

# Egocentric Basketball Motion Planning from a Single First-Person Image

Gedas Bertasius

University of Pennsylvania  
 gberta@seas.upenn.edu

Aaron Chan

University of Southern California  
 chanaaro@usc.edu

Jianbo Shi

University of Pennsylvania  
 jshi@seas.upenn.edu

## Abstract

We present a model that uses a single first-person image to generate an egocentric basketball motion sequence in the form of a 12D camera configuration trajectory, which encodes a player's 3D location and 3D head orientation throughout the sequence. To do this, we first introduce a future convolutional neural network (CNN) that predicts an initial sequence of 12D camera configurations, aiming to capture how real players move during a one-on-one basketball game. We also introduce a goal verifier network, which is trained to verify that a given camera configuration is consistent with the final goals of real one-on-one basketball players. Next, we propose an inverse synthesis procedure to synthesize a refined sequence of 12D camera configurations that (1) sufficiently matches the initial configurations predicted by the future CNN, while (2) maximizing the output of the goal verifier network. Finally, by following the trajectory resulting from the refined camera configuration sequence, we obtain the complete 12D motion sequence.

Our model generates realistic basketball motion sequences that capture the goals of real players, outperforming standard deep learning approaches such as recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and generative adversarial networks (GANs).

## 1. Introduction

Consider LeBron James, arguably the greatest basketball player in the world today. People often speculate about which of his traits contributes most to his success as a basketball player. Some may point to his extraordinary athleticism, while others may credit his polished basketball mechanics (i.e., his ability to accurately pass and shoot the ball). While both of these characteristics are certainly important, there seems to be a general consensus, among sports pundits and casual fans alike, that what makes LeBron truly exceptional is his ability to make the right decision at the right time in almost any basketball situation.

Now, imagine yourself inside the dynamic scene shown in Figure 1, where, as a basketball player, you need to make

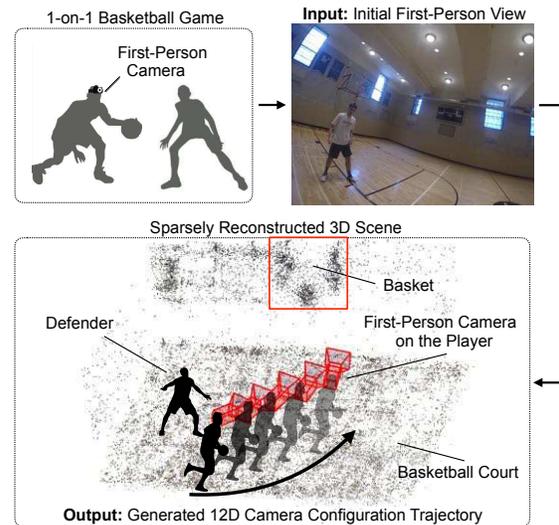


Figure 1: Given a single first-person image from a one-on-one basketball game, we aim to generate an egocentric basketball motion sequence in the form of a 12D first-person camera configuration trajectory, which encodes a player's 3D location and 3D head orientation. In this example, we visualize our generated motion sequence within a sparse 3D reconstruction of an indoor basketball court.

a series of moves to outmaneuver your defender and score a basket. In doing so, you must evaluate your position on the court, your defender's stance, your defender's court position relative to the basket, and many other factors, if you want to maximize your probability of scoring. For instance, if you observe that your defender is positioned close to you and leaning more heavily on *his* right leg, you might exploit this situation by taking a swift left jab-step followed by a hard right drive to the basket, throwing your defender off balance and earning you an easy layup. However, if your defender is standing farther away from you, you may decide to take a few dribbles into an area where your defender cannot get to you quickly, before stopping for a jump-shot. These are all complex decisions that have to be made within a split second – doing this consistently well is challenging.

In this paper, we aim to build a model that mimics this basketball decision-making process. Specifically, our model maps a first-person visual signal to a plausible egocentric basketball motion sequence. This is a difficult problem because little is known about how skilled players make subsecond-level decisions. What we do know is that players make decisions based on what they see: a player may look at how his or her defender is positioned (i.e., feet orientation, torso orientation, etc.) and then, based on that visual information, choose to drive right or left. Leveraging this insight, in this work, we learn our model from data recorded using first-person cameras. First-person cameras allow us to see various subtle details in the basketball game, just as the players do. In contrast, a standard third-person camera typically records a low-resolution view from a suboptimal orientation, making it difficult to observe such details. Because these subtle details may be crucial for predicting where a player will move next, first-person modeling is beneficial.

Our model takes a single first-person image as input and generates an egocentric basketball motion sequence in the form of a 12D first-person camera configurations, encoding a player’s 3D location and 3D head orientation throughout the sequence. To do this, we first introduce a future convolutional neural network (CNN) that predicts an initial sequence of 12D camera configurations, aiming to capture how real players move during a one-on-one basketball game. We also introduce a goal verifier network, which is trained to verify that a given camera configuration is consistent with the final goals of real players. Next, we use our proposed inverse synthesis procedure to synthesize a refined sequence of 12D camera configurations by optimizing the following objectives: (1) minimize the difference between the refined configurations and initial configurations predicted by future CNN while also (2) maximizing the goal verifier network output. Finally, by following the trajectory resulting from the refined camera configuration sequence, we obtain the complete 12D motion sequence.

Our egocentric basketball motion model learns to generate smooth and realistic sequences that capture the goals of real basketball players. Additionally, our model is learned in an unsupervised fashion; this is an important advantage, since obtaining labeled behavior data is costly. Finally, in our experimental section, we show that our method consistently outperforms standard deep learning approaches such as RNNs [16], LSTMs [14], and GANs [11].

## 2. Related Work

**Using Vision for Behavior Modeling.** Developing models that characterize human behavior in everyday tasks, such as walking or driving, has been a long-standing problem in computer science. The work in [32] uses a hidden Markov model (HMM) to learn human driving patterns. Recent work in [23, 34] uses third-person videos of

humans performing simple tasks, like opening a door, to teach robots how to do the same. By observing them from a third-person view, the method in [41] learns to remind humans of actions they forgot to do. Recently, there also has been a surge of methods designed to capture various aspects of social human behavior. For instance, the work in [4] uses a flow field model to track crowds. Also, the work in [31] develops a tracking method that can predict walking trajectories for multiple agents simultaneously, while more recent work in [25] predicts walking trajectories by using a game theoretic approach. Furthermore, the methods in [13, 28, 3, 2] develop social force or social affinity based methods for predicting pedestrian behaviors. In addition, the method in [20], uses Markov decision processes to predict the wide receiver trajectories in football, whereas the work in [19] learns to predict the behavior of soccer players. Finally, the work in [43] uses large amounts of “top-view” basketball tracking data [1] to train deep hierarchical networks for basketball trajectory prediction.

In contrast to these prior methods, our data are obtained via first-person cameras, which allows us to build a model that connects first-person visual sensation to egocentric motion planning ability. Using such first-person data is our main advantage compared to prior methods.

**First-Person Vision.** Most first-person methods have focused on first-person object detection [21, 8, 35, 10, 6, 7] or activity recognition [18, 38, 37, 33, 22, 29, 9]. Several methods have also employed first-person videos for video summarization [21, 24]. Furthermore, recent work in [40] introduced a first-person method for predicting the camera wearer’s engagement, while [5] developed a model to assess a basketball performance from a first-person video. Additionally, [36] proposed applying inverse reinforcement learning on first-person data to infer the goals of the camera wearer during daily tasks. Furthermore, the methods in [30, 39] propose to generate plausible walking trajectories from first-person images. However, the methods in [39, 30] generate walking trajectories without really understanding what the camera-wearer’s actual goals are, often resulting in overly simplistic and unrealistic trajectories.

In contrast to these prior methods, we consider a complex one-on-one basketball game scenario, for which we not only infer the goals of the camera-wearer, but also generate an egocentric basketball motion sequence that is aligned with these goals. We also point out that, unlike the methods in [39, 30] which require depth input or the positions of other people in the scene, our method operates using only a single first-person RGB input image.

## 3. Motivation and Challenges

**Representing a State.** Generating an egocentric basketball motion sequence can be viewed as the problem of mapping a first-person image  $x$  to a sequence  $y_{1:M}$ , where each

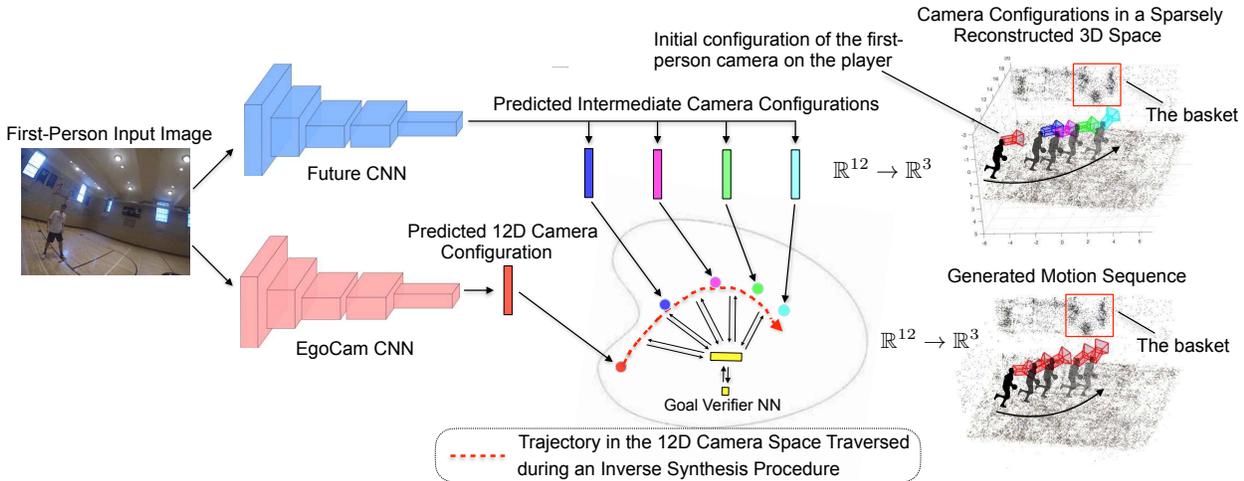


Figure 2: Our model takes a single first-person image as input and outputs an egocentric basketball motion sequence in the form of a 12D camera configuration trajectory, encoding a player’s 3D location and 3D head orientation throughout the sequence. First, we feed the first-person image through our proposed future CNN to predict an initial sequence of future 12D camera configurations. Then, we use our proposed inverse synthesis procedure to synthesize a refined camera configuration sequence that matches the configurations predicted by the future CNN, while also maximizing the output of the goal verifier network. The goal verifier network is a fully-connected network trained to verify that a given 12D camera configuration is consistent with the final goals of real players. Finally, by following the trajectory resulting from the refined camera configuration sequence, we obtain the complete 12D motion sequence.

entry  $y_i$  is a vector representing a state in this sequence of length  $M$ . In this paper, we use a notation where the entire sequence is denoted as  $y$  (without subscript indices), whereas an  $i^{th}$  state in the sequence  $y$  is denoted as  $y_i$ .

We propose to encode each state  $y_i$  in the sequence as a 12D camera configuration, which captures 3D location and 3D orientation of the camera on a player’s head. In contrast to prior models [30, 39] that just use a 2D  $(x, y)$  location representation, our selected camera configuration representation allows us to represent more complex basketball motion patterns, while still being compact and interpretable. Such representation also enables us to learn our model without using manually labeled behavioral data.

**Generating Motion Sequences.** Generating motion sequences that are realistic and smooth is a challenging problem because the actions taken by the camera wearer are noisy. As a result, it is difficult to accurately learn the transitions between two adjacent states from a limited amount of data. Furthermore, due to the recurrent nature of such predictions (i.e., the predicted state  $\hat{y}_{i+1}$  depends on  $\hat{y}_i$ ), the error starts accumulating and exploding, which makes it difficult to generate longer sequences accurately. This issue is especially prevalent with the standard RNN and LSTM approaches, which try to learn such transitions sequentially.

In this work, we propose a model that first predicts a set of intermediate states that are consistent with how real basketball players move. Afterwards, we use our proposed inverse synthesis procedure to generate a full motion sequence. In the experimental section, we verify the effective-

ness of our approach and show that it outperforms methods such as RNNs, LSTMs, and GANs.

## 4. Egocentric Basketball Motion Model

In Figure 2, we present a detailed illustration of our egocentric basketball motion model. First, we use our proposed future CNN to predict an initial sequence of 12D camera configurations, which aim to capture how real players move during a one-on-one basketball game. After that, we use our proposed inverse synthesis procedure to generate a refined camera configuration sequence that (1) matches the initial camera configuration sequence predicted by the future CNN and (2) maximizes the output of the goal verifier network, which is trained to verify whether a given 12D camera configuration aligns with realistic one-on-one basketball goals. Finally, by following the trajectory resulting from the refined camera configuration sequence, we obtain the complete 12D motion sequence. We now discuss each component of our model in more detail.

### 4.1. Egocentric Camera (EgoCam) CNN

An egocentric camera (EgoCam) CNN is used to map a given first-person image  $x_i$  into a 12D camera configuration  $\hat{y}_i \in \mathbb{R}^{1 \times 12}$  that encodes the 3D location and 3D orientation of the camera on a player’s head. We implement our EgoCam CNN using a popular ResNet-101 [12] architecture. The network is optimized using the following L2 loss:

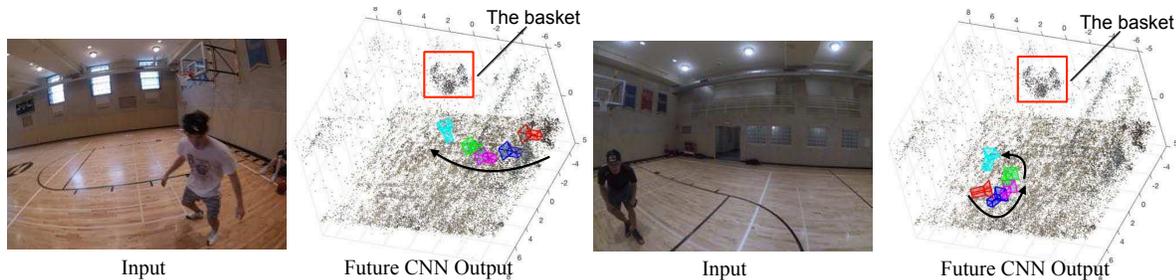


Figure 3: An illustration of the camera configurations generated by our future CNN, which we visualize in a sparsely reconstructed 3D space (best viewed in color). The red camera depicts the initial camera configuration state, while the blue, magenta, green, and cyan cameras correspond to the outputs from the 1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup>, and 4<sup>th</sup> branches of our future CNN, respectively. We note that our future CNN is able to produce a diverse set of intermediate configurations, which allows us to generate a wide array of different sequences at a later step in our model.

$$L_{cam} = \|\hat{y}_i(x_i) - K(x_i)\|_2^2 \quad (1)$$

where  $\hat{y}_i \in \mathbb{R}^{1 \times 12}$  is the predicted camera configuration for an image  $x_i$ , and  $K(x_i) \in \mathbb{R}^{1 \times 12}$  is a vector that encodes a true player’s 3D location and 3D head orientation.  $K(x_i)$  is obtained by flattening the egocentric  $3 \times 4$  camera matrix, which is produced by an unsupervised structure from motion (SfM) algorithm. However, due to the jittery nature of first-person images, SfM works only on a small portion of the input images (i.e.  $\approx 20\%$  of all images). This motivates the need for an EgoCam CNN, which allows us to generate 12D configurations for any first-person image.

Our EgoCam CNN produces similar outputs to the actual SfM algorithm. We also report that we tried retrieving missing 12D configurations using a nearest neighbor algorithm, but observed that our EgoCam CNN produced better results (2.54 vs. 1.97 L2 error in the normalized 12D space).

## 4.2. Future CNN

Given a first person image  $x_i$  from the beginning of a sequence, the future CNN encodes it into multiple configurations  $\phi(x_i) \in \mathbb{R}^{k \times 12}$ , which represent intermediate states that capture how real players move during a one-on-one game. Here,  $k$  is a parameter that controls how many intermediate configurations we want to generate. Our future CNN is implemented using a multi-path ResNet-101 architecture, meaning that after the final convolutional *pool5* layer, the network splits itself into  $k$  branches, each denoted as  $\phi_j(x_i)$  for  $j = 1 \dots k$ . Each branch in the future CNN is responsible for generating its own intermediate state.

To train the future CNN, for every real basketball sequence, we first assign to each camera configuration  $y_i$  of the sequence a value  $s \in [0, 1]$ , indicating its order in the sequence. Let  $y_i$  be the  $i^{\text{th}}$  configuration in a sequence of length  $M$ . Then, we can compute  $s$  as  $s(i) = i/M$ . The  $j^{\text{th}}$  branch in the future CNN is then optimized to predict camera configurations, with  $s$  values falling in the interval

$[\frac{j-1}{k}, \frac{j}{k}]$ , where  $k$  is the number of branches in the future CNN. For instance, if  $k = 4$ , then the second branch in the network will be optimized to predict camera configurations with values  $s \in [0.25, 0.5]$ . This constraint ensures that each branch generates an intermediate state associated with a specific time-point in a sequence, thus generating configurations covering the entire sequence (albeit with wider gaps). We also point out that all input images  $x_i$  that are used to train the future CNN have values  $s \in [0, 0.1]$  to make sure that the future CNN generates accurate intermediate configurations from a first-person image at the beginning of a sequence. The future CNN is then trained using the following loss function:

$$L_{fut} = \sum_{j=1}^k \|\phi_j(x_i) - \hat{y}_{i'}(x_{i'})\|_2^2 \quad (2)$$

where  $\phi_j$  denotes the output from branch  $j$  of a future CNN, and  $\hat{y}_{i'}$  is the output of an EgoCam CNN for an input image  $x_{i'}$ . The constraints on  $x_i$  and  $x_{i'}$  that we discussed above are expressed as:  $s(i) \in [0, 0.1]$  and  $s(i') \in [\frac{j-1}{k}, \frac{j}{k}]$ .

We note that our training setup requires basketball sequences that are trimmed. In this work, we trim the sequences manually: the sequence starts when a player begins his offensive position and ends when a shot or a turnover occurs. We believe that we could also trim sequences automatically by using a player’s head location and pose to assess whether a player is attacking or defending.

## 4.3. Goal Verifier Network

The goal verifier network aims to verify whether a given configuration aligns with the final goals of real players. The goal verifier takes a 12D camera configuration  $\hat{y}_i \in \mathbb{R}^{1 \times 12}$  as input, then outputs a real value  $\psi(\hat{y}_i) \in \mathbb{R}^{1 \times 1}$  in the interval  $[0, 1]$ , indicating how well a given camera configuration captures the final goals of real players.

The key question is: how do we infer the final goals of real basketball players without asking them directly? To

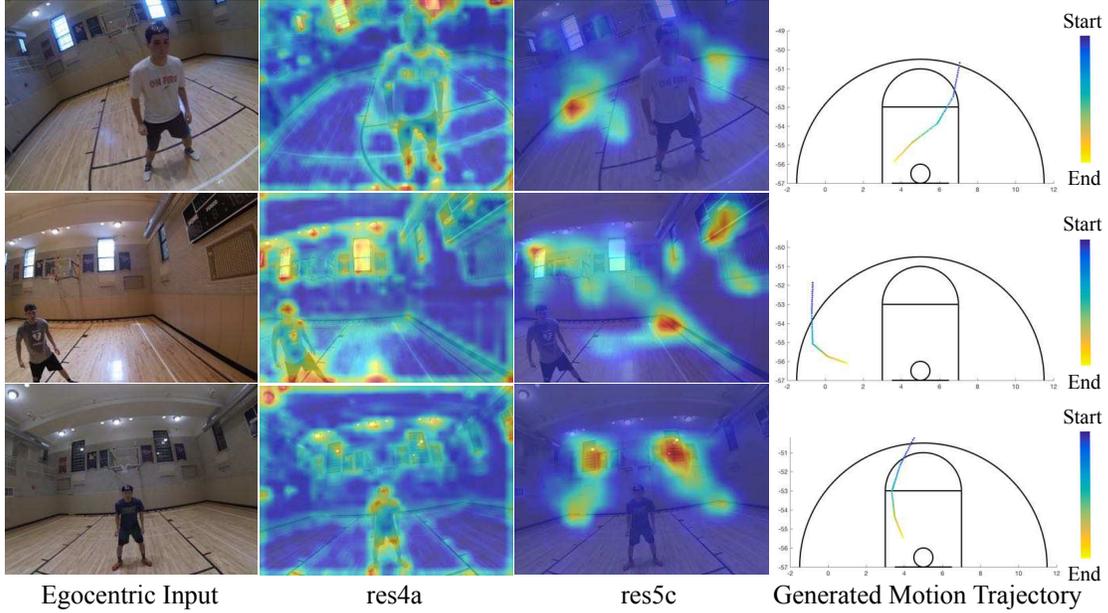


Figure 4: A figure illustrating some of our qualitative results. In the first column, we depict an egocentric input image. In the second and third columns, we visualize the activations of a Future CNN from the res4a and res5c layers. Finally, in the last column, we present our generated 2D motion trajectories. Based on the activations of the Future CNN, we conclude that our CNN recognizes visual cues in an egocentric image, which may be helpful for deciding how to effectively navigate the court and reach the basket, or how to get away from a defender. Furthermore, our generated motion trajectories seem realistic, as they avoid colliding with the defender and typically end near the basket, which reflects how most real players would move.

do this, we assume that the last few images in a given sequence represent the final goals of that player. Of course, such reasoning may not always be correct because, in some sequences, players may fail to accomplish their goals. However, we conjecture that when looking at many sequences of real players, the pattern of goals should be quite distinctive. In other words, if we use the right learning strategy, the network should be able to learn configurations that are typically associated with the final goals of real players.

We propose to employ discriminative training of the goal verifier, which allows us to effectively address the issues of noisy labels in our setting. To create these noisy labels, we assign a value  $g(\hat{y}_i) = 1$  to all configurations with  $s(i) > 0.92$  (configurations that appear at the end of sequences) and a value of zero to all other configurations. Such a scheme allows us to highlight the configurations at the end of sequences, which are more likely to represent the goals of the players. Subsequently, the goal verifier, which is a two-layer network, takes a 12D camera configuration as its input and is trained using a cross-entropy loss:

$$L_g = -[g(\hat{y}_i) \log \psi(\hat{y}_i) + (1 - g(\hat{y}_i)) \log (1 - \psi(\hat{y}_i))] \quad (3)$$

where  $\hat{y}_i \in \mathbb{R}^{1 \times 12}$  is the 12D camera configuration output from an EgoCam CNN (associated with an image  $x_i$ ), and  $\psi(\hat{y}_i) \in \mathbb{R}^{1 \times 1}$  is the output of a goal verifier network

that indicates whether a given 12D camera configuration accurately represents the final goals of real players.

#### 4.4. Motion Planning using Inverse Synthesis

We can now put all the pieces of our model together and show how to generate an egocentric basketball motion sequence that captures the goals of real players. Our approach is partially inspired by the idea of synthesizing the preferred stimuli in a CNN [42, 26, 27], which has been mostly used to visualize activations in the CNNs [42, 26, 27]. In this work, we propose the concept of inverse synthesis for generating a realistic basketball motion sequence.

Consider the two problems that we were solving previously: (1) encoding an egocentric basketball motion sequence via a set of intermediate 12D configurations and (2) verifying that a given 12D camera configuration aligns with the final goals of real players. In the previous sections, we solved these two problems using the future CNN and the goal verifier network, respectively. However, we now want to invert these two problems: given (1) a set of intermediate states predicted by the future CNN and (2) a trained goal verifier network, our goal is to generate a smooth basketball motion sequence in the form of 12D camera configurations.

To do this, we frame the inverse synthesis problem as a search problem in the 12D camera space, where our goal is to find a 12D configuration  $h$  that (1) matches intermediate states generated by the future CNN  $\phi_j(x_i)$  and (2) maxi-

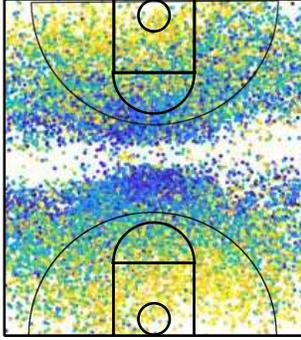


Figure 5: A figure illustrating the 3D locations of every camera configuration from our dataset mapped on a 2D court (best viewed in color). The blue points indicate the configurations from the beginning of sequences, whereas the yellow points depict configurations from the end of sequences. The diversity of real sequences in our dataset allows us to build a powerful basketball motion model.

minimizes the output of a goal verifier network  $\psi(h)$ . We formulate this problem as a minimization problem in the 12D camera configuration space:

$$h^* = \arg \min_h [|\phi_j(x_i) - h|_2^2 - \log(\psi(h))] \quad (4)$$

Here,  $h$  is a 12D camera configuration vector that we initialize to  $\hat{y}_i(x_i)$ , which is the camera configuration of the very first image in the sequence (i.e.,  $s(i) = 0$ , or, equivalently,  $i = 0$ ). The first term in the equation encourages  $h$  to reach configurations predicted by the future CNN, whereas the second term encourages  $h$  to take values that would maximize the output of the goal verifier network. We minimize this function by computing the appropriate gradients and then running gradient descent in the 12D camera configuration space for  $N$  iterations. To select the branch from a future CNN whose output  $\phi_j$  we are trying to match, we compute  $j = \text{floor}(c/(N/k))$ , where  $j$  is the index of a selected branch,  $c$  is the iteration counter, and  $k$  is the number of branches in the future CNN.

During the inverse synthesis procedure, we fix the parameters of all three networks and only adjust  $h$ . Since the inverse synthesis is performed in the 12D camera configuration space, at every step we are generating a new 12D camera configuration. Thus, at the end, we can generate a final camera configuration sequence by simply following the 12D camera trajectory traversed during the optimization.

#### 4.5. Implementation Details

For all of our experiments, we used the Caffe library [17]. Both the future and EgoCam CNNs were based on the ResNet-101 [12] architecture and were trained for 10,000 iterations, with a learning rate of  $10^{-4}$ , 0.9 momentum, a weight decay of  $5 \cdot 10^{-4}$ , and 20 samples per batch.

	Evaluation Tasks	
	PFM ↓	CG ↑
GAN [11]	62.31	0.329
LSTM [14]	5.66	0.678
RNN [16]	5.82	0.612
NN	5.36	-
<b>Ours w/o GV</b>	<b>4.91</b>	0.671
<b>Ours w/ GV</b>	4.93	<b>0.776</b>

Table 1: We assess each method on two tasks: predicting a player’s future motion (PFM), and capturing the goals of real players (CG). The  $\uparrow$  and  $\downarrow$  symbols next to a task indicate whether a higher or lower number is better, respectively. Our method outperforms the baselines for both tasks, suggesting that we can generate sequences that (1) are more similar to a true player’s motion and (2) capture the goals of real players more accurately. We also note that removing a goal verifier (“Ours w/o GV”) leads to a sharp decrease in performance, according to the CG evaluation metric.

We designed our future CNN to have 4 distinct branches, which we experimentally discovered to work well. Each branch consists of a single fully-connected layer that maps *pool5* features to a 12D camera configuration feature. Next, we designed the goal verifier network as a two layer fully connected network with 100 neurons in the first-hidden layer and a single output neuron at the end. To generate final basketball motion sequences, we ran our inverse synthesis procedure for 6,000 iterations with a step size of 0.001.

## 5. Egocentric One-on-One Basketball Dataset

We present a first-person basketball dataset consisting of 988 sequences from one-on-one basketball games between nine college-level players. The videos are recorded in  $1280 \times 960$  resolution at 100 fps using GoPro Hero Session cameras, which are mounted on the players’ heads. We extract the videos at 5 frames per second. We then randomly split all the sequences into training and testing sets, consisting of 815 and 173 sequences, respectively. Each sequence is about 25 frames long.

To obtain better insight about the distribution of our data, we map the 3D locations of all camera configurations from real sequences onto the 2D court and visualize the results in Figure 5. The blue points indicate the configurations at the beginning of sequences, whereas the yellow points represent configurations at the end of sequences. Based on this figure, we note that our dataset contains a wide array of different beginning and ending configurations, enabling us to learn a diverse basketball motion model.

We chose a one-on-one setting because it is more data-efficient for studying the decision-making process of basketball players (compared to the five-on-five setting). For example, first-person videos of five-on-five games capture



Figure 6: A visual illustration of our generated basketball sequences, in the form of first-person images retrieved using nearest neighbor search. We observe that our generated sequence is similar to a real sequence, suggesting that our model could be used for applications such as future behavior prediction.

numerous instances when players do not have the ball and are idly waiting for their teammate with the ball to perform some action. On the contrary, in our one-on-one dataset, players continuously make decisions and take actions to reach their goals, as there are no teammates to rely on.

## 6. Experimental Results

To the best of our knowledge, we are the first to generate egocentric basketball motion sequences in the 12D camera configuration space from a single first-person image. As a result, there are no prior established baselines for this task. To validate our method’s effectiveness, we compare it with other popular deep learning techniques such as RNNs [16], LSTMs [14], and GANs [11]. We construct these baselines using the ResNet-101 [12] architecture. Each baseline takes a single first-person image as input and is then trained to generate 10 future configurations. The RNN and LSTM baselines are optimized with an L2 loss in the 12D camera configuration space, whereas the GAN baseline has the same architecture as RNN, but is trained to fool the discriminator by producing realistic 12D future camera configurations. To maximize the amount of available training data, the training data are constructed by using every feasible training image as a starting point. During testing, each baseline predicts camera configurations recurrently until we observe no significant change in the prediction or until the predicted sequence length exceeds the length of a real sequence. We also include a nearest neighbor baseline, which operates in the 12D camera configuration space.

The goal of our evaluations is to verify that the generated sequences (1) are realistic and (2) that they capture the goals of real players. To do this, we propose several evaluation schemes, which we describe in the next few subsections.

### 6.1. Quantitative Results

**Predicting Future Motion.** To verify that our generated sequences are realistic, we examine whether our method can

predict a player’s future motion. In the context of such evaluation, the sequences that we want to compare typically have different lengths, which renders standard Euclidean and L1 distance metrics unusable. Thus, we use the Hausdorff distance [15], which is commonly used for sequential data comparisons. To implement this idea, we first use our EgoCam CNN to extract camera configurations from every frame in every real sequence in the testing dataset. Then, as our evaluation metric, we compute a distance  $d = D(y, \hat{y})$ , where  $\hat{y}$  is generated using the first image from a real sequence  $y$ , and  $D$  is a Hausdorff distance operator.

In Column 2 of Table 1, we record the average  $d$  over all testing sequences for every method (i.e., the lower the distance the better). We observe that our method outperforms all the other baselines, suggesting that we can use it for future behavior prediction.

**Capturing the Goals of Real Players.** Earlier, we assumed that the last few frames in real sequences approximate the goals of the players. To examine how well our method captures these goals, we first select the last generated configuration  $\hat{y}_M$  in the sequence of length  $M$ . Then, we use nearest neighbor search to retrieve a real configuration  $y_{nn}$  that is most similar to  $\hat{y}_M$ . Finally, we look up where in the real sequence  $y_{nn}$  appears. For instance, if  $y_{nn}$  was the  $i^{th}$  configuration in a real sequence of length  $N$ , then we assign our current prediction a value of  $v = i/N$ . Intuitively, if the last generated configuration is similar to the last real configuration, we conclude that our method was able to capture the goals of real players.

In Column 3 of Table 1, we record the average  $v$  values over all testing sequences for every method (i.e., the higher the better). These results suggest that, out of all baselines, our method is the most accurate at capturing the goals of real players. Also, note that, if we remove the goal verifier, these results drop significantly, indicating the goal verifier’s importance to our system. We conjecture that this happens because the future CNN is trained using a non-

discriminative L2 loss, while the goal verifier network is trained to discriminate which configurations represent the final goals of the players. Thus, similar to GANs [11], due to the discriminative training, the goal verifier may be more useful for generating the configurations that capture the final goals of the players. We also note that the future CNN is still essential, as using a goal verifier network alone produces short and unrealistic sequences.

## 6.2. Qualitative Results

**Future CNN Visualization.** To better understand how the future CNN works, we visualize its outputs in Figure 3. To do this, we take the 12D camera configurations generated by the future CNN, transform them into 3D space, and then visualize them in a sparsely reconstructed 3D scene. Note that the future CNN produces a wide array of different configurations, allowing us to generate diverse sequences.

Furthermore, in Columns 2 and 3 of Figure 4, we visualize the activations of a future CNN from the *res4a* and *res5c* layers, respectively. We observe that, in the *res4a* layer, the CNN focuses on different parts of a defender’s body, which makes intuitive sense, as basketball players often use such information to decide their next action. Furthermore, we also observe that, in the *res5c* layer, the future CNN learns to recognize open spaces in the court that could be used by a player to navigate the court and get away from a defender.

**Visualizing Generated Motion Trajectories.** In the last column of Figure 4, we also visualize sequences generated by our inverse synthesis procedure (by projecting them onto a 2D court). Even though it is difficult to validate whether our generated motion sequences represent optimal motion pattern, they do seem reasonable. In all visualized instances, our model produces motion sequences that avoid colliding with the defender. Furthermore, most of our generated sequences end around the basket, which reflects what a real player would likely try to do.

**Retrieved Image Sequences.** To show what our generated sequences look like in the form of first-person images, we use nearest neighbor search in the 12D camera space to retrieve the most similar first-person images from the training data. We then sample 5 first-person images from the sequence and visualize them in Figure 6, along with a real sequence. The similarity between our generated sequences and the real sequences suggests that our model accurately captures how real players move during a basketball game.

Additionally, to qualitatively validate the need for a goal verifier, in Figure 7, we visualize the very last images in our generated sequences, when the goal verifier is not used (w/o Goal Verifier) and when it is used (w/ Goal Verifier). Note that when we remove the goal verifier from our system, the very last images in our generated sequences look less realistic. That is, they focus on the walls or floor of the basketball court, which is probably not something that real players



Figure 7: A figure comparing the final images of our generated sequences in two settings: **without** and **with** using the goal verifier network. We retrieve these images via nearest neighbor search. Note that the images produced with a goal verifier “focus” on the basket, just as real players would, thus capturing the goals of real players more accurately.

would do. In contrast, adding a goal verifier to our system makes these images focus on the basket, which makes much more sense, since most players finish their sequences when they are looking directly at the basket. Thus, the goal verifier allows us to generate more realistic sequences.

**Video Results.** Due to space constraints, we cannot include all of our qualitative results in an image format. Furthermore, because images are static, they cannot capture the full content of our generated sequences. Thus, we include more video results in the following link: <https://www.youtube.com/watch?v=wRRR14QsUQg>.

## 7. Conclusions

In this work, we introduced a model that uses a single first-person image to generate an egocentric basketball motion sequence in the form of a 12D camera configuration trajectory. We showed that our model generates realistic sequences that capture the goals of real players. Furthermore, we demonstrated that our model can be learned directly from the first-person video data, which is beneficial as obtaining labeled behavioral data is costly.

Our model could be used for a wide array of behavioral applications, such as future behavior prediction or player development, for which we could build models using the data of expert players and then transfer their behavioral patterns to less-skilled players. Furthermore, even though we apply our model on a basketball activity, we do not inject any basketball-specific domain knowledge into the model. Thus, we believe that our model is general enough to also be applied to other activities, which we will explore in our future work. In the future, we would also like to experiment with more complex configurations that would allow us to represent even more complicated behavioral patterns. Finally, we believe that the concept behind our model could be used for various robotic applications, where our model could provide behavioral guidelines for robots.

## References

- [1] <https://www.stats.com>. 2
- [2] Alexandre Alahi, Kratharth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *CVPR*, 2016. 2
- [3] Alexandre Alahi, Vignesh Ramanathan, and Fei-Fei Li. Socially-aware large-scale crowd forecasting. In *CVPR*, pages 2211–2218. IEEE Computer Society, 2014. 2
- [4] Saad Ali and Mubarak Shah. *Floor Fields for Tracking in High Density Crowd Scenes*, pages 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. 2
- [5] Gedas Bertasius, Hyun Soo Park, Stella X. Yu, and Jianbo Shi. Am i a baller? basketball performance assessment from first-person videos. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2017. 2
- [6] Gedas Bertasius, Hyun Soo Park, Stella X. Yu, and Jianbo Shi. First-person action-object detection with egonet. In *Proceedings of Robotics: Science and Systems*, July 2017. 2
- [7] Gedas Bertasius, Hyun Soo Park, Stella X. Yu, and Jianbo Shi. Unsupervised learning of important objects from first-person videos. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2017. 2
- [8] Dima Damen, Teesid Leelasawassuk, Osian Haines, Andrew Calway, and Walterio Mayol-Cuevas. You-do, i-learn: Discovering task relevant objects and their modes of interaction from multi-user egocentric video. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014. 2
- [9] Alireza Fathi, Ali Farhadi, and James M. Rehg. Understanding egocentric activities. In *ICCV*. 2
- [10] Alireza Fathi, Xiaofeng Ren, and James M. Rehg. Learning to recognize objects in egocentric activities. In *CVPR*, pages 3281–3288. IEEE Computer Society, 2011. 2
- [11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. 2014. 2, 6, 7, 8
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, 2015. 3, 6, 7
- [13] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51:4282–4286, May 1995. 2
- [14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. 2, 6, 7
- [15] D. P. Huttenlocher, G. A. Klanderman, and W. A. Rucklidge. Comparing images using the hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(9):850–863, September 1993. 7
- [16] L. C. Jain and L. R. Medsker. *Recurrent Neural Networks: Design and Applications*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1999. 2, 6, 7
- [17] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014. 6
- [18] Kris Makoto Kitani, Takahiro Okabe, Yoichi Sato, and Akihiro Sugimoto. Fast unsupervised ego-action learning for first-person sports videos. In *CVPR*, 2011. 2
- [19] Hoang Le, Peter Carr, Yisong Yue, and Patrick Lucey. Data-driven ghosting using deep imitation learning. In *MITSSAC*, 2017. 2
- [20] Namhoon Lee and Kris M Kitani. Predicting wide receiver trajectories in american football. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016. 2
- [21] Yong Jae Lee and Kristen Grauman. Predicting important objects for egocentric video summarization. *IJCV*, 2015. 2
- [22] Yin Li, Zhefan Ye, and James M. Rehg. Delving into egocentric actions. In *CVPR*. 2
- [23] Yuxuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. *CoRR*, abs/1707.03374, 2017. 2
- [24] Zheng Lu and Kristen Grauman. Story-driven summarization for egocentric video. In *CVPR*, 2013. 2
- [25] Wei-Chiu Ma, De-An Huang, Namhoon Lee, and Kris M. Kitani. Forecasting interactive dynamics of pedestrians with fictitious play. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2
- [26] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 5
- [27] A. Mahendran and A. Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, pages 1–23, 2016. 5
- [28] Ramin Mehran, Alexis Oyama, and Mubarak Shah. Abnormal crowd behavior detection using social force model. In *CVPR*, pages 935–942. IEEE Computer Society, 2009. 2
- [29] Kris Kitani Minghuang Ma. Going deeper into first-person activity recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2
- [30] Hyun Soo Park, Jyh-Jing Hwang, Yedong Niu, and Jianbo Shi. Egocentric future localization. In *CVPR*, 2016. 2, 3
- [31] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc J. Van Gool. You’ll never walk alone: Modeling social behavior for multi-target tracking. In *ICCV*, pages 261–268. IEEE Computer Society, 2009. 2
- [32] Alex Pentland and Andrew Liu. Modeling and prediction of human behavior. *Neural Comput.*, 11(1):229–242, January 1999. 2
- [33] Hamed Pirsiavash and Deva Ramanan. Detecting activities of daily living in first-person camera views. In *CVPR*, 2012. 2
- [34] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. *CoRR*, abs/1707.02920, 2017. 2
- [35] Xiaofeng Ren and Chunhui Gu. Figure-ground segmentation improves handled object recognition in egocentric video. In *CVPR*, 2010. 2
- [36] Nicholas Rhinehart and Kris M. Kitani. First-person activity forecasting with online inverse reinforcement learning. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2
- [37] Suriya Singh, Chetan Arora, and C. V. Jawahar. First person action recognition using deep learned descriptors. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2
- [38] Bilge Soran, Ali Farhadi, and Linda Shapiro. *Action Recognition in the Presence of One Egocentric and Multiple Static Cameras*. 2015. 2
- [39] Shan Su, Jung Pyo Hong, Jianbo Shi, and Hyun Soo Park. Predicting behaviors of basketball players from first person videos. In *CVPR*, 2017. 2, 3

- [40] Yu-Chuan Su and Kristen Grauman. Detecting engagement in ego-centric video. In *European Conference on Computer Vision (ECCV)*, 2016. [2](#)
- [41] Chenxia Wu, Jiemi Zhang, Bart Selman, Silvio Savarese, and Ashutosh Saxena. Watch-bot: Unsupervised learning for reminding humans of forgotten actions. 2016. [2](#)
- [42] Jason Yosinski, Jeff Clune, Anh Mai Nguyen, Thomas J. Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *CoRR*, abs/1506.06579, 2015. [5](#)
- [43] Stephan Zheng, Yisong Yue, and Jennifer Hobbs. Generating long-term trajectories using deep hierarchical networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1543–1551. Curran Associates, Inc., 2016. [2](#)