D6.5 Animation graph traversal optimisation

Grant Agreement nr	780470
Project acronym	SAUCE
Project start date (duration)	January 1st 2018 (36 months)
Document due:	29th February 2020 (delay accepted)
Actual delivery date	27th February 2020
Leader	IK + UPF
Reply to	josep.blat@upf.edu
Document status	Submission Version

Project funded by H2020 from the European Commission

Project ref. no.	780470
Project acronym	SAUCE
Project full title	Smart Assets for re-Use in Creative Environments
Document name	D6.5 Animation graph traversal optimisation
Security (distribution level)	PU
Contractual date of delivery	29th February 2020
Actual date of delivery	27th February 2020
Deliverable name	Animation graph traversal optimisation
Type	DEM
Status & version	Submission Version
Number of pages	10
WP / Task responsible	IK-UPF
Other contributors	IK - UPF
Author(s)	Simon Woeginger, Hermann Plass, Josep Blat
EC Project Officer	Ms. Adelina Cornelia DINU - Adelina-Cornelia.DINU@ec.europa.eu
Abstract	This deliverable is part of the reporting required for WP6 and details the work carried out within WP6 Task 3 - Time, space and world-awareness approach for animation synthesis. This internal report accompanies the demonstrator of the work, which is publicly available at GitHub
Keywords	Machine learning, smart assets, character motion synthesis, PHS motion library,
Sent to peer reviewer	Yes
Peer review completed	Yes
Circulated to partners	No
Read by partners	No
Mgt. Board approval	No

Document History

Version and date	Reason for Change
1.0 03-02-2020	Document created by Josep Blat
1.1 17-02-2020	Version for internal review
1.2 27-02-2020	Final version for submission

Table of Contents

1 EXECUTIVE SUMMARY

This deliverable is part of the reporting required for Workpackage 6 (WP6). This report details the work carried out within WP6 Task 3 *Time, space and world-awareness approach for animation synthesis* carried out during months 7 to 26 (after the extension requested) of the SAUCE project. This internal report accompanies the demonstrator of the work, which is publicly available at GitHub: https://github.com/upf-gti/Machine-Learning-for-Character-Animation. A demonstration video is linked from there (https://youtu.be/Uo6D6h_tlaA).

Within WP6T3. a machine learning model was developed, to be used for character motion synthesis given a high level user request. This is a step forward towards meeting the goals of developing tools and methods for smart asset generation, which are outlined within SAUCE WP6.

Section 1 begins with an introduction to the current cutting-edge machine learning model that the developments of WP6T3 built upon. Section 2 serves as an introduction to the system developed and it includes a brief discussion on the developed system and its limitations. Section 3 describes the approaches to source and produce animation data that could be used as training data for the machine learning model, including tooling developed previously in SAUCE and data supplied by partner organizations. Section 4 illustrates the developed system as a block diagram and provides details on the developments made during WP6T3. Section 5 details preliminary results in testing the system along with a discussion on the applicable use cases.

2 BACKGROUND

The neural network developed during WP63T3 is based on the PhD research project by Sebastian Starke at the University of Edinburgh. This research addresses quadruped motion control for locomotion on a flat surface using a *Mode-Adaptive Neural Network* (MANN)[1].

The MANN is a recursive (feedback) neural network using the 'mixture of experts' concept. It consists of two parts: a main, *motion prediction*, network and an auxiliary, gating, network. The velocity of the feet joints of a given quadruped character along with the desired character direction and velocity are used as input to the gating network. The motion prediction network takes the character trajectory data and the current pose of the figure as input.

The input to the motion prediction network is split into 'future' and 'past' parts. The 'future' data is generated and manipulated during runtime and is used by the network to generate an animation that follows the trajectory. The 'past' portion of the pose provides context which is used to allow the network to produce consistent expected motion.

The output of the network is an updated character pose in the current frame which results in a new location and a new orientation for the whole character.

The present work also used parts of the source code made available from the above research work at Github[2].

[1] The preliminary results of this research were published in *ACM Transactions on Graphics* 37(4), 2018 (*Proceedings of SIGGRAPH 2018*), available online at http://homepages.inf.ed.ac.uk/tkomura/dog.pdf. See exact reference at the end of the report.
[2] https://github.com/sebastianstarke/AI4Animation

3 INTRODUCTION

The overarching goal of WP6 is the development of methodologies and tooling that can be used in the generation of smart assets, as well as in unlocking existing assets in order to enable their re-use in scenarios they were not necessarily designed for.

To further these goals, within WP6T3 a machine learning system was developed on the basis of current cutting-edge approaches as indicated in the previous section. The developments and progress made are detailed in the following section.

The developed system is used to automatically generate character animation given high-level user demands. This forgoes the need for hand authoring animation assets or complex state machine development to control the locomotive state of a given virtual character. This system has been integrated and tested within the *Unity*[3] game engine, as this is a popular engine choice that is widely used across the creative media industry.

3.1 Limitations

The quality of the character animation generated by the system developed heavily depends on the data used during training, as it was experimentally found. The following features of the training data were found to have a high impact on the procedurally generated animation at runtime:

- Fidelity of the animation data used to train the system
- Range of motions included in the training data

The first limiting factor is concerned with the quality of the training animation data. If the training data is low quality this will be reflected in the output of the system.

The second limiting factor of the machine learned model is the range of motions included in the training data. If a user demand is issued and the expected motion is not included in the training data, then the model will not be able to synthesize this motion.

With these two considerations in mind, the training data used must be high quality and include the types of motion expected given the possible high-level user demands. These limiting factors must be taken into consideration when sourcing and generating animation data that will be used to train the developed model.

The next section discusses the methods used to obtain animation data that were suitable for training purposes.

4 TRAINING THE SYSTEM

Two sources for obtaining animation data for the purposes of training were explored:

- Generating animation assets in-house using the tool developed in WP6T2 by IK
- The PHS motion library supplied by partner Filmakademie within SAUCE

The first source of animation data that was investigated is the tool developed in WP6T2. This tool is a plugin of *Autodesk Maya*[4] that is used to automatically generate long composite animation sequences from discrete animation clips.

To give a high-level overview of the tool developed in WP6T2, the workflow is as follows:

1. The user imports a bank of animation assets.
2. The user can specify the animations that should be present in the final composite animation and their ordering.

[3] https://unity.com/
[4] https://www.autodesk.com/products/maya/overview

3. The plugin will then process the animations to generate the composite animation. In-between animations that transition from one animation to another are generated and included in the composite animation.

If the user has access to an animation library of varied, high quality animation clips, this method of generating training data for the neural network addresses the limitations discussed in section 3.1.

In terms of the quality of the animation generated, the automatic generation of the in-betweens in the final composite clip helps to make sure the transitions from any animation to any other animation are smooth.

The ability to select any number of discrete clips and their ordering in the final composite clip caters for generating sets of composite animation clips that contain varied motions.

The second source of animation data used for training the model was the *PHS motion library* created and supplied by partner Filmakademie[5]. This library of motion capture data includes a comprehensive representation of typical biped locomotion and as such it was well suited to train the neural network. In practice, this data was used for training the developed neural network. The resulting trained neural network was able to handle a range of locomotive behaviors that exhibited nuanced and realistic motion.

5 SYSTEM OVERVIEW

The following figure illustrates the operation of the developed neural network as a block diagram.

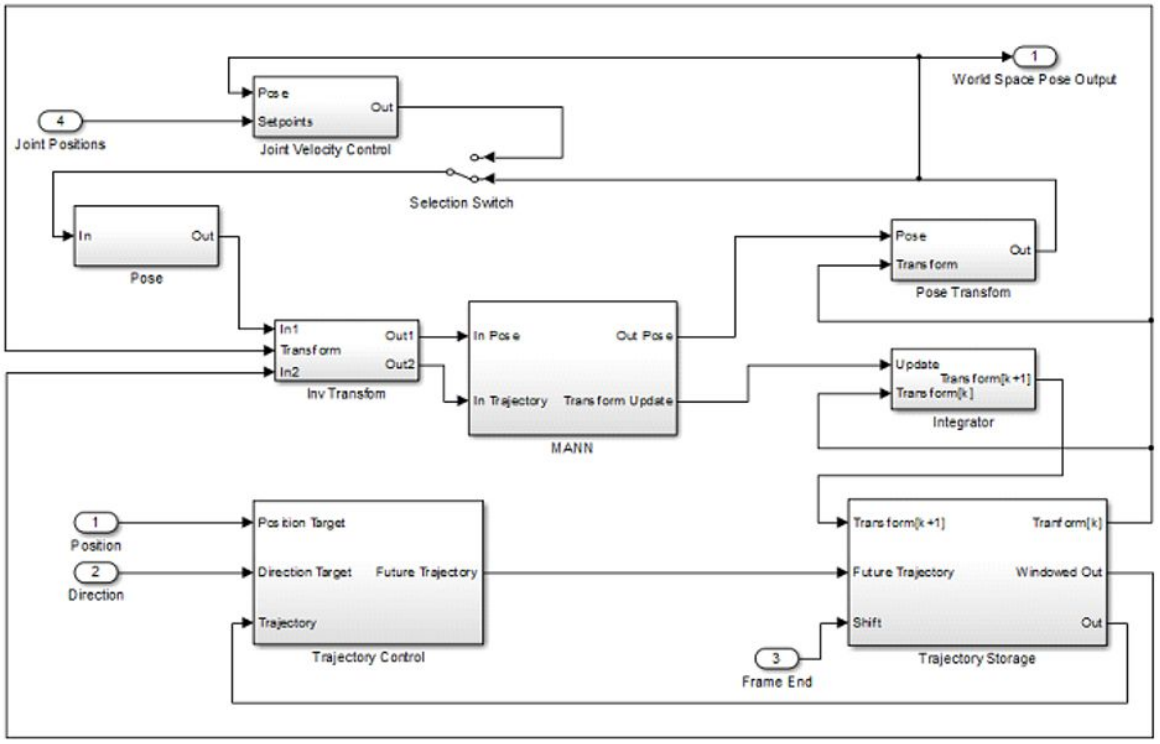

Figure 1: System Overview

[5] The PHS Motion Library is available online for download: https://www.sauceproject.eu/Downloads

5.1 Adaptation of the MANN Solution

The following paragraphs provide details about the work and developments on top of the existing MANN, whose overview was given in section 3.

1. In the previously existing solution, the MANN was applied to human figures, although the original research stated that it was only for quadrupeds. The developed system in WP6T3 has been modified to work well with biped characters
2. The original solution only used velocities of the feet (joints in contact with the floor) to feed to the gating network. The present solution also uses velocities of other joints for this purpose. The optimal selection, found by way of experiment, seems to be the feet, the hands, the root bone and the head. In fact, all joint velocities could be used to help the gating network set up the experts, but the above selection seems to be a reasonable compromise between computation cost and its added value. If only the velocities of the two feet are used to set up the experts, some nuances of the upper body motion (such as the effect of a swinging arm onto the next step) could be lost in training and the network could output a static pose (the figure would freeze), instead of motion as it would be expected by the user. That is, more input to the gating network helps to disambiguate the next pose and to keep the figure in motion. By appropriately choosing joint velocities relevant for the gating network, user-supplied action labels can be removed from its input. In the present solution, action labels have not been used.
3. The original solution (at least the one that was made public) used PI control of the figure speed (magnitude of horizontal velocity) combined with some low-pass filtering to propagate the horizontal velocity and the forward (facing) vector of the figure based on user keyboard or gamepad input. The target velocity and the target direction that resulted from this were then used to especially shape the future position and the future velocity profiles to make the figure move. The magnitude control made it difficult to slow down the figure and to make it move back, while the special shaping of the future trajectories introduced some unnecessary lag into the system that the PI control had to deal with. The present solution uses two-dimensional PI velocity control. It calculates a target velocity in 2D based on the desired 2D position of the figure in the new frame. There is no filtering of the PI control output, the desired speed of response is set by choosing the PI parameters appropriately. The future position trajectory is generated directly from the PI velocity output using linear extrapolation, while the velocity is set equal to that in all the future samples. It might seem that having several future samples containing the same data is inefficient. But it was found that the network performs better when the number of the past and the number of the present + future samples are the same, thus offering the network a certain time-based symmetry. When the number of the future samples is reduced, the runtime performance deteriorates. Figure orientation (the forward-facing direction) is not explicitly controlled in the present solution. The desired direction that is passed from the user is just propagated to all the future samples, leaving the rest to the network.
4. In the original solution, the especially-shaped future velocity and position profiles could also be optionally blended with the network-predicted future trajectory (comprising future position, direction and velocity). However, this kind of blending is a source of disturbance to the main PI control, which is why trajectory prediction by the network has been excluded from the present solution. The future trajectory was completely removed from the network output, which freed up resources and simplified the task of the network. The training also showed a slightly better residual error for the same number of epochs. However, in this setup the user must supply the desired orientation of the figure to be written to the future samples. The user normally has this information readily available from the desired trajectory (e.g., as the gradient of a spline curve).
5. The developed solution also introduces secondary control of body posture in terms of individual joint trajectories (Joint Velocity Control in Figure 3). This is based on modifying joint velocities contained within the Pose using PI control to follow user-supplied joint trajectories. The effect of this new feature was quite limited, however.

5.2 Modified Motion Control System

The modified motion control system was shown above in Fig. 1. It includes an optional Joint Velocity Control block that can change selected joint velocities contained within the pose to more closely match the desired joint trajectories. The MANN block no longer generates a trajectory prediction. Trajectory samples no longer contain action labels.

Fig. 2 below shows the final implementation of the Trajectory Control block. The desired velocity is estimated from the last two trajectory positions and the new position target. Then a velocity update is computed using a control technique analogous to the PI velocity algorithm for position, only the velocity itself is the manipulated variable here. The computed update is added to the current velocity to obtain the target velocity. Using the velocity algorithm avoids integrator wind-up problems as there is no error integration as such.

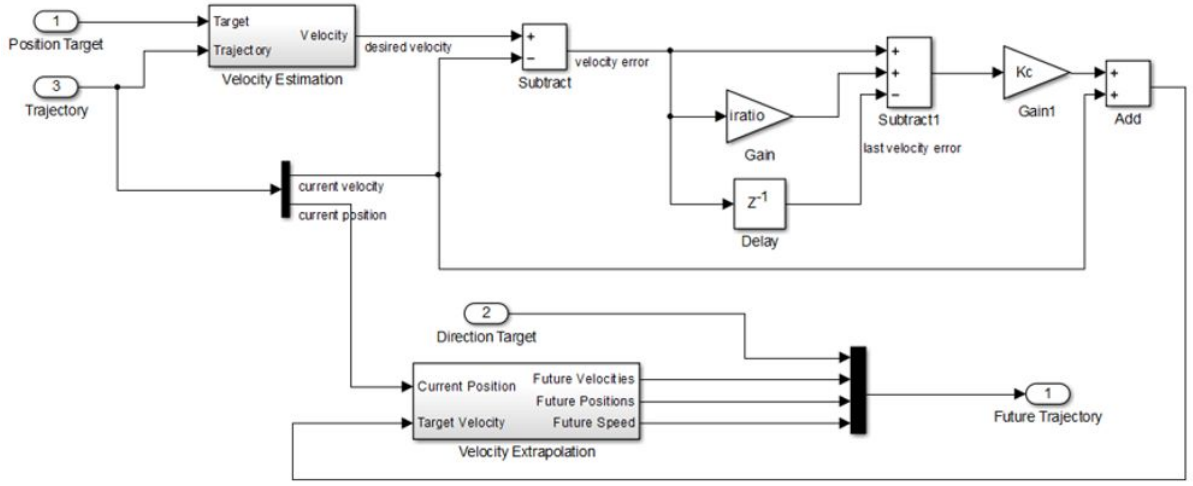

Figure 2: Final implementation of the Trajectory Control block using PI velocity control and velocity-based extrapolation from the current sample to generate future trajectory

The Velocity Extrapolation block generates the future trajectory samples based on the current position and the computed velocity target. The speed parameter is derived from the target velocity (just using its magnitude multiplied with the sampling time, the gain mismatch with respect to training is compensated by feedback control). The present and all the future trajectory samples receive the same values of velocity, speed and the user-supplied direction (or the last direction, if not supplied).

Fig. 3 below shows the procedure used by the optional Joint Velocity Control block to modify joints velocities to match their desired position. This is PI velocity algorithm for position control, with the position update being converted to an equivalent velocity update that is added to the current velocity.

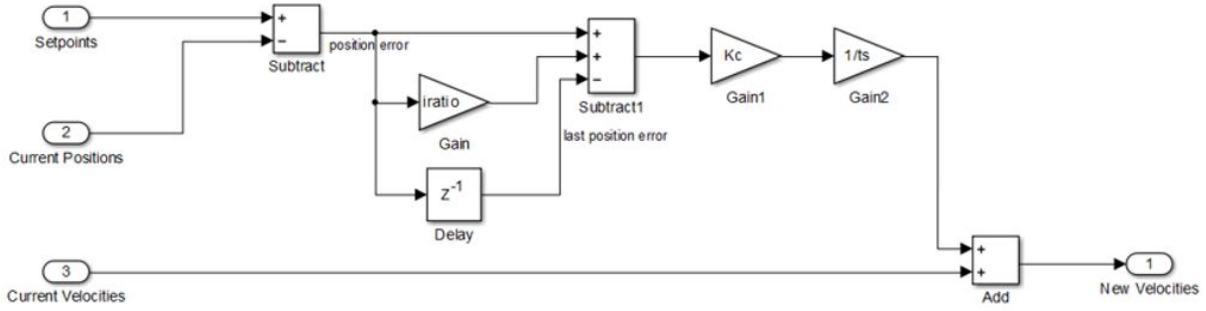

Figure 3: Modifying joint velocities inside the Joint Velocity Control block

6 RESULTS

The animation produced by the machine learning system is procedurally generated at runtime. Issuing the high-level user demand can either take place offline or online during scene play back, depending on the application. Both scenarios are discussed next.

6.1 Issuing high-level user demand during scene creation

In an offline environment during scene creation within Unity, a user can specify the trajectory a character should follow and the velocity it should travel at. This is done within a Unity engine editor session, using in-engine tooling to plot a Bezier curve during scene development. Once the scene is played back, given a suitably trained network, the developed system will produce character locomotive animation allowing the character to traverse the trajectory at the required speed.

The image below shows an example setup of the system in action. The purple trajectory, which the character will traverse, is plotted by the user during the editor session. As the user increases the desired speed of the character, the locomotive state will adapt from a walking state to a jogging state and finally to a running state. Since the training data used in this test included a wide range of locomotive behaviors, the system is able to output suitable and nuanced animation including upper body arm swinging and leaning naturally depending on the trajectory.

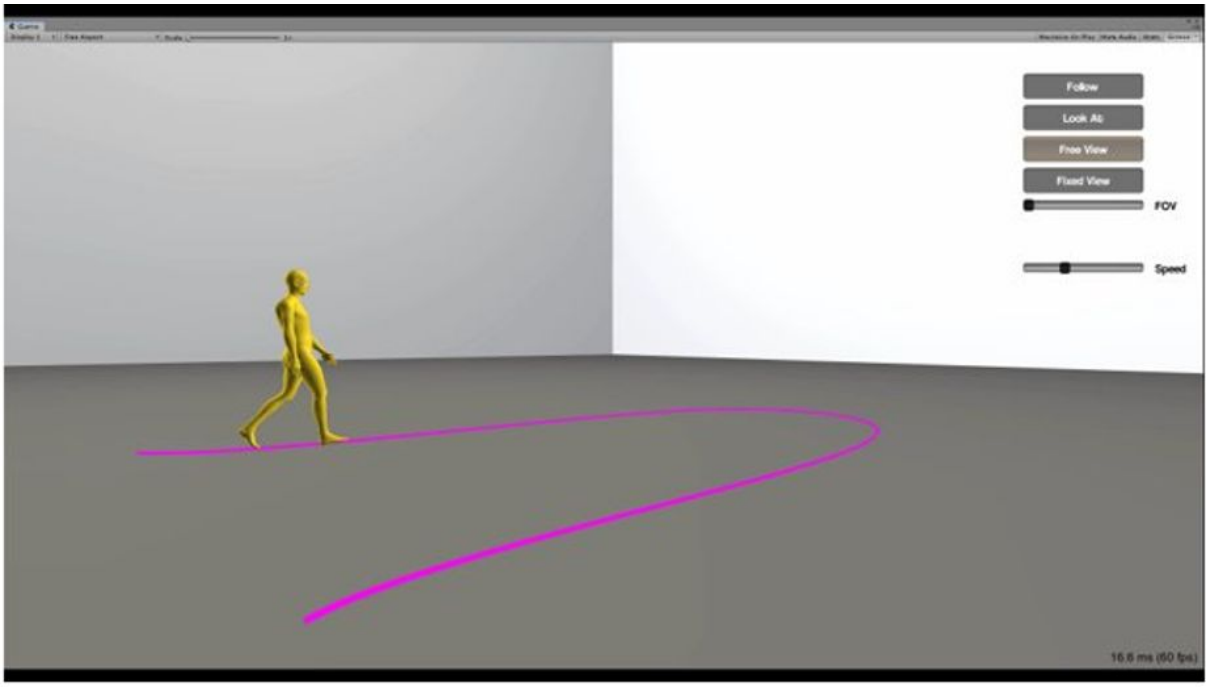

Figure 4: Example of setup during scene creation

6.2 Issuing high-level user demand at runtime

In this scenario, the trajectory and desired velocity can be driven dynamically based on world state by some implementation of an AI system. Traditional in-engine navigation meshes could be used to procedurally generate the desired trajectory for the character. This approach would impart a world and time awareness in our character, as animation is generated depending on the evolution of the virtual world state.

The second method is intended for use in an online environment such as a typical VR application. In this scenario the position and rotation of the HMD are used to drive the neural network which will

procedurally generate animation for the avatar in the virtual world as the human wearing the HMD moves through physical space. This offers a potential solution to a hard problem to solve traditionally: generating lower body animation in VR applications when you only have the head and hands as tracking points. Modern approaches to this solution typically implement a state machine to control the lower body animation that is driven based on the available data points which can be unreliable. Preliminary testing has shown that the machine learning system can generate reasonable lower body animation given a demand on the head.

7 DISCUSSION, CONCLUSION AND PERSPECTIVES

IKinema passed documentation and code to UPF, that has been working to adapt it for further work, with some initial support from the code creator (until November). We discuss some aspects of this ongoing work next.

1) The current implementation demo is based on Unity. It is an interactive scene where the user can define a trajectory and some speed parameters for a character to move along it while satisfying the user requirements. This demo is working well. The demo should allow interactive control by the user, for instance using a gamepad or unity navmeshes. This aspect of the demo in the context of a video game does not seem to be working correctly, and still refinement seems to be needed to demo this reliably.

2) The target for the adaptation of this IK work for further work was to achieve full integration within *WebGLStudio*. While progress in understanding and owning the work has been made, this goal has not been achieved, and being able to retrain the network seems a significant challenge, where further efforts are needed. UPF-GTI expects to understand better in the coming month (March 2020) the extent to which this can be achieved, and further efforts have been added.

The results of this deliverable, and indeed WP6T2 and WP6T3 should lead to further work on WP6T4, *Automatic synthesis of in-between animation and procedural transition*. The results of T3 show that directly applying ML techniques, drawing on historic animation data banks, the generation of smooth animations, including smooth transitions and in-betweening, satisfying high level semantic demands, should be achievable. Provided advances in 2) are made, this is the goal that UPF-GTI intends to push. As indicated earlier, a better assessment of the achievability will be made during March.

8 References

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Trans. Graph.* **37**, 4, Article 145 (July 2018), 11 pages. DOI:https://doi.org/10.1145/3197517.3201366

9 Acronyms and abbreviations

AI: Artificial Intelligence
HMD: Head-mounted display
MANN: Mode-adaptive neural network
PI controller: Proportional + Integral controller
VR: Virtual Reality