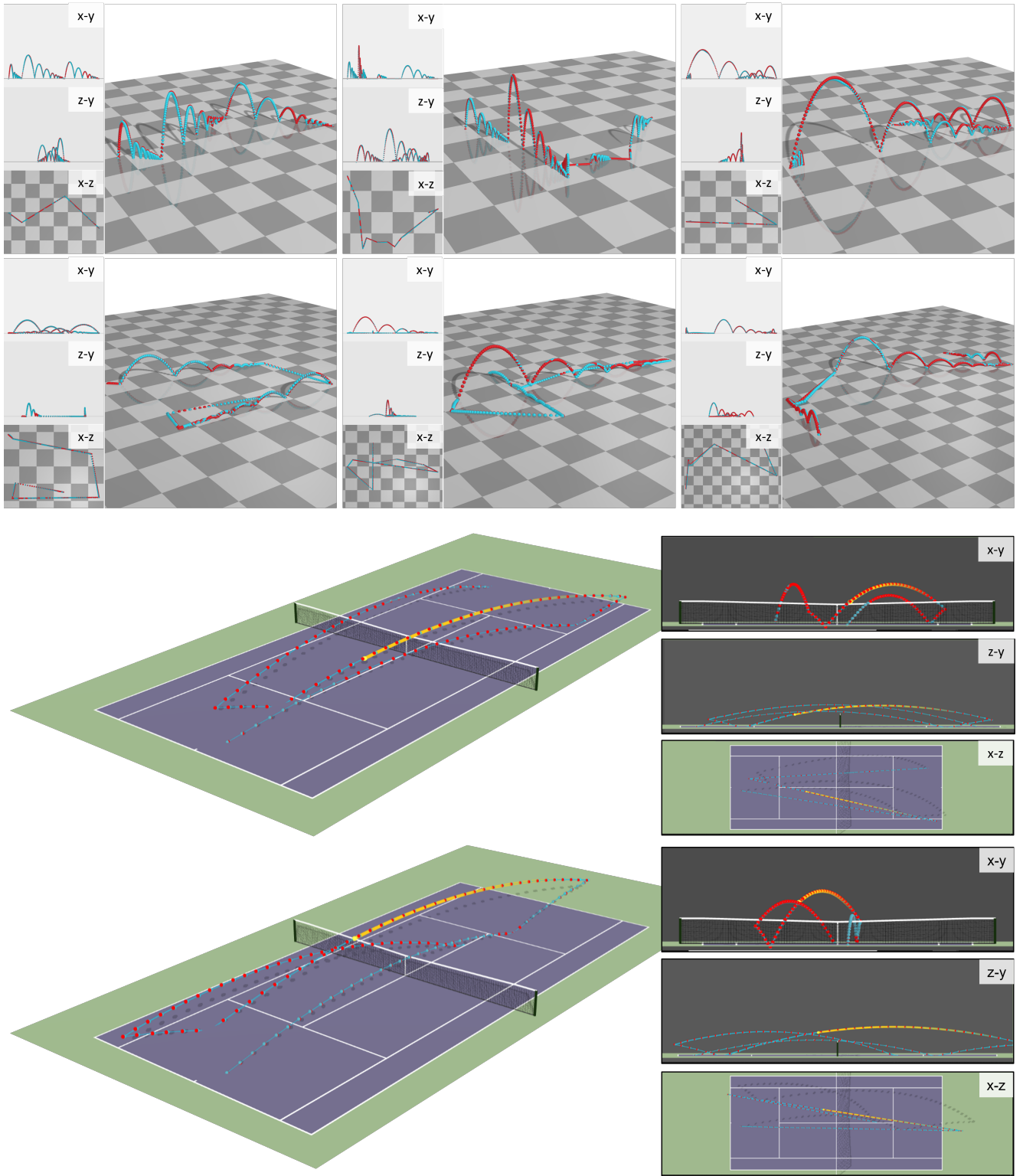


Figure 11. Qualitative results on synthetic datasets. Blue: our predictions. Red: ground truth. The first row is the results from Synthetic Mocap and each checkerboard block is $75 \times 75 \text{ cm}^2$. The second row is the results from Synthetic IPL and each checkerboard block is $250 \times 250 \text{ cm}^2$. The last two rows are the results from Synthetic Tennis.

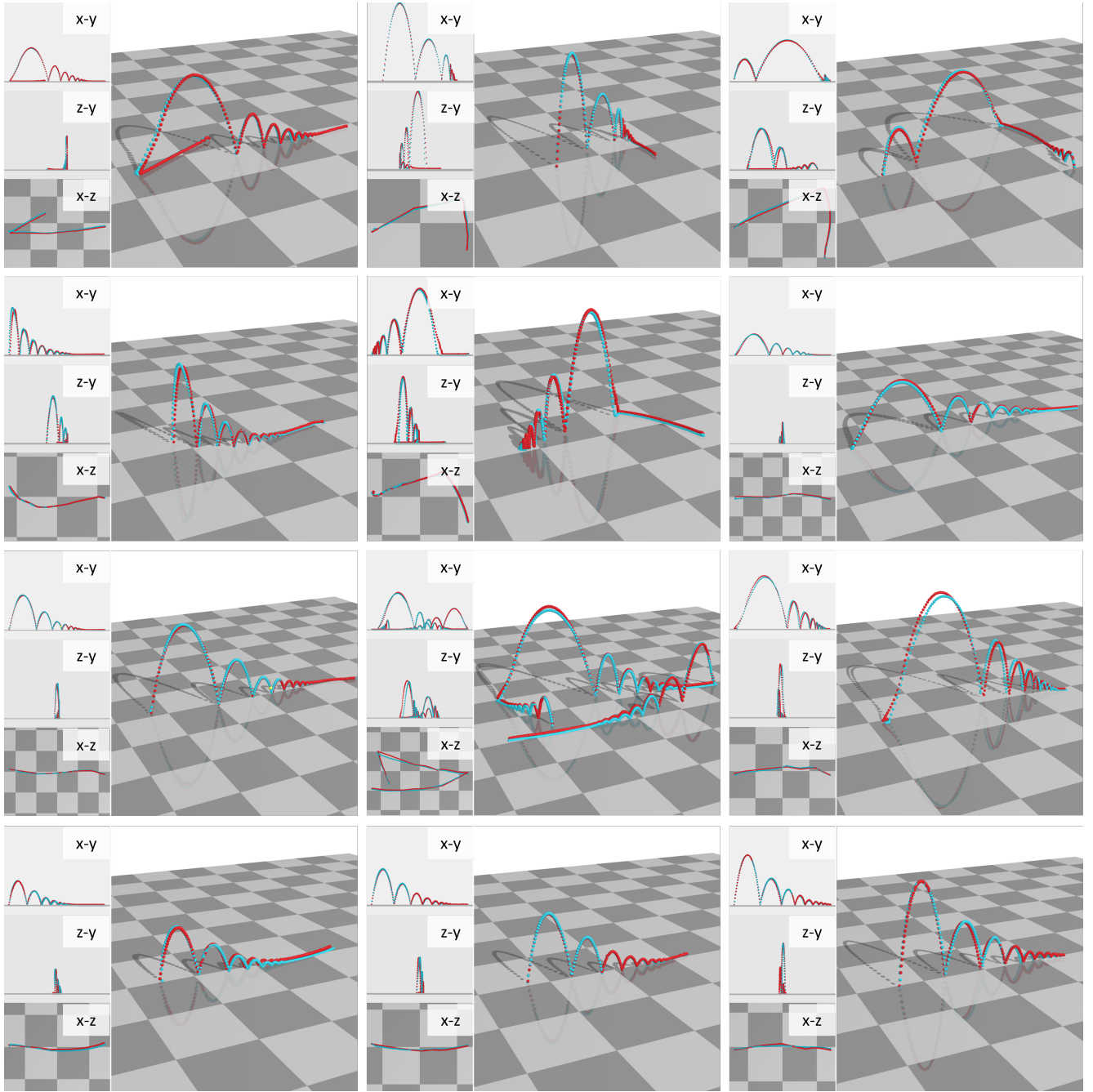


Figure 12. Qualitative results on Real Mocap dataset. Blue: our predictions. Red: ground truth. Each checkerboard block is 75×75 cm².

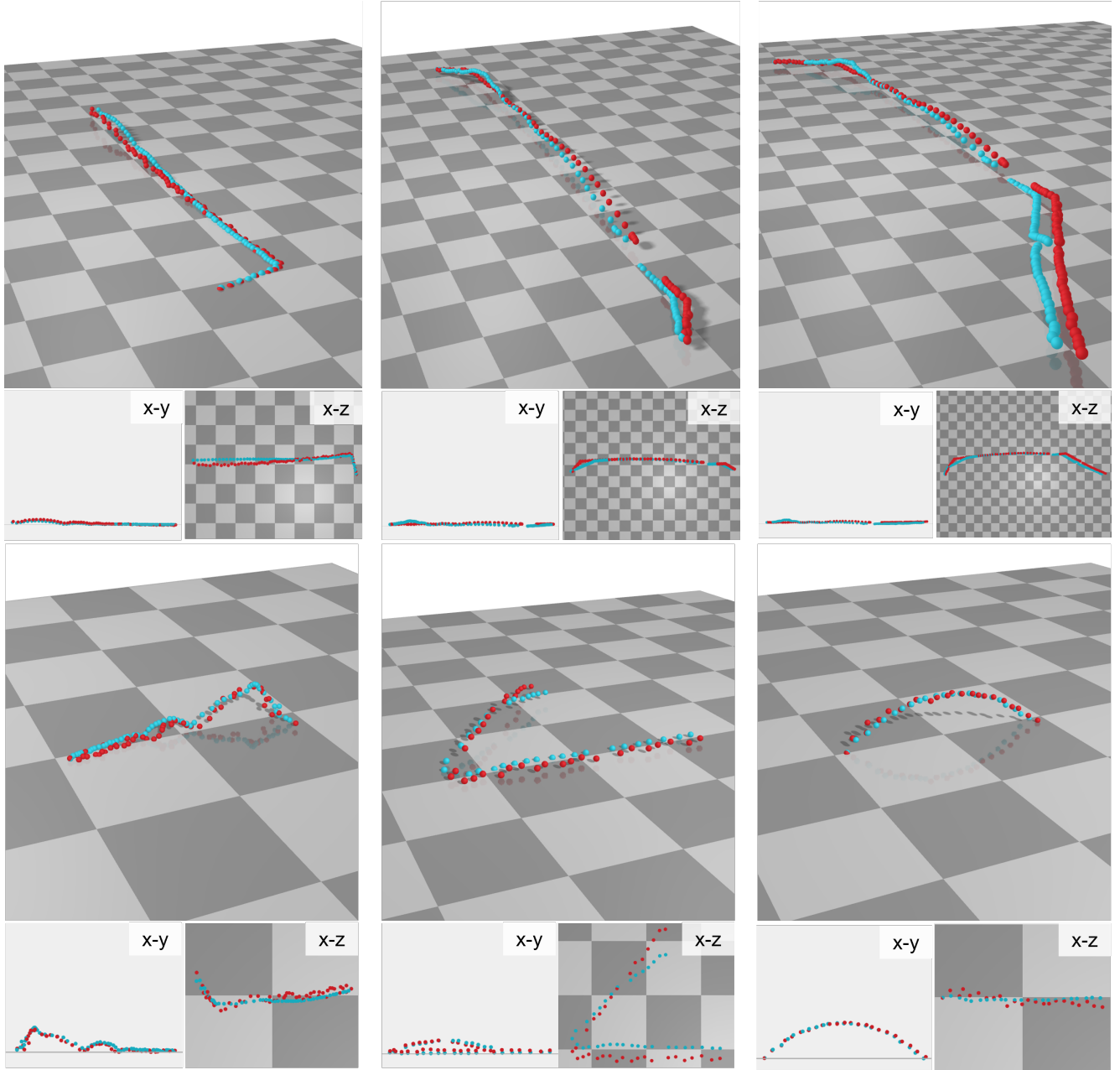


Figure 13. Qualitative results on Real IPL dataset. Blue: our predictions. Red: ground truth. Each checkerboard block is 250×250 cm².

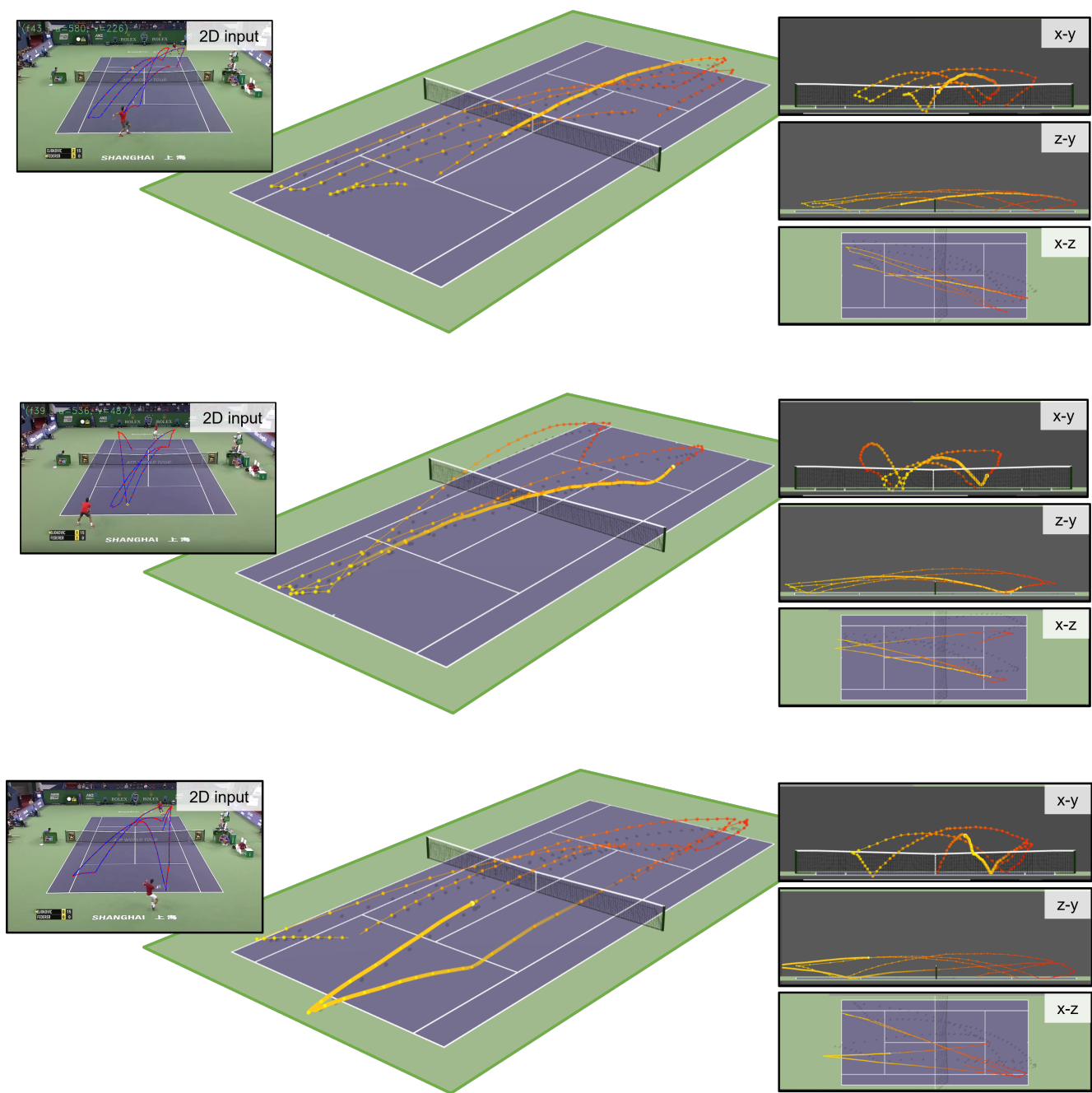


Figure 14. Qualitative results on Real Tracknet (Tennis) dataset.

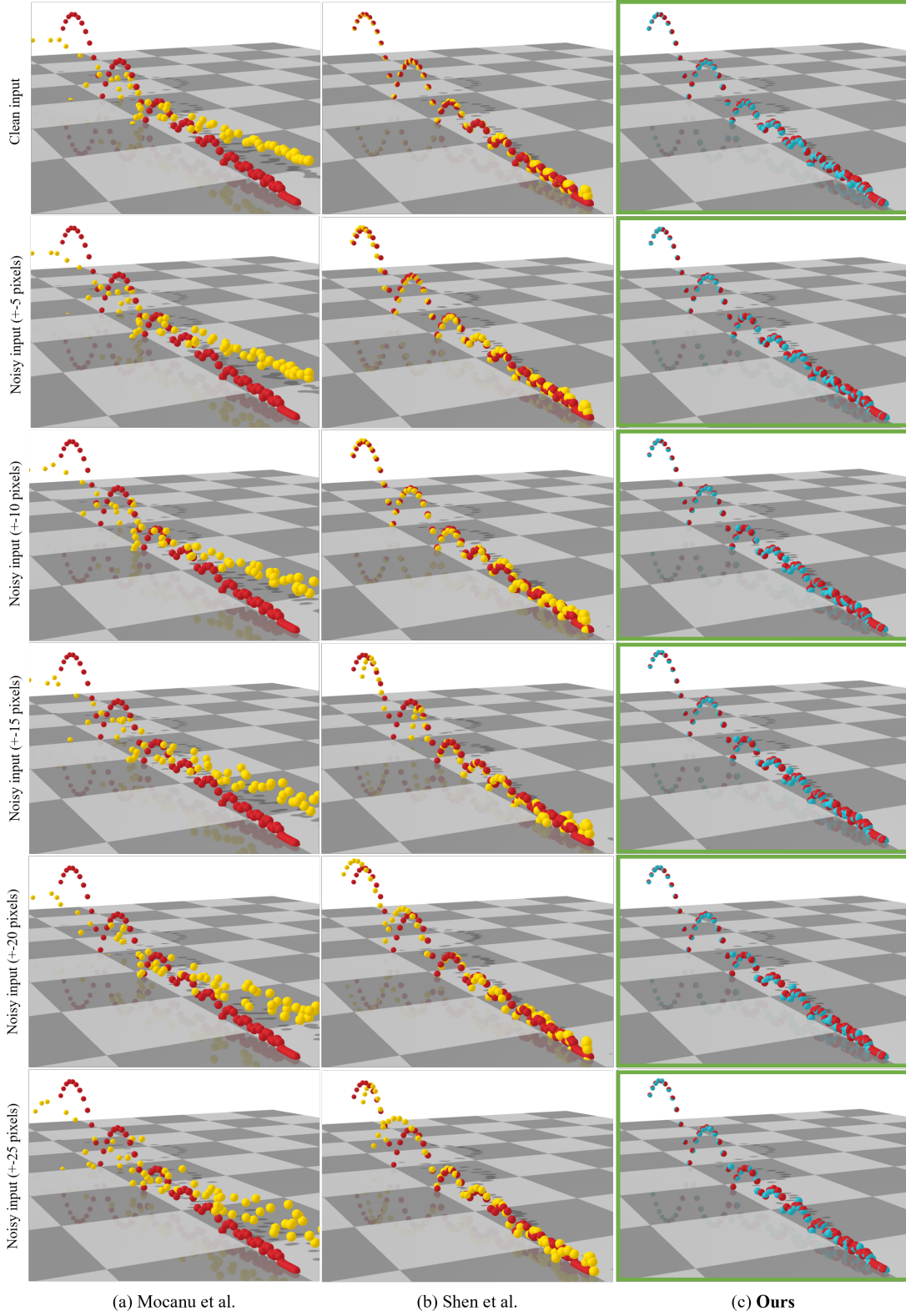


Figure 15. State-of-the-art comparison with a learning-based approach Mocanu et al.[31] and a physics-based approach Shen et al.[47] on a simplified test trajectory that matches their requirements. Each row uses a different noise level. Our predictions are shown in blue, prior work in yellow, and ground truth in red. Each checkerboard block is $50 \times 50 \text{ cm}^2$.

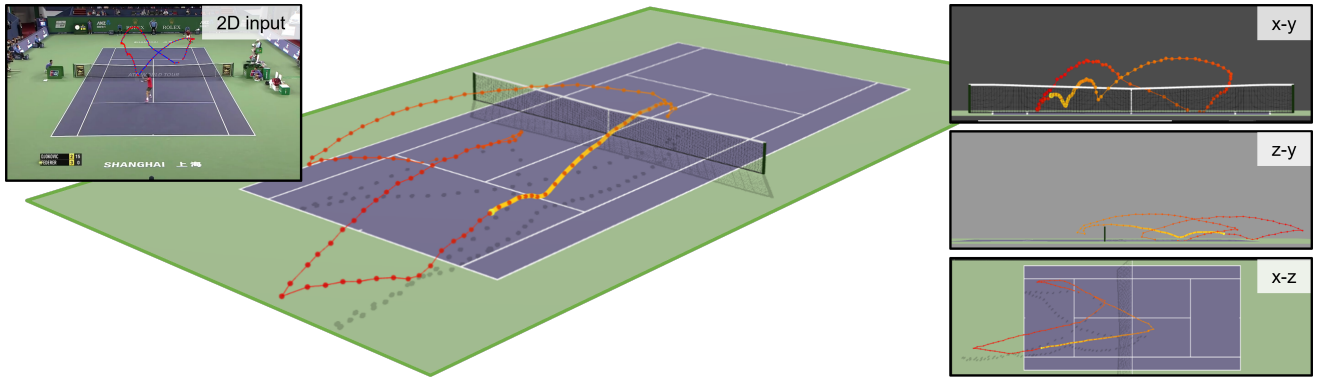


Figure 16. Failure cases on Real Tracknet(Tennis) dataset. This trajectory comes from a volley shot close to the net where the ball bounces right back without hitting the ground, but our prediction shows some slight drop in the ball's height.

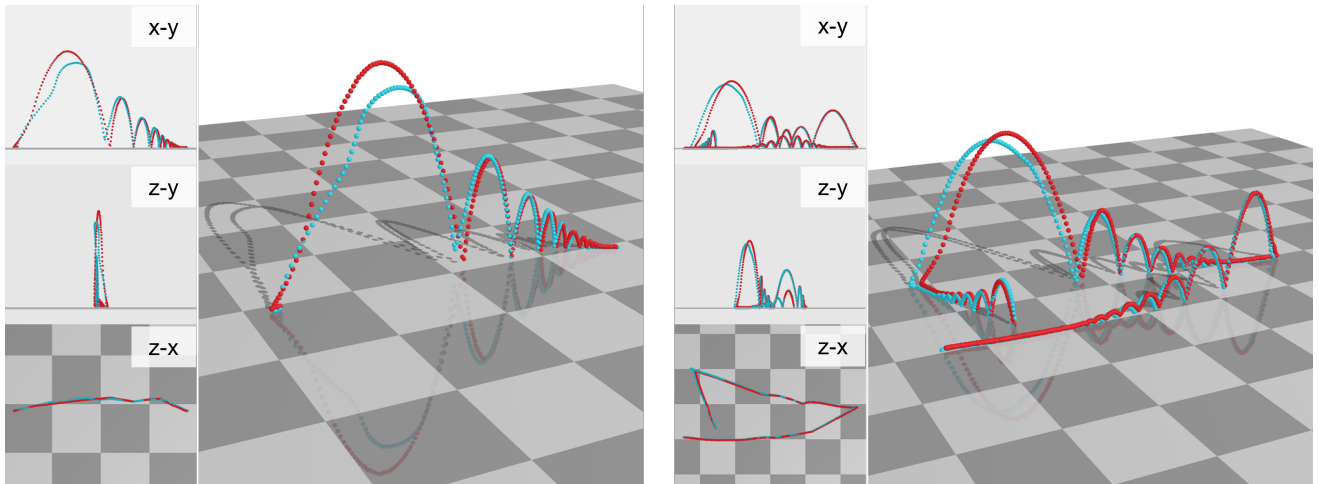


Figure 17. Failure cases on Real Mocap dataset. Blue: our predictions. Red: ground truth. Each checkerboard block is $75 \times 75 \text{ cm}^2$.

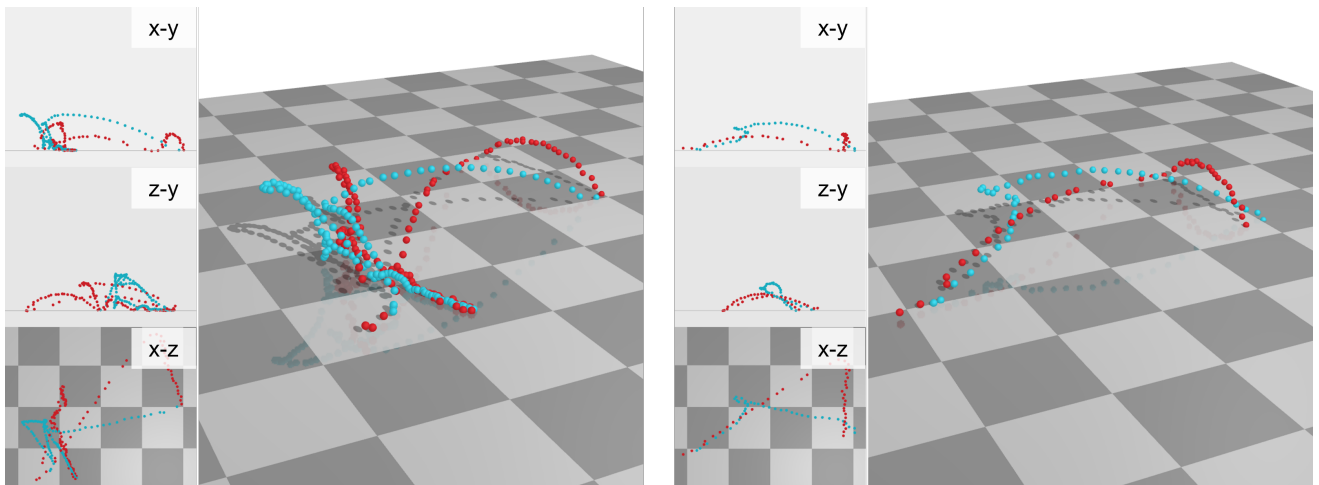


Figure 18. Failure cases on Real IPL(soccer) dataset. When a soccer player chests the ball, the trajectory may look very different from the training trajectories, leading to these errors. Blue: our predictions. Red: ground truth. Each checkerboard block is $250 \times 250 \text{ cm}^2$.

Table 18. How helpful is simulated data? We report NRMSEs \pm S.E. of training on Real and Synthetic Mocap, as well as on Synthetic then fine-tuning on Real. Using Synthetic for training or pre-training outperforms training on Real alone.

Training data	Distance	Height
Real Mocap	0.29 ± 0.04	7.99 ± 1.68
Synthetic Mocap	0.17 ± 0.01	7.12 ± 1.14
Synthetic + Real (Fine-tuned)	0.08 ± 0.004	5.23 ± 1.06