Figure 1: Some images of our system in action. The character learns to block strikes from several directions from motion capture data with no additional animation logic.

Towards Data-Driven Sword Fighting Experiences in VR

Javier Dehesa
University of Bath
Bath, UK
J.Dehesa@bath.ac.uk

Andrew Vidler
Ninja Theory Ltd
Cambridge, UK
andrew.vidler@ninjatheory.com

Christof Lutteroth
University of Bath
Bath, UK
C.Lutteroth@bath.ac.uk

Julian Padget
University of Bath
Bath, UK
J.A.Padget@bath.ac.uk

ABSTRACT

We present a data-driven animated character capable of blocking attacks from a user in a VR sword fighting experience. The system uses motion capture data and a machine learning model to recreate a believable blocking behaviour, suggesting the viability of full-featured data-driven interactive characters in VR. Our work is part of a larger vision of VR interaction as a two-level problem, separating spatial details from design concerns. In this context, here we provide the designers of the experience with a character from which a "blocking" behaviour can be requested without further spatial specifications. This puts down a first building block in the construction of a controllable data-driven VR sword fighter capable of multiple behaviours.

CHI'19 Extended Abstracts, May 4–9, 2019, Glasgow, Scotland Uk

ACM ISBN 978-1-4503-5971-9/19/05.
https://doi.org/10.1145/3290607.3312930

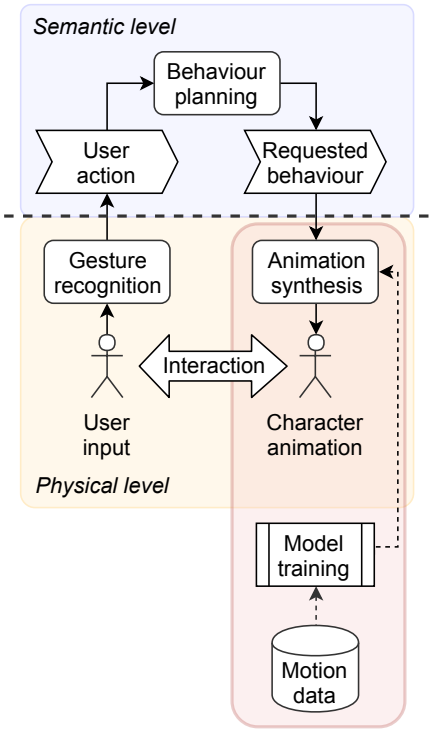

Figure 2: Overall data-driven interaction diagram. Dotted lines represent offline flow and solid lines online flow. At the physical level (orange), data-driven models are learned from gesture and motion data. At runtime, these are used to transition into and out of the semantic level (blue), where the experience designer decides how to react to the gestures. Here we focus on the animation synthesis aspect of the system (red).

KEYWORDS

Virtual reality, machine learning, animation

INTRODUCTION

Virtual reality is one of the most powerful media for delivering interactive experiences, yet there are still many kinds of interaction mechanics that are proving challenging to incorporate. Sword fighting is a representative example of an appealing scenario that, although sophisticatedly recreated in screen-based interactive media, is nearly infeasible to reproduce in VR with existing techniques. The main reason for this is that close-range interaction in VR is difficult to handle with precision, due to the variability and unpredictability of the user input, and sword fighting requires a fair amount of precision, even at a basic level. Additionally, sword fighting is itself more open in nature than other interactions, such as shaking hands, where established methods like inverse kinematics could be used successfully.

In our view, designers should use a similar language whether they are working on screen-based or VR media, in spite of the additional complexities of the latter. Figure 2 represents our vision of a fully data-driven interaction framework for VR. Here the *semantic level* corresponds to the usual design space, where the desired behaviour can be programmed in high-level terms such as "if the user attacks from the left, block". In order to get to this point, the user input and the animation data are processed at the *physical level*, using respectively a gesture recognition model, to parse the input as discrete actions, and an animation synthesis model, to realise the requested behaviour in concrete animations. Importantly, in this approach the physical level is implemented using data-driven techniques, while the semantic level is based on knowledge-driven logic, providing a balance between automation and transparency.

This paper introduces our work in animation synthesis for sword fighting in VR. We present a sword-wielding character capable of blocking sword attacks from a user in a VR simulation. This demonstrates the feasibility of implementing data-driven behaviours in interactive VR characters, and provides a foundation for a fully-featured sword-fighting agent. The character anticipates the motion of the user and puts the sword in a position that blocks the attack trajectory before being hit, using a data-driven model that learns from previously collected sword-fighting motion captured animation data. Our system is implemented using a standard commercial VR kit, featuring head and hands tracking.

While it could seem that this could be solved analytically, for example by extrapolating the user's sword trajectory and using inverse kinematics on the character to match the sword position, this soon proves unsuccessful. The variability in the input is too high to make reliable estimations and the lack of real weight in the controller results in acceleration spikes that are difficult to model. In

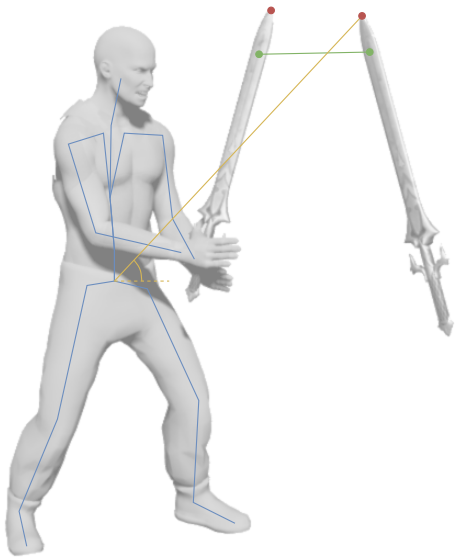

Figure 3: Each input example contains the location, orientation and velocity of the joints (blue), recent positions of the sword tips (red), distance and closest points between the swords (green) and view angles from the centre of the character to the user's sword tip (yellow).

addition to that, learning from motion data allows the model to capture specific traits of the style performed by the actor.

Note that, obviously, the hand of the user cannot be really blocked with current VR hardware, so a certain level of "cooperation" from the user is required. While different techniques exist to improve the user experience in this sense (such as haptic feedback and physics simulation), these fall outside the scope of our work, at present.

RELATED WORK

Much work has been done in the HCI community in recreating faithful interactions between humans and virtual characters. DeVault et al. [5] and Sagar [13] have showcased examples of highly sophisticated systems, capable of human abilities such as gaze detection or imitation. However, there has been less focus on close-range "physical" interactions in virtual worlds. In our view, a solution to this problem should combine (and adapt) techniques from different fields. With respect to the user input, good gesture recognition techniques have existed for a long time now [15], and machine learning has proven very successful in this field [2]. There exist as well several models on which behaviour planning can be based, from simple rule-based reasoning to more advanced architectures like belief-desire-intention (BDI) [12].

Here we look at the animation of the interactive characters. Data-driven animation is a difficult but growing field. The introduction of the motion graph structure [9], which builds animations from a network of short clips, gave raise to several systems based on or inspired by it [7, 10, 14], some of which have been implemented in commercial applications [3, 4]. These systems would be difficult to adapt to interactive VR, though, as they are not tailored to the high variability in user input discussed above. However, recent results in locomotion synthesis [8, 16] have shown that motion capture can be learned to a high level of detail through machine learning models capable of generalising to new situations.

METHODOLOGY

Our starting point is the data that will be used to train the system. We used Vicon Bonita motion capture equipment to record several sword fighting sequences featuring attacking and blocking actions at different angles. In total, nearly 15 minutes worth of training material were produced, recorded at 30 frames per second (over 26,000 frames). Each captured frame contains the position of 24 skeletal body joints of the blocking actor (corresponding to the virtual character), plus the position of the tip of both swords.

The data is preprocessed in order to build the training database of examples. Each frame is transformed into an example consisting of an input and an output vector. The input features, represented in Fig. 3, include: the location of the joints; the velocity of the joints (difference with respect to the

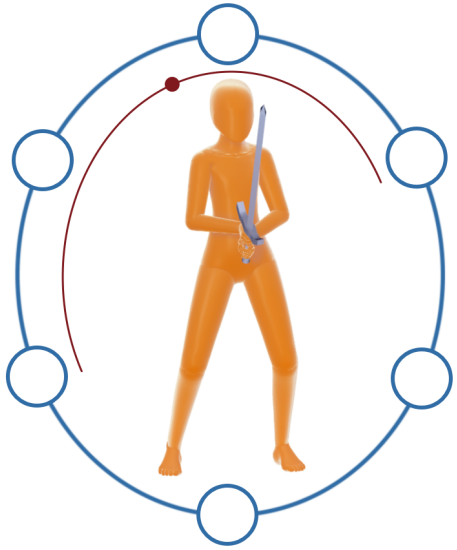

Figure 4: Each set of parameters, represented by the blue circles, is associated with different strike directions around the character. The tip of the user's sword determines the current direction (red dot in the figure), and the four closest sets of parameters (red arc) define a cubic spline from which the weights are interpolated.

previous frame); the orientation of the joints, given as their local X and Y unit vectors; the position of the tips of both swords in the current frame and three and six frames before, which implicitly express a measure of their velocity and acceleration; the distance and closest points between the swords; and the yaw and pitch angles from the character's centre to the user's sword tip. The output vector contains the location, velocity and orientation (as a pair of vectors) of the joints in the next frame, as well as the location and orientation of the character's sword tip (the location of the sword hilt is not necessary as it can be derived from these). All positions are relative to the character root (middle point between the feet).

The data is then used to train a neural network using TensorFlow [1]. A regular feed-forward network did not produce satisfactory results, so we developed a model that takes into account the direction of the incoming strikes, based on the idea of phase-functioned neural networks [8]. The basic architecture of the model is made of three dense layers. However, since the behaviour of the defender varies significantly with the direction of the incoming strike, we create multiple sets of network weights for these layers, each one associated with a particular direction, as shown in Fig. 4. Using these, the particular weights to be used for a given arbitrary direction is computed as a cubic spline interpolation of the four closest sets of weights. Using a cubic spline guarantees that the mapping from strike directions to network weights is continuous and smooth. At runtime, the tip of the user's sword is used to estimate the strike direction, so attacks coming from left or right will use different network weights more specialised in their corresponding reactions. This results in better character reactions than a bigger neural network with a single set of weights, which we attribute to the reduced complexity of the patterns associated to each set of weights.

The trained network is embedded into a VR scenario built in Unreal Engine 4 [6] for an Oculus Rift VR kit [11]. The only additional post-processing applied to the animation produced by the model consists in correcting the position of the character's sword tip to ensure the hilt of the sword is always within the hands. This prevents sporadic issues where errors in the network output cause the sword to "fly" away from the hand momentarily.

RESULTS & DISCUSSION

Figure 1 shows some images of our results. The character is capable of recreating blocking reactions adapted to the strikes of the user, using different kinds of movements. Blocks are performed similarly to how a real sword fighter would do them, since they are derived from the acted motion capture data, and otherwise the character remains in a credible "waiting" pose whenever the user is not attacking. All of this is achieved with no additional animation work (besides the motion capture session) or hand-crafted logic. This is a significant simplification in the development process of interactive VR scenarios.

There are still notable limitations in the system. On the one hand, since everything is learnt from motion capture data, actions that are very different from the training data can produce unstable results. This can be mitigated both by collecting more motion capture data and by augmenting the existing examples, applying alterations to the speed or the trajectory of the player's sword. Also, due to the architecture of our model, attacks directed to the centre of the character (such as a frontal stab) are not handled well, since there is no specific subset of parameters dedicated to it (see Fig. 4, in which there is no circle in the middle). We expect to improve in this regard with a more advanced model that allows for subsets of parameters distributed along more than one dimension. Finally, so far we have only modelled a character standing in a fixed position. It should be possible to model stepping and pivoting too, but the extension is not straightforward. Since the position of the character determines the coordinate system, it is necessary to maintain a stable estimation of its location and orientation.

FUTURE WORK

Our work shows that complex interactive VR scenarios like sword fighting can be modelled through a data-driven process, using machine learning to overcome the unpredictability and variability of the input. In future iterations of this work we plan to develop critical metrics assessing both the similarity between the generated behaviour and the motion capture data, as well as the quality perceived by users of the system. Code and data will be released for public evaluation.

Having implemented a blocking behaviour, the natural continuation would be to implement additional abilities, such as attacking and evading, and the capability of switching between each of them. We believe these scenarios are broadly similar and thus can be learned by building on the techniques outlined here.

In order to implement the full proposed interaction framework (Fig. 2), the remaining elements are now under development. Gesture recognition on VR input may also be implemented effectively using machine learning (within the limited possibilities of the hardware). The behaviour planning component can be prototyped as a set of rules mapping recognised actions to requested behaviours, before moving on to more advanced agent architectures. The value of the design, however, lies in how these three elements interact and cooperate to combine data-driven techniques with already existing design knowledge to build a complete VR sword fighting character.

ACKNOWLEDGEMENTS

This work was funded and supported by the Centre for Digital Entertainment at the University of Bath (EPSRC award EP/L016540/1) and Ninja Theory Ltd. Demonstration environment and sword model by Epic Games, Inc.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/.

[2] Maryam Asadi-Aghbolaghi, Albert Clapés, Marco Bellantonio, Hugo Jair Escalante, Víctor Ponce-López, Xavier Baró, Isabelle Guyon, Shohreh Kasaei, and Sergio Escalera. 2017. Deep Learning for Action and Gesture Recognition in Image Sequences: A Survey. In *Gesture Recognition*. Springer, 539–578.

[3] Michael Büttner. 2013. Reinforcement Learning Based Character Locomotion in Hitman: Absolution. In *Game Developers Conference 2013 (GDC 2013)*.

[4] Simon Clavet. 2016. Motion Matching and The Road to Next-Gen Animation. In *Game Developers Conference 2016 (GDC 2016)*.

[5] David DeVault, Ron Artstein, Grace Benn, Teresa Dey, Ed Fast, Alesia Gainer, Kallirroi Georgila, Jon Gratch, Arno Hartholt, Margaux Lhommet, Gale Lucas, Stacy Marsella, Fabrizio Morbini, Angela Nazarian, Stefan Scherer, Giota Stratou, Apar Suri, David Traum, Rachel Wood, Yuyu Xu, Albert Rizzo, and Louis-Philippe Morency. 2014. SimSensei Kiosk: A Virtual Human Interviewer for Healthcare Decision Support. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '14)*. International Foundation for Autonomous Agents and Multiagent Systems, 1061–1068.

[6] Epic Games, Inc. 2016. Unreal Engine 4.13. https://www.unrealengine.com.

[7] Rachel Heck and Michael Gleicher. 2007. Parametric Motion Graphs. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games (I3D '07)*. ACM, 129–136.

[8] Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-Functioned Neural Networks for Character Control. *ACM Transactions on Graphics* 36, 4 (2017), 42:1–42:13.

[9] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2002. Motion Graphs. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '02)*. ACM, 473–482.

[10] Yongjoon Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popović, and Zoran Popović. 2010. Motion Fields for Interactive Character Locomotion. In *ACM SIGGRAPH Asia 2010 Papers (SIGGRAPH ASIA '10)*. ACM, 138:1–138:8.

[11] Oculus VR. 2016. Oculus Rift. https://www.oculus.com/rift.

[12] Anand S. Rao and Michael P. Georgeff. 1991. Modeling Rational Agents within a BDI-Architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*. Morgan Kaufmann, 473–484.

[13] Mark Sagar. 2015. BabyX. In *ACM SIGGRAPH 2015 Computer Animation Festival (SIGGRAPH '15)*. ACM, 184–184.

[14] Adrien Treuille, Yongjoon Lee, and Zoran Popović. 2007. Near-Optimal Character Animation with Continuous Control. *ACM Transactions on Graphics* 36, 3 (2007), 7:1–7:7.

[15] Junji Yamato, Jun Ohya, and Kenichiro Ishii. 1992. Recognizing Human Action in Time-Sequential Images Using Hidden Markov Model. In *Proceedings of the 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '92)*. IEEE, 379–385.

[16] He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-Adaptive Neural Networks for Quadruped Motion Control. *ACM Transactions on Graphics* 37, 4 (2018), 145:1–145:11.