SportsNGEN: Sustained Generation of Realistic Multi-Player Sports Gameplay

Lachlan Thorpe[1], Lewis Bawden[1], Karanjot Vendal[1], John Bronskill[2] and Richard E. Turner[2]

[1]*Hawk-Eye Innovations Ltd., U.K.*
[2]*University of Cambridge, U.K.*

Abstract: We present a transformer decoder based sports simulation engine, *SportsNGEN*, trained on sports player and ball tracking sequences, that is capable of generating sustained gameplay and accurately mimicking the decision making of real players. By training on a large database of professional tennis tracking data, we demonstrate that simulations produced by *SportsNGEN* can be used to predict the outcomes of rallies, determine the best shot choices at any point, and evaluate counterfactual or *what if* scenarios to inform coaching decisions and elevate broadcast coverage. By combining the generated simulations with a shot classifier and logic to start and end rallies, the system is capable of simulating an entire tennis match. We evaluate *SportsNGEN* by comparing statistics of the simulations with those of real matches between the same players. We show that the model output sampling parameters are crucial to simulation realism and that *SportsNGEN* is probabilistically well-calibrated to real data. In addition, a generic version of *SportsNGEN* can be customized to a specific player by fine-tuning on the subset of match data that includes that player. Finally, we show qualitative results indicating the same approach works for football.

1 INTRODUCTION

The application of machine learning methods has proven beneficial to many sports applications (Zhao et al., 2023). In particular, sports simulation and analysis can provide valuable insights to sports teams when attempting to understand how small changes to player formation or playing style could impact the next period of play, or their chances of winning (Hauri and Vucetic, 2022; Teranishi et al., 2022; Wang et al., 2023). In addition, realistic gameplay simulation is critical in computer gaming scenarios (Kurach et al., 2020).

Tremendous progress has been made in the area of sports trajectory prediction (Yue et al., 2014; Zheng et al., 2016; Le et al., 2017b; Zhan et al., 2019; Li et al., 2021; Tang et al., 2021; Wu et al., 2021; Alcorn and Nguyen, 2021; Omidshafiei et al., 2022), however it is difficult to precisely mimic training data over long periods of time. Figure 18 shows how the prediction error of the player and ball positions increases with time when simulated tennis data from our system is compared to the training data. Sports are inherently unpredictable over time scales longer than several seconds and as a result, deterministic prediction is not possible or useful in many scenarios. Instead, it is important to capture the different ways a match will evolve in a statistically accurate manner by modelling the complete distribution of player decision making.

We propose that generated sports simulations should be: (i) highly *realistic* and capture the complete distribution of real player behaviour; (ii) *sustained* for the duration between natural breaks in the gameplay; (iii) *customizable* via fine-tuning or other method to emulate the style of play of a particular player and/or team; and (iv) *measurable* in that metrics are available to evaluate the quality of the simulations (as opposed relying on a human expert) such that the simulations can be improved by optimizing the metrics.

Recently, the transformer architecture (Vaswani et al., 2017) has been applied to multi-agent spatiotemporal systems problems to generate realistic sports simulations and understand player behavioural patterns (Alcorn and Nguyen, 2021). Instead of generating words as in natural language processing, temporal player and ball movements can be generated by training a transformer model to predict the next position from a sequence of tracking data.

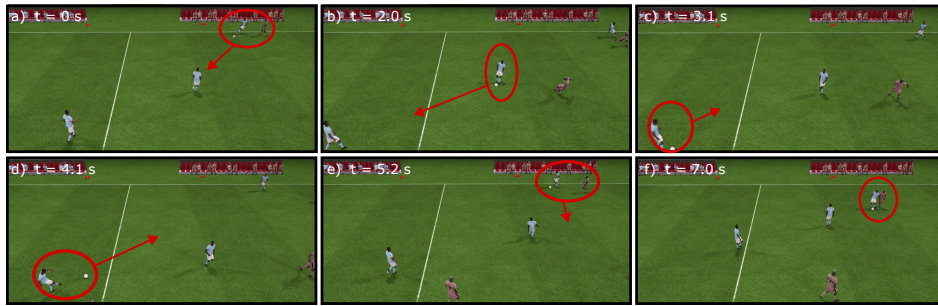However, to the best of our knowledge, no pre-

Figure 1: Frames from a football match simulated using *SportsNGEN*. The panels depict a passing sequence involving 3 players. The ball is in the red circle, with an arrow depicting the play that follows. Link to video: https://youtu.be/M0kkKiGVNzk.
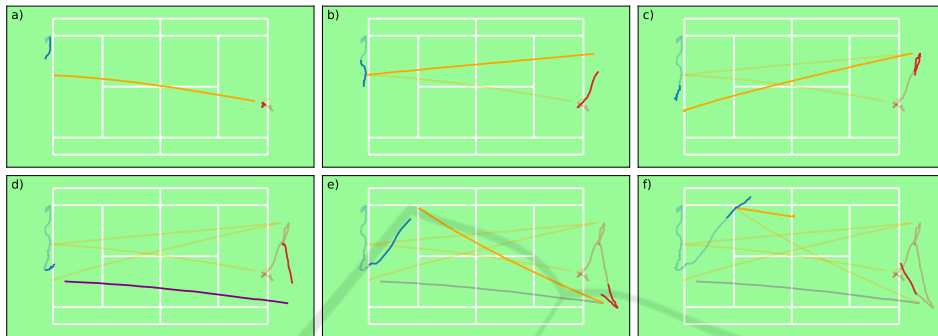

Figure 2: Simulated tennis rally between 2 players using 3 shots of training data as input. Frames a) - c): Training data shots. Frames d) - f) Simulated rollout. Red and blue markings indicate player movement. The lines indicate shot trajectories. The current shot is opaque while earlier shots are more transparent. The purple line is the first simulated shot. Link to video: https://youtu.be/A1_vv12V5q0.

vious work has been successful in generating realistic, sustained, and customizable simulations, learned from player and ball tracking data, for more than a few seconds. In this work we present Sports Neural Generator or *SportsNGEN* that realizes the goals of realistic, sustained, customizable and measurable sports gameplay. Figure 1 and Figure 2 depict football[1] and tennis sequences, respectively, generated by our approach along with links to simulation videos.

Our contributions: (i) A transformer decoder based simulation engine, *SportsNGEN*, trained on player and ball tracking data as well as match metadata, capable of simulating the movement of all players and the ball simultaneously in a sports game scenario. The simulations are sustained between breaks in play. (ii) Training and evaluating *SportsNGEN* on a large database of professional tennis tracking data. *SportsNGEN* is capable of simulating an entire tennis match by combining the generated simulations with a shot classifier and logic to start and end rallies. (iii) We show that our model can be used to inform tennis coaching decisions and best shot options by evaluating counterfactual or *what if* options.

(iv) We demonstrate through ablations that the following enhancements significantly improve convergence and generated simulations: a) extending the player and ball representations to include relative velocity, distance to the ball, and time into the game or sequence; b) adding small perturbations to the ball positions during training to allow the model to correct for errors; and c) Adding context tokens to allow the model to adapt to different playing surfaces. (v) We devise a novel optimization method by defining metrics to statistically evaluate the quality of generated tennis data. By altering simulation hyperparameters, we show that the simulations can be optimized to be statistically similar to the behaviour of real players. (vi) We demonstrate that a generic version of our model can be customized to a specific tennis player by fine-tuning on match data that includes that player.

2 RELATED WORK

In this section, we discuss related work in the categories of sports analytics, and game simulation, and trajectory prediction.

---

[1]We use the term football to refer European football or soccer.

Group Activity Recognition and Sports Analytics. Miller et al. (2014) develop an approach to represent and analyze the underlying spatial structure that governs shot selection among professional basketball players. Le et al. (2017a) employ an imitation learning approach to analyze football defensive strategies. Hauri and Vucetic (2022) propose a transformer-based architecture with a Long Short-Term Memory (LSTM) embedding to recognize basketball group activities from player and ball tracking data. Teranishi et al. (2022) evaluate football players who create off-ball scoring opportunities by comparing actual movements with the reference movements generated via trajectory prediction. Chen et al. (2023) use a probabilistic diffusion approach to model basketball player behavior. The model only considers player movement and no other metadata. Wang et al. (2023) present a football tactics assistant that focuses on analyzing corner kicks which allows coaches to explore player setup options and use those with the highest likelihood of success.

Game Simulation. Kurach et al. (2020) introduce a game engine that simulates football gameplay with an environment for evaluating RL algorithms. Liu et al. (2021) demonstrate an RL approach, where the agents progressively learn to play football initially from random behavior, to simple ball chasing, to showing evidence of cooperation. Braga and Barros (2022) introduce a simulator for robot football optimized for performing RL experiments. Yu et al. (2023) introduce a RL environment where agents are trained to play basketball.

Finally, Yuan et al. (2023) describe a method to learn simulated tennis skills from broadcast videos. However, this approach only models one shot cycle at a time, using statistical analysis to predict the desired shot location for specific players. There is no coupling between the current shot and previous shots, so strategic play is limited. Also, the players are only allowed to move on the baseline, so no volleys or inner court play is permitted, restricting realism.

Our approach is not RL based. It instead learns in a discriminative fashion from sequences of gameplay tracking data, which obviates the need to use physics based models of gameplay or learning gameplay from scratch with RL. This also enables us to build predictive models for specific players which can be important for analysis and gaming scenarios. Overall, our work is distinct from the above works in that our goal is to generate sports gameplay that captures the complete distribution of player behaviour, where the aggregation of non-deterministic simulations is statistically similar to real data. This acts as a powerful tool for strategic analysis, evaluating how player decision making affects the outcome of a period of play.

Sports Trajectory Prediction. There is a rich literature on trajectory prediction in general, and sports trajectory prediction in particular. Yue et al. (2014) learn predictive models for basketball play prediction given the current game state. Zheng et al. (2016) model spatiotemporal trajectories over long time horizons using expert demonstrations capable of generating realistic, but short rollouts. Le et al. (2017b) present an LSTM based imitation learning approach for learning multiple policies for team defense in professional football. However, no policy is learned for the position of the ball. Zhan et al. (2019) describe a hierarchical framework for sequential generative modeling that can generate high quality trajectories and encode coordination between agents. However, their framework cannot generate entire games. Li et al. (2021) describes an approach for multi-agent trajectory prediction using a graph neural network. When evaluated on basketball data, only short trajectories were considered. Tang et al. (2021) propose the concept of collaborative uncertainty, to model the uncertainty in interaction in multi-agent trajectory forecasting. Wu et al. (2021) propose a generative adversarial network (GAN) to generate short basketball player and ball trajectories. Alcorn and Nguyen (2021) introduce `baller2vec`, a multi-entity transformer that can model coordinated agents. It employs a special self-attention mask to learn the distributions of statistically dependent agent trajectories and is shown to generate realistic trajectories for basketball players or the ball itself for short durations. Our work builds upon `baller2vec` to enable sustained gameplay simulations by simultaneously simulating both the player and the ball. Omidshafiei et al. (2022) study the problem of multiagent time-series imputation in the context of football in order to predict the behaviors of off-screen players.

3 METHODOLOGY

In this section we provide a complete description of our approach to generating sports simulations. A flow diagram of *SportsNGEN* is shown in Figure 3.

3.1 Input Data

We index the N players and the ball in a match with $n \in \{1, \ldots, N, \texttt{ball}\}$. We then define an *object token* $O_{\tau,n}$ at index τ to represent the state of nth player or
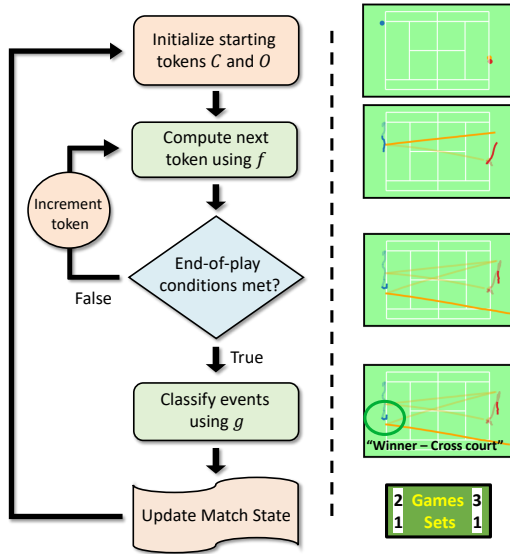
Figure 3: Left: *SportsNGEN* flow diagram. Right: Cartoons from a simulated tennis match corresponding to the flow chart steps.
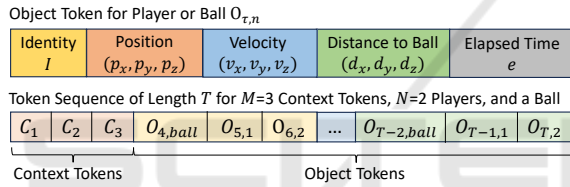

Figure 4: Top: Layout of an object token $O_{\tau,n}$. Bottom: Sequence of T tokens for M=3 context tokens, N=2 players, and a ball.

ball as:

$$O_{\tau,n} = \{I_n, (p_{x,\tau,n}, p_{y,\tau,n}, p_{z,\tau,n}), (v_{x,\tau,n}, v_{y,\tau,n}, v_{z,\tau,n}),$$
$$(d_{x,\tau,n}, d_{y,\tau,n}, d_{z,\tau,n}), e\}$$

where p denotes position, v velocity, d distance to the ball, $I \in \mathbb{R}^1$ a learned identity for a player that can capture their style of play, $e \in \mathbb{R}$ elapsed time into the game or sequence depending on the sport, and $x, y, z \in \mathbb{R}^3$ are components in a 3D coordinate system. The position data are typically supplied as the center of mass (COM) of the ball or player from a sports tracking system. For all players, position is 2D only i.e. $p_{z,\tau,n} = v_{z,\tau,n} = 0$ and for the ball, distance d is set to 0. The e component of the feature vector is useful to model long-term dependencies due to player fatigue and team strategy or for ensuring simulated tennis rallies are realistic in length. We normalize the p, v, and d components of O by appropriate values for each sport.

As a crucial step in generating sustained simulations, we add a small amount of uniform noise to the position p and velocity v of the ball. We find that training on noise-free ball trajectories does not lead

to stable simulations as any errors in the prediction lead to out-of-distribution inputs at the next time step, which the model cannot correct.

In addition to the object tokens, we also define a set of *context tokens* $\{C_1, \ldots, C_M\}$ specific to each sport that contain information that would influence gameplay such as the score, the identity of the opposing team, the location of the game, and the weather. We convert each piece of contextual information into feature vectors, either through learned encodings for discrete information such as the stadium, or training a network to convert a representation of the score into a feature vector. Figure 4 depicts the components of a token and the order of tokens in a training sequence.

Cropping Sequences. We crop the input training sequences to eliminate data outside of actual gameplay. The data removed includes players getting into position for the next play or switching sides which are not essential for simulation. To train the model efficiently using batches, we define a maximum sequence length of tokens T and cut any sequences longer than this into multiple sequences. Shorter sequences are padded to make up the remainder of the maximum length. The sequence length T depends on the sample rate of the data, and the length of previous data relevant to predicting the next time step. Tracking data can be sampled up to 50 Hz. Although this provides extremely fine detail, for team sports like football with 23 objects on the pitch, a period of 5 seconds at 50 Hz would produce a sequence length of 5750 tokens, making the model impractical to train. Since many of the dynamics in matches are longer than 5 seconds, we make a compromise between sample rate and computational cost.

3.2 Transformer Decoder Model

We use a transformer decoder model f that is an extended implementation of `baller2vec` (Alcorn and Nguyen, 2021) to predict future player and ball states given the current and recent history of states. We make a significant update to the `baller2vec` experiments by modelling both the ball and the players simultaneously.

The model f is run in an auto-regressive mode with a rolling window of length T, using a specified period of previous predictions to predict the ball and player state at the next step. We use the same attention method as `baller2vec`, permitting each object token to attend to every object token up to and including its own time step. We adjust the attention mask so that each object token can attend to the context tokens, influencing the predictions for player and ball

movement.

We treat the update step as a classification as opposed to a regression or diffusion problem, by splitting the area of possible next locations for the ball and players into a 3D and 2D grid, respectively, of discrete bins that indicate the relative offset ρ from the current position p as this is easier to learn and can bound motion to physically possible values. A depiction of a grid for a football player and the ball is shown in Figure 5.

Figure 5: Visualization of the 2D and 3D classification grids used to predict the position of a player and the ball at the next time step.

We use nucleus sampling (Holtzman et al., 2020) to sample the location in the output grid based on the output probabilities of f. When the grid location has been selected, we turn the discrete value into a continuous value by sampling from a uniform distribution across the bin. If the initial conditions for the player or ball have zero velocity, this helps to force the simulation into motion by avoiding continuous velocity predictions of zero.

To enable the model to learn the behavior of individual players, the bin size must be fine grained enough for predictions to capture distinguishing features. In many sports, important statistics include how fast a player can run, or how far they can hit, throw or kick the ball. Formally, the probability distribution of predicting a particular bin location k for an object n at step $\tau + 1$ is

$$p(\rho_{\tau+1,n} = k | O_{1:\tau,n}) = f(O_{1:\tau,n}, k).$$

The value of ρ is then sampled from the distribution:

$$\rho_{\tau+1,n} \sim p(\rho_{\tau+1,n} = k | O_{1:\tau,n}).$$

Based on the sampled value of ρ and the mapping between bins and physical distance, the updated values of position $p_{\tau+1,n}$, velocity $v_{\tau+1,n}$, and distance to the ball $d_{\tau+1,n}$ can be computed. Since we use the `baller2vec` attention mask, the positions of the ball and each player can be updated simultaneously at each time step.

We detect the end of a simulation or break in a play with logic specific to each sport. For example, we can end simulations if a ball goes out of bounds

or in some sports if the ball makes contact with the ground, or if the time in the period of play runs out. When generating simulations, we set a maximum input sequence length of T tokens. For a player and ball state update at step $\tau + 1$, we input from $\tau - T$ to τ steps of initial token data into the model f. If T time steps of data are not yet simulated, the missing tokens are padded with zeros and masked. Specifically, simulations are rolled autoregressively out at the ith step as

$$\rho_i \sim p(\rho_i = k | O_{i-1}, O_{i-2} \ldots O_{i-T}).$$

3.3 Event Classification

We also train an event classifier g which is run after a break in gameplay. Examples of events would be passes, runs, fouls, goals, the type of shot played, and so on. The event classifier g has the same input and architecture as f, but does not use attention masking, and uses separate prediction heads for each different type of event. The event classifier can be used for defining the initial conditions for the next play and gathering statistics about the period of play.

4 TENNIS IMPLEMENTATION DETAILS

In this section, we detail the implementation of *SportsNGEN* for tennis. Initial rally conditions, boundary logic and relevant player statistics are well defined, so we can demonstrate the capabilities of the system.

We use a proprietary dataset of tennis tracking data for approximately 15,000 tennis matches containing 7.6 million rally sequences. The data contain COM locations for each player and the ball sampled at 25 Hz, with the center of the court at $(x, y, z) = (0, 0, 0)$, whose components refer to the length, width, and vertical directions, respectively.

The data also contains metadata about each match and rally, including: the players in the rally, the tournament and court, the rally winner, whether the rally was a first or second serve, and what shots were played. The tracking data set is cut up into individual sequences that start at the toss before a serve and end shortly after the rally is finished. We set a maximum sequence length of input data to be 6 seconds. We found that increasing the sequence length to be more than 6 seconds became computational impractical and did not improve the model accuracy. This suggests that professional tennis players' decision making is not strongly affected by information further than 6 seconds into the past. We also double the size of the

data set by flipping the data along the x and y axes simultaneously.

We allow for ± 25 mm of uniform position and per unit time velocity noise in the x dimension and ± 12.5 mm of noise in the y and z dimensions. If the added noise is any smaller than this, the simulations start to break down. For output classification, we use 61 bins for each dimension, scaled for the ball such that the maximum velocity is fractionally faster than the current fastest serve speed. This results in $61 \times 61 = 3721$ and $61 \times 61 \times 61 = 226981$ possible bin locations for the player and ball output, respectively. At 25 Hz, this equates to a ball bin size of $\{x, y, z\} = \{46, 13, 10\}$ mm.

The playing surface is important contextual information when predicting rallies in tennis. The expectation is that hard and grass courts have the fastest bounces, and clay courts absorb more momentum from the impact resulting in slightly slower bounces and longer rallies. We learn context vectors for each surface and tournament in the dataset, and also encourage the model to learn the difference between first and second serve types by including context vectors for both.

We generate initial conditions based on historical examples from the data when particular players are serving first or second serves from specific sides. We take the initial condition as the start of the toss movement during the serve. This initial condition includes the positions and velocities in all dimensions for both players and the ball. We can detect the end of the rally through simple logic on the movement of the ball. If the ball continues past a player, is close to stationary near the net, bounces out of bounds or bounces twice on one side of the net, then we can deem the rally to have finished. At this point, we stop the simulation and collect the rally data using the event classifier.

To understand who won the rally, and for analysis of the point, we train the event classifier to classify the type of shot being played at every step within the simulation. This includes the type of stroke (groundstroke, serve, volley, etc.), the direction of the shot (cross court, down the line, etc.), whether the shot is a winner, error, or a continuation of the point, and if an error is forced or unforced.

The event classifier g receives as input a simplified version of the input token, without any identity I or context C components. In the training data, shot type labels are consistent across time steps between shots. The model is expected to predict the same, only varying its prediction when the ball contacts a racket. When a rally is finished, we convert the tracking data from the rally into the shot type classifier input, run the model once to identify where the changes

in shot type are, and take the model shot type between changes as the final label for each shot. The winner or error classification for the final shot of the rally tells us who won the point, and the shot type labels help us break down the shots for statistical analysis. To combine rallies together to simulate an entire match, all that is left to do is implement logic to increment the score, calculate who is serving, from which end and which side. These can be used to obtain the initial conditions for the next point.

Tennis Network Architecture. The input tokens $O_{\tau,n}$ are embedded with a 3 layer MLP with input size 30, hidden sizes 256 and 512, and output size 2048. The transformer decoder, f, has 4 layers, 2048 embedding dimension, 8 heads, 4 expansion factor, and 0.2 dropout. The shared player output network is a single linear layer with input size 2048 output size equal to 61×61 bins. The ball output network is a single linear layer with input size 2048 and output size equal to $61 \times 61 \times 61$ bins.

5 EXPERIMENTS

For the tennis experiments, we selected 3 male professional players with varying styles to evaluate *SportsNGEN* and simulated 6 matches between each combination of two players. Each match was the best of 3 sets. We repeat this experiment across 3 different tournaments, one for each surface type: hard, clay and grass. For comparison, we then collect data where these players have played each other on these surfaces. Using both real and simulated data, we compute relevant statistics and define an evaluation metric for each statistic as the difference between the two.

For physical metrics, we compare the median, inter-quartile range (IQR), and Wasserstein distance between the distributions of real and simulated data for the following quantities collected across all matches:

- **Toss contact height:** Height of the ball at the contact point with the racket during serving.

- **First and second serve speeds:** Maximum recorded speed during the serve.

- **Return speeds:** Maximum speed of a return of serve.

- **Groundstroke speeds:** Maximum speed of all groundstrokes.

We also compute additional relevant statistics based on aggregated data. For these quantities, a scalar value is aggregated over many rallies for each player.

The absolute difference between the real and simulated aggregated scalars is compared.

- **First serve %:** Percentage of first serves that are in bounds.
- **Double fault %:** Percentage of second serves that are out of bounds.
- **First and second serve win %:** Percentage of rallies won when serving on first and second serve, respectively.
- **Ace %:** Percentage of first serves that are aces.
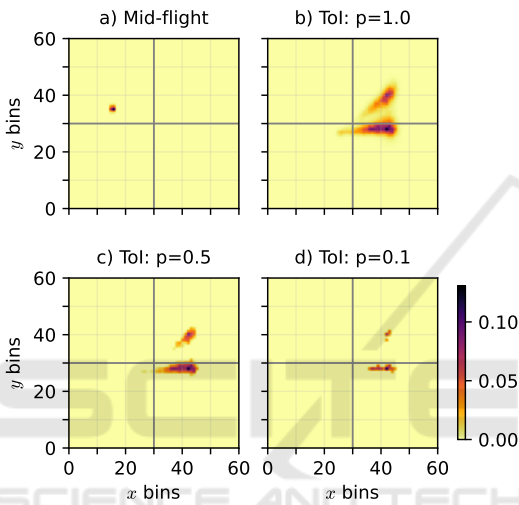- **Serve points won %:** Percentage of rallies won as server.

Figure 6: Bin probabilities for the ball projected into the xy plane during a) mid-flight and b) - d) at time of impact (ToI) for 3 values of top-p. The center of each diagram, bin $(x, y) = (30, 30)$, corresponds to no movement. Yellow indicates a probability of 0 while progressively darker colors indicate higher probabilities.

Varying the Top-p Sampling Parameter. Figure 6 shows typical output probability distributions, projected into the xy plane, for an update step of the ball in mid-flight, and at the moment the ball is about to be hit. The peaks in intensity for the mid-flight predictions (a) are distributed over very few bins since the model has learned the physical constraints of the system (e.g. drag, gravity), and can therefore be very confident in how to update the ball state. The remaining panels (b)-(d) depict the probability distributions for the ball at the time of impact (ToI) – the point at which a player hits the ball for various values of top-p. The distributions in these cases contain multiple separated peaks in intensity in the xy plane. This corresponds to different decisions a player may make when choosing to play the shot either down the line or across the court. By sampling from these two modes,

we are able to perform counterfactual analysis (see Section 5). As top-p decreases, the probability of sampling a cross court shot decreases, demonstrating the need to optimize top-p in order to accurately capture the player behaviour in the simulations. In general, we will see that a low top-p value will result in less variety in playing style, but a high top-p value will result in many outliers.

We see the similar patterns in Figure 7a), the cumulative probability for the player and ball at ToI for a return and during mid-flight for a shot as a percentage of number of contributing bins. For mid-flight predictions of the ball, the probability distribution is concentrated over few bins, with 90% of the distribution contained within 0.002% of the total bins. When predicting changes in direction (e.g. at ToI), the probability distribution is spread over more bins, up to 0.5% of the bins are required to populate 90% of the cumulative probability.

Figure 7b) and (c) show how the various metrics vary with top-p. In (b), the number of non-realistic rallies (rallies that must be discarded based on logical checks) increases with a value of top-p both that is too high, and too low. Increasing top-p increases the probability that the ball trajectory may be updated in a way that defies physical constraints and would be forced to be removed. With too low top-p there may be too few options for the ball and player to update in a way that leads to a realistic rally. In (c), we see that with the exception of double fault percentage, the metrics reach optimal values when top-p is in the range of 0.8 to 0.9.

Object Token Component Ablation Study. In Figure 8, we quantify how the additional components in the token vector O affect the convergence and final accuracy of the physical metrics when compared to a baseline model that does not use velocity v, distance to the ball d, elapsed time e, or context tokens C (similar to that used in `baller2vec`). The plots show that *SportsNGEN* converges faster and reaches better results than the baseline model when averaged across all physical metrics. We also see faster convergence to ~20% non-realistic rallies.

Varying the size ι of the player encoding vector I in Figure 9, we find that the accuracy increases until $\iota = 20$ where there are diminishing returns for further increases. This is also supported by Figure 10 d)-f), where the data with no player ID I has a much broader distribution of serve speeds, and a nearly identical median serve speed for all three players.

Context Token Study. We add context tokens to encode the tournament, court surface type, and
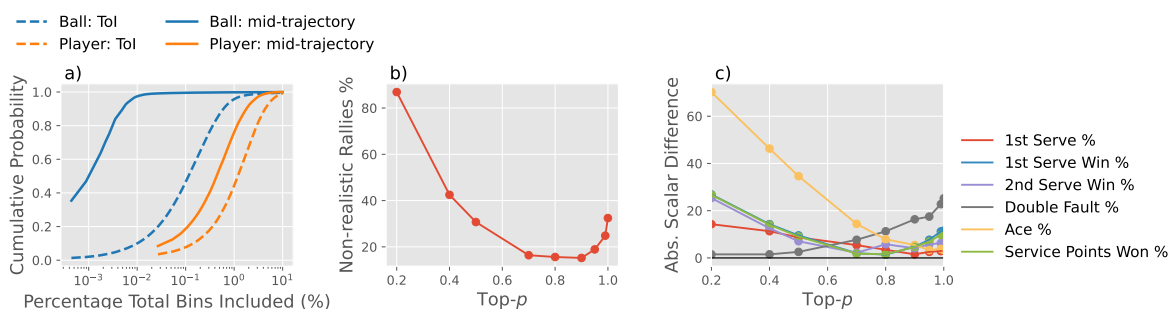
Figure 7: A) Cumulative probability for the player and ball at ToI for a return and during mid-flight for a shot as a percentage of number of contributing bins. b) Proportion of non-realistic rallies that are discarded during match simulation. c) Absolute difference between aggregated statistics in the training data and the simulations as a function of top-p.
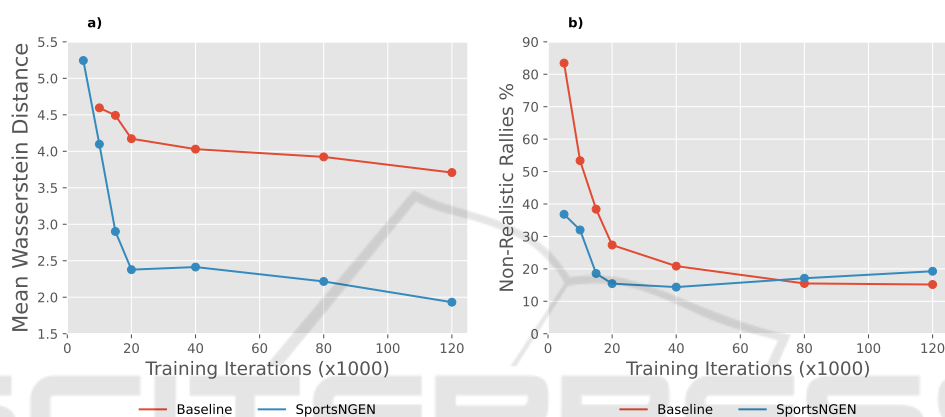

Figure 8: A comparison of convergence for *SportsNGEN* against a Baseline model without v, d, e and context tokens C, for a) An average of the 4 physical metrics, and b) Non-realistic rallies as a function of training iterations.

Figure 9: Varying the player ID I size ι to show how various metrics can be improved with a larger ι. As a control, we train a generic model without player ID I.

whether the serve is the player's first or second. Typically the second serve is expected to be slower since players will prioritize accuracy over speed to avoid losing a point through double fault. Figure 10 shows that the addition of a serve context token C_{serve} as well as the player ID component I in O reduce the difference between real and simulated serve speeds and

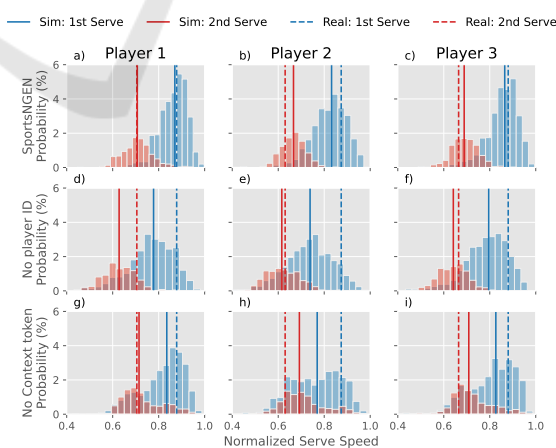produce narrower distributions between first and second serve speeds.

Figure 10: The distribution of first and second serve speeds for all three players for the following models: (top) *SportsNGEN*, (middle) a model with no player ID vector I, (bottom) a model with no serve context token C_{serve}.

To quantify the effect of the playing surface, we use the coefficient of restitution by taking the ratio of
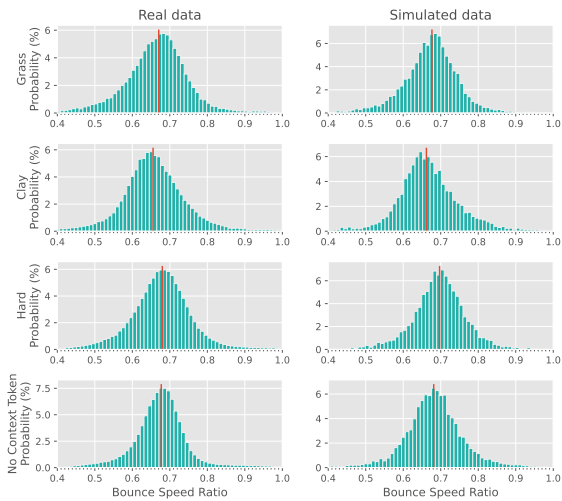
Figure 11: Ratio of speed after to before the ball bounce. Each row contains results for a different court surface type. The columns are real (left), and simulated (right) data. The last row is surface type agnostic, containing a weighted average of the data for each court.

the speed after to before the bounce. A value less than 1 means the ball has lost momentum and indicates a slower surface. Figure 11 shows this metric for three court types and for the surface agnostic case, for both real and simulated data. The median value for each court type follows the expected trend: typically clay courts have the slowest bounces, and hard courts have the fastest, which is better represented when we introduce the surface token into the model.

We also demonstrate that *SportsNGEN* is realistic throughout the rally with Figure 12 showing the distribution of rally lengths for real data, and simulations from *SportsNGEN*. Although we see a slightly higher peak in rally lengths in (b), we see both distributions with a peak at a small number of shots per rally, and tailing off towards 15 shots.
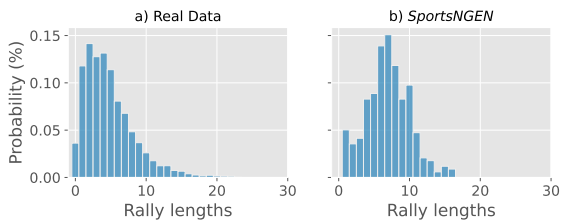

Figure 12: Length of rallies in number of shots for a) the original training data for the given three players on hard surfaces, b) simulated data using *SportsNGEN*.

Transfer Learning. As an extension to training a model capable of capturing the behavior of all players, we also train a generic model f_{gen} which learns a single feature vector I_{gen}, called the *generic player* vector where $I_n = I_{gen}, n \in N$. We then fine-tune f_{gen}

with matches containing a specific player, and transfer learn a new set of $I_n \in N$ for the player that can represent their behavior against a generic opponent. This could be used for quickly customizing a pretrained model to a new player on the circuit.

Figure 13 shows various metrics as a function of the number of training sequences that are required to fine-tune f_{gen} such that the generic player ID vector I is adapted to a new player. In the simulations, f_{gen} is the opponent for the fine-tuned model. The groundstroke and return metrics improve as the number of training samples increases whereas the serve metrics fluctuate with the first serve speed getting worse. This can be explained by the low variability of the serve distribution being easier to learn when compared to highly variable groundstroke patterns.
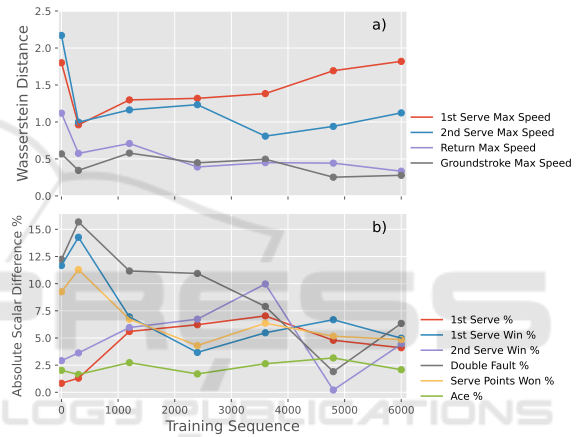

Figure 13: Learning features of a specific player by fine-tuning a generic model, showing a) the Wasserstein distance for physical data, and b) difference to training data for statistical metrics.

6 APPLICATIONS

In this section we explore *SportsNGEN* applications.

Predicting Rally Outcomes. A key intended application of *SportsNGEN* is generating insights for coaching and sports broadcasts. To prove it's validity for these applications, the model should accurately forecast the probability that each player wins a rally as it develops.

We can test *SportsNGEN*'s ability to do this in the following way. We sample random rallies from the training data, and roll out the model from a given random time step 100 times, to generate a win percentage for both players. Repeating this for a large number of starting points, we form a histogram of predictions by stratifying the predictions into bins.

Figure 14 shows the histogram of events contributing to the win percentage calibration plot. For each event, 100 simulated rollouts are used to generate the win percentage. The mean win percentage generated by the model is close to 50% which is to be expected for tennis rallies. In addition there are situations in which the winner is very likely already determined (if the random time chosen is close to the end of the rally, for example). As a result, the bins close to 0 and 100% are also more populated which explains the higher error in the more sparsely populated bins close to 20% and 80%.

For each prediction, we also have the ground truth of who won the rally in the training data. Taking the 90% bin for example, if the model is well-calibrated, the corresponding ground truth rallies should be won by the player in 90% of cases. Figure 15 shows that the win percentages generated by the *SportsNGEN* are well-calibrated, with deviations where data are sparse.
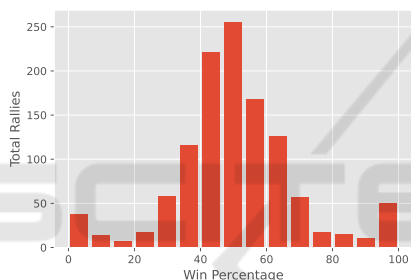

Figure 14: Histogram of win percentages output by the model when simulating rollouts in a random rally at a random point.
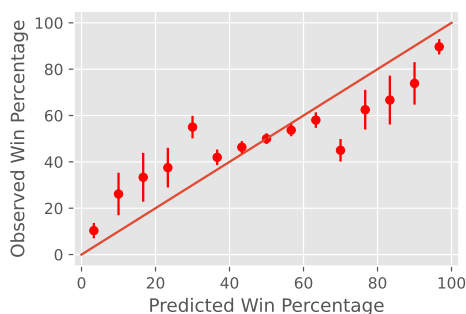

Figure 15: Predicted win percentages vs. observed win percentages for *SportsNGEN*. The solid line shows ideal calibration. The win percentages output by *SportsNGEN* are well calibrated.

Counterfactuals. Figure 16 demonstrates one way the *SportsNGEN* can be used to inform coaching decisions. A point indicated by the red dot in a real rally is chosen as a branch point in time, just as a player is about to play a shot. In the real rally, the shot af-

ter the branch point goes straight down the middle – indicated by the purple line in (a).

We can force alternative shot selection by sampling from the cross court mode in Figure 6 and analysing how the rally would have played out. In (b) and (c), two alternatives are depicted. We quantify the strength of each shot selection by running 100 simulations until the end of the rally, sampling equally from both modes at this branch point. We then aggregate statistics to calculate a win percentage for each shot choice.

Playing a shot to either of the two corners gave the player roughly equal probability of winning at 58%, whereas the original choice of hitting to the middle reduced the probability below 50%. Pushing the opponent farther to the edge of the court may explain this advantage. Figure 6 shows that the probability of the cross court mode is lower, highlighting that the player would more often opt for the safer shot down the middle.

Figure 17 shows the results of many simulations forcing a certain type of shot for the shot shown in purple. It shows that even if there are constraints imposed on the type of shot, there can still be variability in play. Running this simulation for many shots and aggregating win percentages can give insight into the kinds of tactics that would be advantageous, and since the player and court can be specified and trained on real data, it could be specifically useful for improving the play style of a player in a particular situation.

Football. Though we focused this work on tennis, we have had success using *SportsNGEN* to simulate football matches with a high degree of realism using the same model architecture. Click on https://youtu.be/M0kkKiGVNzk for a video demonstration of sustained passing sequences. The player and ball positions are derived from COM data.

7 LIMITATIONS

An important limitation of SportsNGEN is that it is not designed to handle out of distribution situations. Unconventional initial conditions can produce unreliable results. This extends to unseen players in which the model will default to a "generic" player representation. This method is also computationally intensive, requiring 2 days to train on an NVIDIA A100 GPU. While we believe our method is applicable to many sports, we have only trained models for football and tennis. Other sports may introduce difficulties.
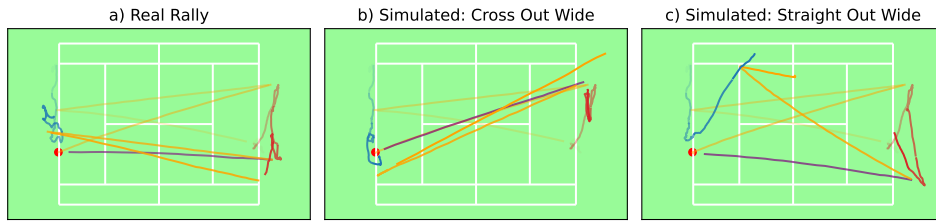
Figure 16: A real rally, and two simulated rallies for a different shot type, where the color transparency indicates time into the rally (with opaque being the end). The ball trajectory is orange, with the shot at which the simulations start shown in purple. The point at which the two simulations are branched is denoted by a red dot. The players are shown as blue and red traces.
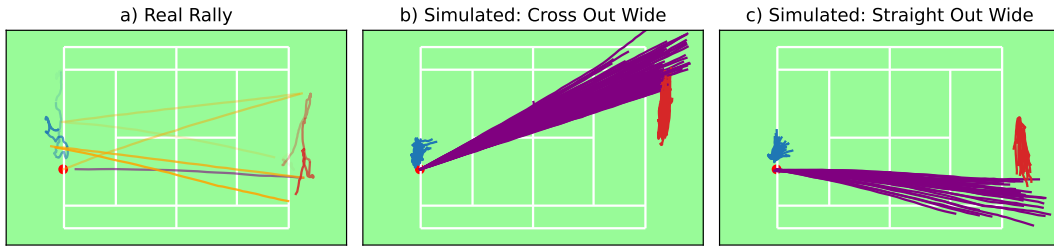

Figure 17: A real rally a), and many simulated rallies for two different shot types b), c). In the real rally, the increasing color opacity indicates time into the rally. The ball trajectory is orange, with the shot at which the simulations start shown in purple, the point at which this is branched is denoted by a red dot. The players are shown as blue and red traces. In the simulations, only the shots after the decision are shown to highlight the possibilities arising from the simulation engine.

8 DISCUSSION

In this work, we detailed *SportsNGEN* that is capable of generating realistic sports gameplay when trained on player and ball tracking sequences. A unique aspect of the system is the ability to customize gameplay in the style of a particular player via fine-tuning. Also, it is straightforward to use *SportsNGEN* to inform coaching decisions and game strategy through counterfactuals. In the future, we plan to adapt *SportsNGEN* to sports beyond tennis and football.

ACKNOWLEDGEMENTS

REFERENCES

Alcorn, M. A. and Nguyen, A. (2021). baller2vec: A multi-entity transformer for multi-agent spatiotemporal modeling. *arXiv preprint arXiv:2102.03291*.

Braga, P. H. and Barros, E. S. (2022). rsoccer: A framework for studying reinforcement learning in small and very small size robot soccer. *RoboCup 2021: Robot World Cup XXIV*, 13132:165.

Chen, X., Wang, W.-Y., Hu, Z., Chou, C., Hoang, L., Jin, K., Liu, M., Brantingham, P. J., and Wang, W. (2023). Professional basketball player behavior synthesis via planning with diffusion. *arXiv preprint arXiv:2306.04090*.

Hauri, S. and Vucetic, S. (2022). Group activity recognition in basketball tracking data–neural embeddings in team sports (nets). *arXiv preprint arXiv:2209.00451*.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Kurach, K., Raichuk, A., Stańczyk, P., Zając, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O., et al. (2020). Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 4501–4510.

Le, H. M., Carr, P., Yue, Y., and Lucey, P. (2017a). Data-driven ghosting using deep imitation learning.

Le, H. M., Yue, Y., Carr, P., and Lucey, P. (2017b). Coordinated multi-agent imitation learning. In *International Conference on Machine Learning*, pages 1995–2003. PMLR.

Li, L., Yao, J., Wenliang, L., He, T., Xiao, T., Yan, J., Wipf, D., and Zhang, Z. (2021). Grin: Generative relation and intention network for multi-agent trajectory prediction. *Advances in Neural Information Processing Systems*, 34:27107–27118.

Liu, S., Lever, G., Wang, Z., Merel, J., Eslami, S., Hennes,

D., Czarnecki, W. M., Tassa, Y., Omidshafiei, S., Abdolmaleki, A., et al. (2021). From motor control to team play in simulated humanoid football. *arXiv preprint arXiv:2105.12196*.

Miller, A., Bornn, L., Adams, R., and Goldsberry, K. (2014). Factorized point process intensities: A spatial analysis of professional basketball. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 235–243, Bejing, China. PMLR.

Omidshafiei, S., Hennes, D., Garnelo, M., Wang, Z., Recasens, A., Tarassov, E., Yang, Y., Elie, R., Connor, J. T., Muller, P., et al. (2022). Multiagent offscreen behavior prediction in football. *Scientific reports*, 12(1):8638.

Tang, B., Zhong, Y., Neumann, U., Wang, G., Chen, S., and Zhang, Y. (2021). Collaborative uncertainty in multiagent trajectory forecasting. *Advances in Neural Information Processing Systems*, 34:6328–6340.

Teranishi, M., Tsutsui, K., Takeda, K., and Fujii, K. (2022). Evaluation of creating scoring opportunities for teammates in soccer via trajectory prediction. In *International Workshop on Machine Learning and Data Mining for Sports Analytics*, pages 53–73. Springer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, Z., Veličković, P., Hennes, D., Tomašev, N., Prince, L., Kaisers, M., Bachrach, Y., Elie, R., Wenliang, L. K., Piccinini, F., et al. (2023). Tacticai: an ai assistant for football tactics. *arXiv preprint arXiv:2310.10553*.

Wu, G., Zhao, S., Lin, J., and Silva, C. (2021). Basketball gan: Sportingly acceptable trajectory prediction.

Yu, C., Yang, X., Gao, J., Chen, J., Li, Y., Liu, J., Xiang, Y., Huang, R., Yang, H., Wu, Y., and Wang, Y. (2023). Asynchronous multi-agent reinforcement learning for efficient real-time multi-robot cooperative exploration. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, page 1107–1115, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Yuan, Y., Makoviychuk, V., Guo, Y., Fidler, S., Peng, X., and Fatahalian, K. (2023). Learning physically simulated tennis skills from broadcast videos. *ACM Trans. Graph*, 42(4).

Yue, Y., Lucey, P., Carr, P., Bialkowski, A., and Matthews, I. (2014). Learning fine-grained spatial models for dynamic sports play prediction. In *2014 IEEE international conference on data mining*, pages 670–679. IEEE.

Zhan, E., Zheng, S., Yue, Y., Sha, L., and Lucey, P. (2019). Generating multi-agent trajectories using programmatic weak supervision. In *International Conference on Learning Representations*.

Zhao, Z., Chai, W., Hao, S., Hu, W., Wang, G., Cao, S., Song, M., Hwang, J.-N., and Wang, G. (2023). A survey of deep learning in sports applications: Perception, comprehension, and decision. *arXiv preprint arXiv:2307.03353*.

Zheng, S., Yue, Y., and Hobbs, J. (2016). Generating long-term trajectories using deep hierarchical networks. *Advances in Neural Information Processing Systems*, 29.

APPENDIX

Prediction Error Versus Time. Figure 18 shows the results from 200 simulations initialized from a random point in a random rally. The simulations are evolved for 1.75 seconds and the RMSE is plotted compared with the ground truth data for the ball and players. The baseline is taken as a linear extrapolation of the velocity of the player and ball frozen at the time the simulation begins. Our simulation performs better than a linear extrapolation over a short time, indicating it has learned how to sensibly predict and update the state vectors as a function of time.
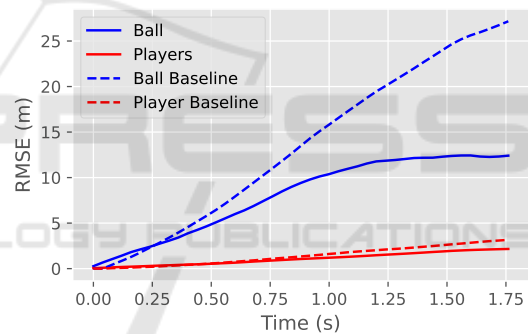

Figure 18: Root Mean Squared Error (RMSE) compared to real tennis data as a function of time, for both ball and player positions when simulating forward from a random in a rally. *SportsNGEN* performs better than a baseline of linear extrapolation.