Detecting the Point of Release of Virtual Projectiles in AR/VR

Goksu Yamac* Trinity College Dublin
Niloy J. Mitra† University College London
Carol O'Sullivan‡ Trinity College Dublin

Abstract

Our aim is to detect the point of release of a thrown virtual projectile in VR/AR. We capture the full-body motion of 18 participants throwing virtual projectiles and extract motion features, such as position, velocity, rotation and rotational velocity for arm joints. Frame-level binary classifiers that estimate the point of release are trained and evaluated using a metric that prioritizes detection timing to obtain a ranking of joints and motion features. We find that wrist joint and rotation motion feature are most accurate, which can can help when placing simple motion tracking sensors for real-time throw detection.

Index Terms: [I.3.6] [Computer Graphics]—Interaction Techniques; [I.3.7] [Computer Graphics]—Virtual Reality

1 Introduction

Augmented and Virtual Reality (AR/VR) technologies are rapidly changing Human-Computer Interaction (HCI). The focus is shifting from static, goal-oriented and mediated interactions towards more active, natural and immersive experiences, thus requiring a deeper understanding of human behavior, motion and perception. However, the problem of simulating natural physical interactions with virtual entities within a dynamically changing environment (real or virtual) remains an open challenge [10].

Figure 1: VR scene and capture equipment

We explore the physical interaction of throwing a virtual ball within a Virtual Environment (VE), without an intermediary device (which reduces agency [2]). We aim to determine which features should be tracked using low-cost cameras or wearable sensors, in order to detect the real-time point of release (*PoR*) of the ball. Previous work has mainly focused on offline analysis of real throwing for performance feedback, whereas our approach aims to detect virtual throwing events in real-time. We introduce a metric that prioritizes detection timing to evaluate these models.

We capture the motion of 18 participants throwing a virtual ball in an immersive VR game, using two different types of tracking: i) *real-time* arm and finger tracking to control VR interactions and determine the PoR; and ii) simultaneous capture of full-body motion for later *offline analysis*. Multiple frame-level binary classifiers are trained to detect the PoR of the ball based on different combinations of the full-body motion features (rotation, position, linear and rotational velocity) and the arm joints (wrist, elbow and shoulder). Our results show that the wrist joint and the rotation motion feature provide the highest accuracy for detecting the PoR.

*yamacg@tcd.ie
†n.mitra@ucl.ac.uk
‡carol.osullivan@tcd.ie

2 Related Work

Human action/activity recognition (HAR) is a very active research topic in Ubiquitous Computing and HCI, with applications in many areas such as healthcare monitoring [3], smart environments [1], security surveillance [8] and offline classification of events in sports for performance assessment [7]. In comparison to video-based methods for HAR [5], wearable sensor-based approaches to HAR are more resilient and flexible and avoid occlusion, lighting and privacy concerns [4].

In recent years, previous pattern recognition approaches that use handcrafted features in HAR have been replaced with hierarchical feature extraction for Deep Learning [11]. This enables the adoption of diverse network types, *e.g.*, recurrent neural networks, long short-term memory networks, stacked autoencoders, and the learning of more advanced features to allow successful recognition of more complex human activities. Our approach is similar to Vepakomma et al. [9], where a framework utilizing multiple wearable and ambient sensors is used to recognize complex daily life activities. Following a sliding window sampling, they extract various statistical features and train a supervised neural network.

3 Throw Detection

Our dataset consists of motion data and release timecodes of virtual throws. A frame-level binary classifier is trained, that uses a sliding window analysis to detect the PoR frame based on the estimated probabilities inferred from four motion features input: Position (P), Velocity (V), Rotation (R) and Rotational Velocity (RV). Features are extracted from the throwing arm's joints: wrist (W), elbow (E) and shoulder (S). The classifier is supervised based on PoR timecodes.

Capturing Throws: The Virtual Environment (VE) is displayed with the HTC Vive™ headset in Unity. A 5cm diameter ball is placed on a block and a 50cm diameter target is on the ground (see Figure 1). All target positions are generated randomly at real-time for 18 participants (10F, 8M, aged 18-38), recruited from University students and staff. Each participant performed 63 virtual throws, giving an overall total of 1134 throws for all participants.

We implemented two tracking types: i) *real-time* capture using a HTC Vive controller attached to the throwing forearm for arm tracking and ManusVR™ gloves for finger tracking (to allow the virtual ball to be picked up and released in real-time); and ii) full-body motion capture at $120Hz$ for *offline analysis* with a 21-camera Vicon™ optical motion capture system and 53 body markers (see Figure 1). A timecode sent from the motion capture system at release time was used for synchronization. Between the grab and release events, a velocity estimation algorithm continuously calculated the average velocity over a window of the nine previous frames, which was applied to the ball at the frame it was released. For the release algorithm, we used the rotational rate of change in the index and middle finger phalanges, which provides a good indication of an opening hand. The associated timecode is used as the ground truth PoR for our motion analysis.

Data Preprocessing: To extract throwing motion sequences, we center a window of 51 frames at the PoR frame of each of the 1134 throws, as 51 frames (425 ms) includes the most relevant phases of a throwing motion. Global position is converted to relative position by changing the origin of the coordinate system to be the hip center of the participant. We also calculate the position of the arm joints projected onto the direction the thrower is facing.

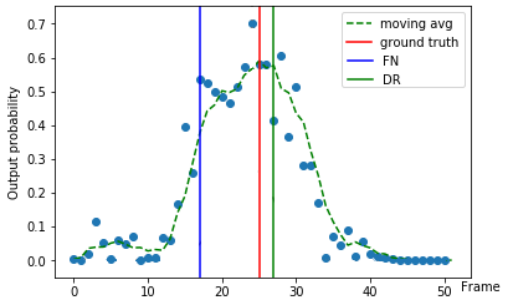


Figure 2: Illustration of FN and DR methods. Scattered points represent the estimated probabilities of a frame representing a PoR.

Velocity V is estimated using the finite-difference method on positions. Similarly, Rotational Velocity RV is estimated using the finite-difference method on the angle-axis representation of the rotation data. For every frame, the joint motion features of 6 previous frames, up to and including the current one, are extracted and sequenced in an array to form a *sliding window* of size w. Finally, for all throws, a binary vector (the *target* vector) of size W is also defined, where the element corresponding to the ground truth PoR frame (*i.e.*, the center frame) is set to *True* and all others are set to *False*. The skewness of the dataset is handled by using a weighted loss during training.

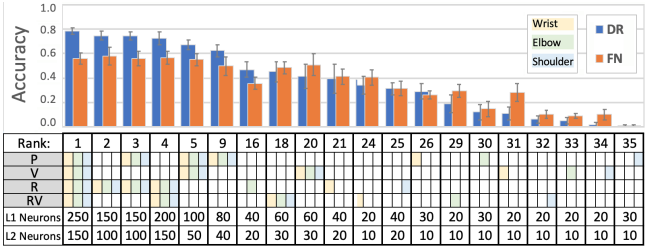
Model Training: The input to our model is defined by a selection of motion features per joint. Out of the 4 motion features available for each of the 3 joints, any combination of these joint-feature pairs can be selected, *e.g.*, $\{V,RV\}$ for W and $\{P\}$ for E. Each of the motion features contribute to the size of the input by their dimensions: P is three-dimensional, R is four-dimensional, and both V and RV are one-dimensional. The size of the model input is calculated by summing the dimensions of motion features used per joint, multiplied with the sliding window size w, *e.g.*, in the selection of $\{V,RV\}$ for W and $\{P\}$ for E, $w*((1+1)+(3))$. We refer to each selection as a joint-feature configuration. We train 35 models, each with different joint-feature configurations, consisting of stacked, fully connected feed-forward neural network layers with a sigmoid layer at the output and binary cross-entropy as the loss function. Rectified linear unit (ReLU) activation is used, and each layer is followed by batch normalization. The models are developed using PyTorch [6].

Detection Methods: We use two methods to estimate the PoR frame in a throw window from the model's output probabilities (see Figure 2). In our first method, *First Nonzero* (FN), we start classifying each frame in the throw window starting with f_{t-25} and detect a release at the first estimated PoR frame. We use a classification threshold of $p = 0.5$. Our second method, *Delayed Response* (DR), involves applying a moving-average filter and checking the rate of change of the output probabilities of the model. Similar to FN, we apply a threshold to avoid the algorithm being prone to noise.

Model Evaluation: Commonly used binary classification metrics do not provide enough insights into the capabilities of a time series classification model. We therefore introduce a metric, *within*, which uses the size of the time mismatch between the actual and estimated PoR frames to calculate accuracy, *i.e.*, $within(d)$ measures the ratio of how many of the detections fall within a distance of d frames:

$$within(d) = \frac{1}{100}\sum_{i=1}^{100} \mathbb{1}(|arg(\hat{Y}_i = 1) - arg(Y_i = 1)| \leq d) \qquad (1)$$

where Y_i represents the target vector for one throw, $\hat{Y}_i$ represents the predicted target vector, $\mathbb{1}(.)$ is the indicator function and 100 is the size of the test set; d is the distance in frames from the ground truth. This metric thus assesses a model's ability to cluster the estimated PoR frames around the real PoR frame, whether they are correctly classified or not.



Rank:	1	2	3	4	5	9	16	18	20	21	24	25	26	29	30	31	32	33	34	35
P																				
V																				
R																				
RV																				
L1 Neurons	250	150	150	200	100	80	40	60	60	40	20	40	30	20	30	20	20	20	20	30
L2 Neurons	150	100	100	150	50	40	20	30	30	20	10	20	10	10	10	10	10	10	10	10

Figure 3: Selection of joint-feature combinations and within-5 results of the FN and DR methods, ranked by DR accuracy

4 Results and Discussion

The trained models were tested by randomly subsampling the full set of throws into 10 subsets of size 100 each. For both methods, FN and DR, we performed a semi-exhaustive analysis over a group of joint-feature configurations to rank motion features and joints, based on their performance for frame-level throw classification. Of the 35 configurations we have evaluated overall, Figure 3 shows a selection of 20, color-coded and ranked based on their $within(5)$ accuracy for the DR method. Their original rank, and the number of neurons on layers 1 (L1) and 2 (L2) of the trained model are also shown.

Accuracy over 70% for a single motion feature was only achieved for R (rank 2). Configurations using only one motion feature for a single joint show that R provides higher detection accuracy in general, *e.g.*, ranks 16,21,25, but not 24. Configurations that use a single motion feature on a single joint also indicate that the wrist joint provides higher accuracy (*e.g.*, ranks 26,30,35 for P; 31,33,34 for V). This is intuitive since the wrist is positioned close to the ball and it follows a very close motion pattern as the ball. We conclude that: i) Rotation provides the highest accuracy as a single motion feature; and ii) Wrist joint provides the highest accuracy if used alone. In future work we will run a user-study to explore how throws with inaccurate release timings (*i.e.*, too early or late) are perceived, which will facilitate the interpretation and utility of our results.

Acknowledgments

This research was supported by Science Foundation Ireland, Grant Agreement 13/RC/2106, at the SFI ADAPT Research Centre, TCD.

References

[1] K. Avgerinakis, A. Briassouli, and I. Kompatsiaris. Recognition of activities of daily living for smart home environments. In *Intelligent Environments*, pp. 173–180, 2013.

[2] D. Coyle, J. Moore, et al. I did that! Measuring users' experience of agency in their own actions. In *ACM CHI*, pp. 2025–2034, 2012.

[3] J. Hoey, T. Plötz, et al. Rapid specification and automated generation of prompting systems to assist people with dementia. *Pervasive and Mobile Computing*, 7(3):299–318, 2011.

[4] M. Janidarmian, A. Roshan-Fekr, et al. A comprehensive analysis on wearable acceleration sensors in human activity recognition. *Sensors*, 17(3):529, 2017.

[5] S. Ke, H. Thuc, et al. A review on video-based human activity recognition. *Computers*, 2(2):88–131, 2013.

[6] A. Paszke et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. 2019.

[7] D. Schuldhaus, C. Zwick, et al. Inertial sensor-based approach for shot/pass classification during a soccer match. In *KDD Workshop on Large-Scale Sports Analytics*, pp. 1–4, 2015.

[8] D. Singh and C. K. Mohan. Graph formulation of video activities for abnormal activity recognition. *Pattern Recognition*, 65:265–272, 2017.

[9] P. Vepakomma, D. De, et al. A-wristocracy: Deep learning on wrist-worn sensing for recognition of user complex activities. In *IEEE Wearable and implantable body sensor networks (BSN)*, pp. 1–6, 2015.

[10] A. Villegas, P. Perez, et al. Realistic training in vr using physical manipulation. In *IEEE VR Workshop (VRW)*, pp. 109–118, 2020.

[11] J. Wang, Y. Chen, et al. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 119:3–11, 2019.