What Players do with the Ball: A Physically Constrained Interaction Modeling

Supplementary materials

Andrii Maksai Xinchao Wang Pascal Fua
Computer Vision Laboratory,
École Polytechnique Fédérale de Lausanne (EPFL)
{andrii.maksai, xinchao.wang, pascal.fua}@epfl.ch

We begin by describing the ball states we use for different sports, and associated physical models, as discussed in Sec. 3.3 of the paper. Next, we give the details of the state classifier that we use in Sec. 4. Then we describe the details of ball graph creation procedure of Sec. 5. We finish by giving details of the post-processing for the videos we show, and the details on computational efficiency of our approach.

1 Ball states

Here we describe the physical models associated with the different ball states, which we introduced in Sec. 3.3 of the paper. We use different states for the ball in different sports. We use some states - *flying*, *in_possession* for all sports, and others only for some. For volleyball we add *strike*, and for basketball *pass*. For both soccer and volleyball we add *rolling*, and for basketball our sequences did not feature the ball rolling on the ground. Tab. 1 describes the physical constants for all states, introduced in Eq. 5 of the paper. Note that in all cases we limit the absolute value of acceleration / speed / location of the ball, which means that we actually have 2 constraints for each row in the table, with exactly opposite coefficients

$$A^{s,c}(P_c^t - 2P_c^{t-1} + P_c^{t-2}) + B^{s,c}(P_c^t - P_c^{t-1}) + C^{s,c}P_c^t - F^{s,c} \leq K(3 - M_{s,c}^t - M_{s,c}^{t-1} - M_{s,c}^{t-2}) ,$$

$$-A^{s,c}(P_c^t - 2P_c^{t-1} + P_c^{t-2}) - B^{s,c}(P_c^t - P_c^{t-1}) - C^{s,c}P_c^t + F^{s,c} \leq K(3 - M_{s,c}^t - M_{s,c}^{t-1} - M_{s,c}^{t-2}) .$$

They limit the values from above and from below, respectively. Notation used is the same notation we use in Eq. 5 of the paper. Note that physical models for states *flying*, *strike* and *pass* are identical. We differentiate between those states using our state classifier.

2 State classifier

Here we describe the state classifier, introduced in Sec. 4. For each sport, we learn a multi-class Random Forest classifier to predict the ball state. We use

State(s) s	Axis c	$A^{s,c}$	$B^{s,c}$	$C^{s,c}$	$F^{s,c}$	Explanation
flying, strike, pass, rolling	X	1	0	0	0	Constant speed in ground plane
flying, strike, pass, rolling	Y	1	0	0	0	Constant speed in ground plane
flying, strike, pass	Z	1	0	0	$\frac{-g}{fps^2}$	Negative g acceleration in vertical plane
rolling	Z	0	0	1	0	Constant height of the ball
flying, pass, rolling	X, Y, Z	0	1	0	$\frac{20000(mm)}{fps}$	Maximum speed limitation
strike	X, Y, Z	0	1	0	$\frac{35000(mm)}{fps}$	Maximum speed limitation

Table 1: Physical models associated with different states of the ball. $g = 9810(\frac{mm}{s^2})$ denotes the free fall acceleration, fps denotes frame rate of the video sequence. Coefficients shown only for constraints that limit from above. Coefficients for limiting from below have the same magnitude as those in the table but are negative.

the 3D ball location and the number of people around it as features. We use neighbourhoods of sizes 1000mm / 500mm and 2000mm / 2500mm for volleyball / basketball / soccer respectively.

When we are simultaneously tracking the ball and the players, the ground truth positions of the players are unknown. In this case, we substitute the number of people by the predicted number of people, which we compute by integrating the Probabilistic Occupancy Map near the ball location. To improve the performance of the classifier given limited amount of available data we take advantage of the court symmetry of the field with respect to the 180 degrees rotation around the center. We treat all data points as if they were located on one side of the field.

Fig. 1 presents a partial evaluation of tracking accuracy as a function of the number of frames in the training data to address the question of whether the training data we use is sufficient. Clearly, the more the better, but above 1000 frames the further improvement becomes small.

Fig. 2 depicts the output of our classifier for volleyball data. Near the ground, probability of having a freely flying ball is low, and most predictions correspond to *flying* and *in_possession*. At 2.5 meters height, predictions are mixed, with *flying* predictions at the ends of the field, that correspond to locations from which the players serve the ball. At the height of 3.5m predicted state is mostly *flying*, except for a stripe in the middle, where players often strike the ball after the jump. Possession by players at such height is not likely. Higher than 4 meters classifier predicts *flying* for all locations. We have checked that cross-validation classification error was under 7% for all our datasets.

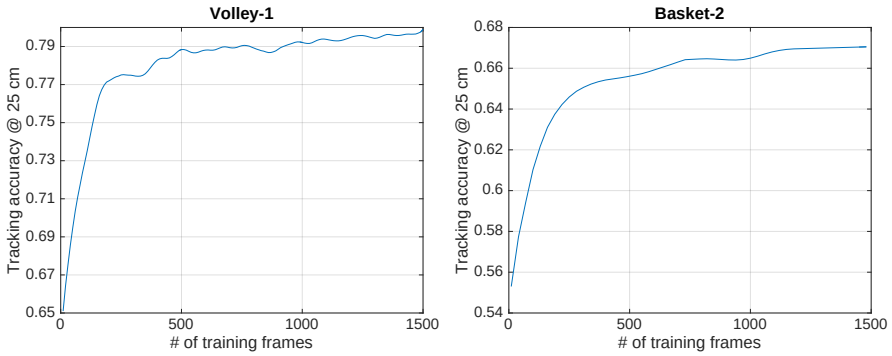


Figure 1: Tracking accuracy at 25 cm for volleyball and basketball as a function of the number of training frames. For consistency, we increased the number of training frames for Basket-2 to 1500.

3 Graph construction

Here we describe the full process of graph construction briefly introduced in Sec. 5.2 of the paper. We first give details of graph construction for states associated with a physical model. We then describe graph construction for *in_possession* state, *not_present* state, and the rule for building edges between nodes with different states.

3.1 States with physical model

As discussed in Sec. 5.2 of the paper, we compute tracklets by joining detections using the K-Shortest-Paths algorithm (Fig. 2(a)). Then we create trajectories that go through these tracklets, and additional trajectories that join tracklets and previously built trajectories. The trajectories of the first type (red in Fig. 2(b)) represent hypothesized ball locations where the detector has been unable to find the ball. The trajectories of the second type (red in Fig. 2(c)) represent hypothesized ball locations when the whole trajectory is missing.

To create ball trajectories that go through the tracklets, we use the following procedure:

1. We start from every pair of consecutive detections in the tracklet. We fit a trajectory that obeys the physical model and goes through these two points. Note that there is only one straight line (for *rolling*) and only one parabola (for *flying*, *strike*, *pass*) that goes through two given points. Uniqueness of parabola is due to fixed force of gravity.

2. We grow the trajectory. At each next frame, if there is indeed a detection within the distance D_l of the trajectory, we add it to the trajectory, and recompute the best model fit through a new set of detections. If there are several detections, we pick the one with the highest confidence. We continue growing the trajectory until in leaves the tracking area

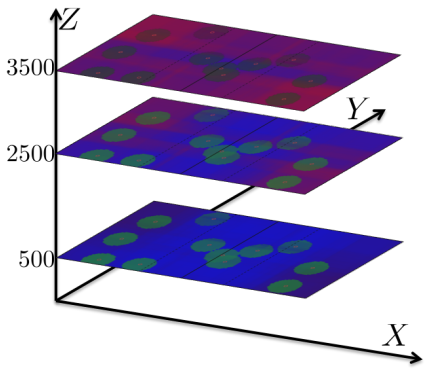


Figure 2: Example of the classifier output on the volleyball data. Input to the clasifier is the 3D location of the ball and the number of people in the vicinity. Picture shows the output of classifier for different xy-locations of the ball, and different heights. Ground-plane positions of the players are denoted by little red circles. R,G,B components of the color indiciate the output of the classifier, probability of states *flying*, *in_possession*, and *strike*, respectively.

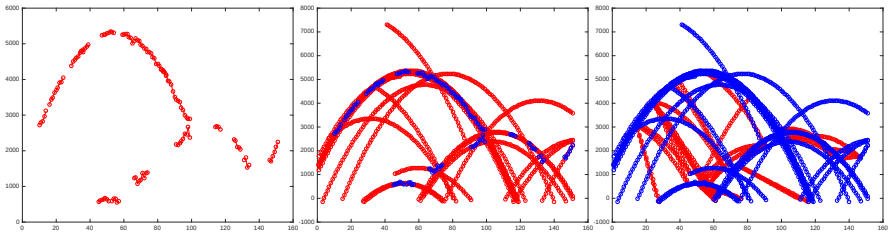
3. From a set of trajectories, we keep only those that are associated with the maximal set of detections. In other words, if we have a trajectory with a set of detections, that is a subset of detections of some other trajectory, we discard the former.

A result of this procedure are the red curves of Fig. 2(b). D_l represents the distance between the detection and continuous location of the ball, as defined by Eq. 4 of the paper. We use $D_l = 25cm$ for basketball and volleyball, and $D_l = 75cm$ for football. The value is larger for football both because the cameras are further away from the players and because players often spin the ball so that it follows a curved, rather than straight trajectory. Furthermore, friction can be quite high for the rolling ball, violating the constant velocity constraint.

To join the tracklets and previously built trajectories, we consider all starting and ending points of trajectories and tracklets. We link every pair that are at most 4 seconds apart. We have empirically found that linking those further apart does not improve matters in terms of accuracy, but increases the computational burden. Fig. 2(c) depicts the result of this procedure.

3.2 In_possession state

For the *in_possession* state, we create a copy of every node and every edge in the players graph. We associate with each node a detection with the highest confidence at the distance of D_p from the players possible location. We use $D_p = 1m$ for basketball and volleyball, and $D_p = 2.5m$ for football. The value is larger for football because players can bounce the ball further away from themselves when they run with it.



(a) (b) (c)

Figure 3: Example of building the ball graph. X-axis denotes frames index. Y-axis denotes height of the ball. **(a)** Detections, joined into tracklets. **(b)** Blue: detections, joined into tracklets. Red: set of trajectories associated with maximal sets of detections. **(c)** Blue: tracklets, and trajectories built in the previous step. Red: newly built trajectories that join endpoints of tracklets and trajectories built in the previous step. The resulting set of trajectories depicted is the whole set of nodes and edges built using a particular physical model. Best viewed in color.

3.3 Not_present state

For the *not_present* state, we have one node associated with this state in each time frame. We connect such node by an edge to the node with *not_present* state in the next time frame, as well as to all nodes of all states close to the border of the tracking area. More formally, we connect it to all nodes that are within distance D^{max}, where D^{max} is the maximum distance the ball can travel in any of the states within one frame time. As shown in Tab. 1, we take $D^{max} = \frac{35000(mm)}{fps}$.

3.4 Edges between states

With the exception of *not_present* state, discussed above, we follow one rule for all edges that connect nodes of different states: we join those in the neighbouring frames which are within D^{max} distance of each other.

3.5 Post-processing

We applied post-processing in the form of smoothing to the results for better appearance of videos. **Results reported in the paper are results without smoothing. Furthermore, as we show below, smoothing does not significantly affect tracking accuracy.** Smoothing of the following form was used:

- State smoothing. Ball that left possession of the player and returned within 0.1 seconds is assumed to have never left.
- State smoothing. Ball that was assigned to possession, but returned to flying withing 0.1 seconds and did not change course, is assumed to have stayed on course during this period.

- Ground collision smoothing. We assume the ball to be in contact with the ground in the case of collision if it is below a certain threshold. If there are several such points, we assume them all to have equal zero height.

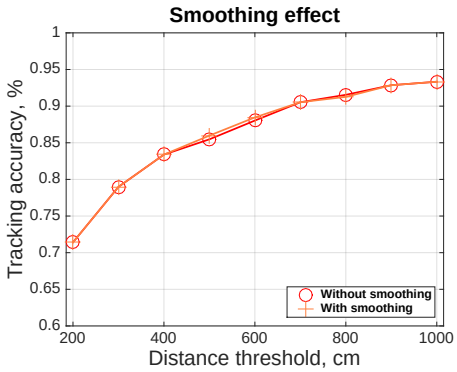


Figure 4: Tracking accuracy curve for results with and without smoothing on Volley-1 dataset

As Fig. 3 shows, tracking accuracy with and without smoothing does not differ much. It was obtained on Volley-1 dataset, and we saw similar results on all other datasets.

4 Computational efficiency

We provide a partial evaluation in Table 2. Running on the volleyball and basketball sequences is faster than on the soccer one, because the ball graph of the soccer sequence is larger than the others by an order of magnitude due to the spurious detections and higher edge density. More specifically, D_p, the vicinity in which the ball can be possessed by players, is higher for soccer as mentioned in the supplementary materials. This results in more states for the soccer graph.

Dataset	**100 frames**	**250 frames**	**500 frames**
1,2	0.2/0.4/1	0.5/1.2/2	3/5.3/15
1,2	0.1/0.3/0.9	5/9.1/12	8/12/38
Apidis	0.2/0.4/0.6	1/2.2/3	7/13.1/25
Issia	3/4/5	16/38/60	794/1072/1350

Table 2: Min/Average/Max processing time (measured in seconds) on batches of different lengths on a 2.5Hz Intel Core i7 processor. Results were computed on several non-overlapping intervals.