Modeling Physically Simulated Characters with Motion Networks

Robert Backman and Marcelo Kallmann

University of California Merced

Abstract. We present a system that allows non-programmers to create generic controllers for physically simulated characters. The core of our system is based on a directed acyclic graph of trajectory transformations, which can be modified by feedback terms and serve as reference motions tracked by the physically simulated character. We then introduce tools to enable the automatic creation of robust and parameterized controllers suitable for running in real-time applications, such as in computer games. The entire process is accomplished by means of a graphical user interface and we demonstrate how our system can be intuitively used to design a simbicon-like walking controller and a parameterized jump controller to be used in real-time simulations.

Keywords: physics-based character animation, humanoid control.

1 Introduction

While kinematic motion generation easily offers real-time performance and predictability of the resulting motions, they lack realistic interactions with the environment and reactions to disturbances. Physics-based simulated characters have great potential to improve these aspects, however useful controllers remain difficult to design and to integrate in real-time applications.

Although research on physically simulated characters have tremendously improved the creation of new and more robust motion controllers for a variety of tasks [1–3], controller design remains complex and difficult to extend to generic situations and tasks.

This paper proposes a system to enable the easy exploration, creation, and customization of real-time controllers without requiring low-level programming. The system enables the user to graphically create a directed acyclic graph of trajectory transformations, which can then be modified by feedback terms in order to generate reference motions that can be tracked by the physically simulated character.

Tools for expanding the controllers in respect to desired parameterizations are also introduced in order to enable the creation of robust controllers suitable for running at real time frame rates. We demonstrate how our system can be used to design a simbicon-like walking controller, to create a parameterized jump controller, and to adapt motion capture data to new constraints.

2 Related Work

The main current approach for real-time motion generation remains based on kinematic animations, typically using a finite set of motion capture or keyframed clips augmented with some blending and warping. For example, Levine et al. [4] show how a small number of motions can well control a kinematic character in real time. But as with any kinematic approach, the motions generated are limited to the examples given.

Physics can also be used to transform kinematic animations. For instance, de Lasa et al. [5] show a feature-based method for creating controllers for dynamic characters; and Popovic and Witkin [6] use a space-time optimization approach to transform motions in physically accurate ways.

The role of physically simulated characters in video games and movies has been anticipated for quite some time but to this date is mostly present as passive ragdoll interactions. Promising research has however been proposed to improve this situation. Methods for automatically switching from kinematics to physics only when needed have been developed [7], and methods for the extraction of the style in a motion capture clip have been transferred to physically simulated characters [8]. Complex motion capture sequences have also been achieved with physically simulated characters [3], however requiring computationally heavy sampling strategies.

Since the 90s researchers have been implementing increasingly robust control strategies for achieving a diverse set of skills. Early work by Hodgins et al. [2] showed how controllers could be adapted to new characters, Maccietto et al. [9] used momentum control to allow a character to follow motion capture data while responding to large disturbances, Yin et al. [10] introduced simple feedback rules for the swing leg to produce robust walking behavior, and Coros et al. [1] demonstrate robust bipeds performing various skills such as pushing and carrying objects, extending early work on the use of virtual forces with a Jacobian transpose method [11].

In summary, previous work in physics-based character animation has focused on the development of successful controllers for a number of tasks and situations. In this paper we introduce a system that is able to expose the creation and exploration of such controllers to designers in an intuitive way. In doing so our system proposes two main contributions: 1) a set of data processing nodes to model controllers with graph-like connections able to form complete control feedback loops, and 2) a simple and effective sampling-based algorithmic approach to automatically achieve robustness and parameterization of designed controllers.

3 System Overview

The core of our system has two main modules: an *animation module* and a *physics module*. The animation module generates target angles based on input trajectories. The physics module contains a tracking controller that produces the torques necessary to drive each joint of the physically simulated character towards the target angles specified by the animation module. Additionally, the physics module has a virtual force controller that adds additional torques to achieve higher level requirements such as staying balanced or achieving a global position of a body part in the joint hierarchy. See Figure 1 for an overview of the main parts of the system.

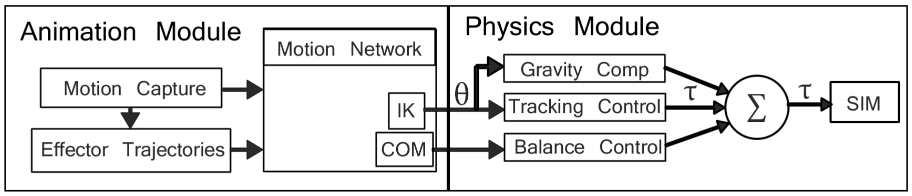

Fig. 1. Data flow of the system.

The animation module operates at 60fps feeding a stream of postures to the physically simulated character. The output trajectories are specified in joint angles, or also as end-effector positions, which are then converted to joint angles using Inverse Kinematics (IK).

The physically simulated character is composed of a set of rigid bodies connected by hinge, universal and ball joints as shown in Figure 2. Each rigid body in the character is approximated by a bounding box for fast collision tests. The character is simulated using the Open Dynamics engine (ODE) and is running at 1200 FPS. The reason for the high simulation frame rate is to well handle high speed contacts, which is a potential problem with typical forward dynamic simulations.

The following Section describes the physics module. The animation module is then presented in Section 5.

4 Physics Module

The Physics Module has two main components: tracking control and virtual force control. The tracking control makes the character follow joint angles specified by the animation module. The virtual force control allows higher level goals such as maintaining balance, global effector goals and gravity compensation.

4.1 Tracking control

A PD servo at each joint is responsible for tracking the target angular value θ_{goal} and rotational velocity v_{goal} specified from the animation module for each rigid body. We use PD gain coefficients of $k_p = 4kn/rad$ and $k_d = 2\sqrt{k_p}$.

The goal angle θ_{goal} for each joint can be either specified in local coordinates or in a *heading-based frame*. The heading-based frame is calculated by starting

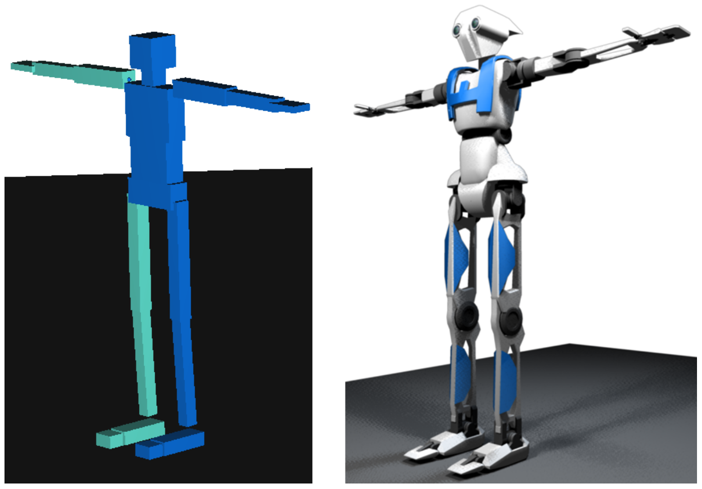

Fig. 2. Simplified model for collision detection (left) and full character model (right).

with the root joint global orientation then aligning it with the world up vector. Knowing the heading of the character is useful for encoding the notion of front, back, left and right of the character independent of its orientation and is critical for balance feedback. However, this approximation can be a problem if the character flips upside down since the heading will suddenly change.

4.2 Virtual force control

Virtual model control was introduced by Pratt et al. [11] for bipedal robots and has then been used in many systems [1]. Considering the character is under actuated, it is desirable to control certain components with external forces. It would be a trivial matter to have a character stay balanced and track arbitrary motions by applying external forces to each rigid body that composes the character, however this would effectively defeat the purpose of physics simulation since the result would typically be unrealistic. To approximate this level of global control we can imagine a virtual external force acting on a rigid body to achieve some goal, then convert this virtual force to internally realizable torques that span from the affected body up the joint hierarchy to a more stationary body.

For example, to control the position of a character's hand in global coordinates we calculate a virtual force that will move the hand towards a goal configuration (see Figure 3), and then convert this virtual force into a set of torques that can be applied to the arm joints to achieve a similar motion. Ideally this chain of rigid bodies would span all the way to the foot in contact with the ground but in practice it only needs to go to the torso of the character.

Another use of virtual control is to control the swing foot during a walk cycle since it will rarely be at the same state as the input motion, this can be critical for preventing premature foot contact with the ground. To maintain static balance of the character we employ a similar virtual force on the COM to bring its floor projection to the center of the support polygon and then convert the virtual force to joint torques for the stance leg. It is also a simple way to control the velocity of the COM while walking.

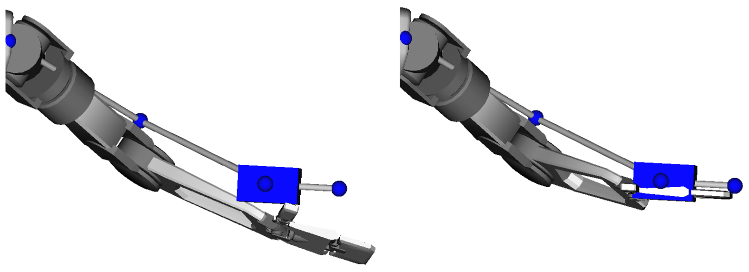

Fig. 3. Global targets may be difficult to reach with joint control (left). Virtual forces can be effective (right).

Gravity compensation torques are also computed to allow lower gain tracking by proactively countering the effects of gravity. For each joint that is considered for gravity compensation the COM and mass of the descendant joints is calculated, then a torque is applied that would counter the moment this mass would create. Gravity compensation is typically only applied to the upper body but can also be used for the swing leg for walking.

An important component of our balance strategy is controlling the orientation of the torso independent of the contact configuration. Without considering the root orientation the character typically leans over and falls as soon as it lifts the swing leg due to the sudden change in torque requirements for the stance leg. However, since the torso has no parent joints directly in contact with the ground it cannot directly be actuated, so instead a virtual torque is calculated for the root that must be distributed to the stance leg, for double support this torque is distributed to both legs.

With these components we are able to achieve full body posture control to maintain balance while completing complex global objectives, such as touching a characters toe while standing on one foot, as showin in Figure 4.

5 Animation Module

The animation module creates kinematic motions that are used as input to the physics module. The motion can be described as a sequence of full body postures determined at each frame by spline trajectories controlling joint angles or effector trajectories (that are later converted to angles by IK). Trajectories can be encoded by a sparse list of control points and tangents or by a dense list of points that are linearly interpolated (typically the case of mocap data).

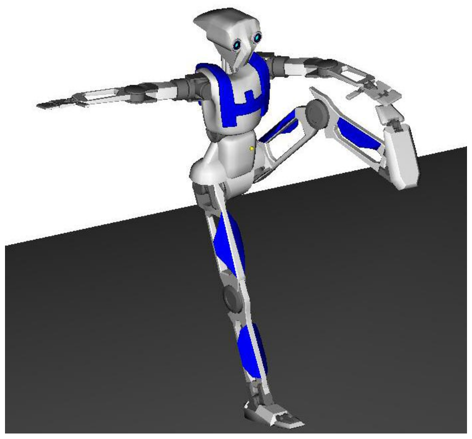

Fig. 4. Character performing balance and reaching

Another common type of trajectory in our system is a step function, where each control point is constrained to be either zero or one, representing true or false. The output may be a combination of the above methods, for example the upper body may be driven by Euler angles derived from motion capture data while the lower body may be driven by feet IK trajectories.

The methods described above can create simple motions that maintain balance such as jumping to a known goal, but they fail when there are external disturbances or changes in the environment, or if the motion gets complicated with large sudden changes in contact. Typically, to control balance, a separate system of feedback controllers is layered over the animation module to override or modulate the reference motions that are produced. But we are interested in a composite/unified approach that brings feedback terms directly into the animation module.

In addition to having control points (or frames) serving as input for the animation module, feedback variables become additional inputs to the animation system and gain parameters become additional outputs. Trajectories can thus control any parameter in the system. For example for jumping, a trajectory is needed for representing a global gain multiplier for the tracking control so the character can become stiff before jumping, then less stiff during flight, then gradually stiffer to the default value for landing. Another trajectory can be used to change the gains on a virtual force that is computed for the swing foot that gradually transitions towards the end of a step.

By exposing the lowest level control parameters to the designer we raise the risk of them creating non-functioning controllers, but we also provide the potential of exploration and creation of endless possible controllers.

6 Time-Dependent Directed Acyclic Graph

The physics module is able to maintain balance and achieve high level goals such as foot and hand global positions but the parameters are static. The animation engine generates motion but it has no notion of the physics. To interconnect these two modules we present a system called a Time-dependent Directed Acyclic Graph (T-DAG), inspired by the Directed Acyclic Graph in Autodesk Maya.

The T-DAG interconnects the animation module with the physics module. To foster the intuitive development of controllers we then propose a graphics user interface to expose the parameters of the physics module and to connect them with appropriate channels from the animation module.

Any relevant parameter can be exposed as a *channel*. Channels can represent the orientation or position of an IK end-effector, an individual joint angle, boolean values, feedback parameters or gain parameters. Some examples of feedback parameters are: the pressure of the stance foot, the velocity or relative position of the COM, and many others.

The user interface allows the individual or group assignment of channels to any type of trajectory. For example with a forward jump, since the motion is typically left-right symmetric, we have one trajectory that specifies the rotation of the foot but we connect it to both feet.

6.1 Operation Nodes

To transform the motions several *operation nodes* are introduced. Each node in the control graph takes as input a set of channels (trajectories or feedback parameters), performs an operation, and then outputs the transformed value.

The animation module outputs trajectories based on editable Splines or on feedback parameter that are used as input (see Figure 5). After a group of nodes is connected in a desired way, the T-DAG network can be saved as a template to be used for other controllers or duplicated for other channels. Then, by varying the input trajectories several goals can be achieved without changing the network connections.

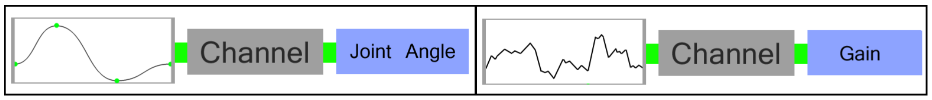

Fig. 5. Animation nodes can be based on spline input (left) or feedback input (right).

Several key operations needed to model controllers are available as nodes to be interconnected and added to the T-DAG. These operations are illustrated in Figure 6, and are described below:

– Addition node: allows to add trajectories. It can take any number of inputs.
– Multiplication node: allows to multiply the values from two trajectories.

– Modulation node: this node requires one step function and at least one trajectory input. For each high step in the control input (step function) the first input trajectory is scaled in time to fit within the step, and for each low step the second input trajectory, if there is one, is scaled in time and fit within the low step.
– Switch node: it also requires one step function and one or two input channels. If the step function is high it outputs the first input, and if it is low it outputs the second input (if there is one).

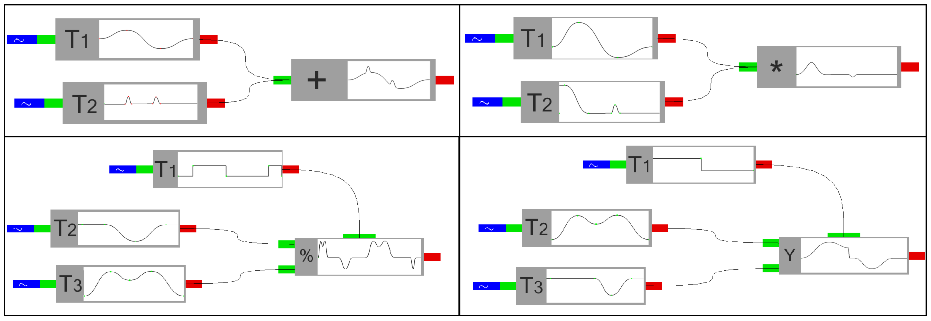

Fig. 6. Operation nodes, in top-down left-right order: addition, multiplication, modulation and switch.

An intuitive graphical user interface was developed to allow designers to edit and explore T-DAGs. Figures 8, 9 and 10 are direct snapshots from the graphical input panel of our motion network editor. The accompanying video to this paper illustrates interactive sessions with our system[1].

7 Trajectory Randomization to discover controllers

Once a T-DAG is built trajectories can be connected in different ways. They can be designed by hand (using editable Splines) or they can come from motion capture. Editable offset trajectories can also be easily associated to motion capture trajectories by using an addition node, allowing different possibilities for customization. Any set of control points can then be exposed to an automatic sampling procedure that will explore successful variations of the controller in order to achieve parameterizations for real-time execution.

A *cost function* is selected as a network of feedback channels, for example jumping requires the character to be statically balanced at the end of the motion and to minimize the total torque of the tracking control. Walking requires the character to be upright and have the COM moved by some minimum distance. Additionally, there is a boolean parameter provided by the physics module which

[1] video available at http://graphics.ucmerced.edu/projects/12-mig-mnet/

looks for non-foot contacts with the environment which multiplies with the cost to ignore any motions that have *bad contacts*. There is also an *objective function* that is the goal of the simulation, the goal can be a certain target distance, or for example, a certain distance and height for a jump controller to achieve.

Once the objective and cost networks are constructed, a sampling process can be initiated. The trajectory control points that are exposed are randomly sampled within initial *sampling bounds* and the simulation is run along with the controller. If after n tries the controller does not achieve the objective, the sampling bounds are enlarged and the process re-starts. If the objective is satisfied then the control points are saved to a file along with *motion descriptors* of the outcome (the achieved jump distance, walk speed, etc).

After several motions are found that successfully complete several objectives, the successful motions are then used as starting points for a new round of iterations. We randomly choose new objectives and then use radial basis interpolation of the k-closest previous successful examples to find a set of trajectories which would ideally meet the objective. This typically does not work at first since there is no guarantee that interpolating successful controllers will give a new functional controller, but it works well as a starting point for the next round of sampling. The longer the sampling process runs, the better the interpolations become. When enough coverage of the desired variations is achieved, an effective parameterization of the objective space is achieved.

8 Results and Discussion

We show three T-DAG examples for achieving walking, jumping and an example of motion capture editing.

For achieving walking (see Figure 7), we start with a step in place motion that was user created and the objective was to move at least 5 meters while staying balanced for 10 seconds. The sampled control points were the root joint and stance foot rotation in the sagittal plane, and the desired velocity of the COM. Then the motion that traveled the furthest was used as the starting point for the next iteration of trials with an increased objective distance of 10 meters.

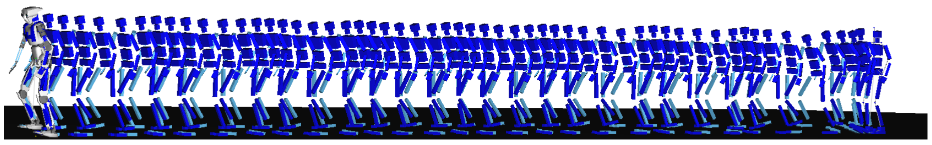

Fig. 7. Example walking using the Simbicon controller.

Figure 8 shows the used feedback nodes, which implements the Simbicon feedback rules $\theta_d = \theta_{d_0} + CdD + CvV$ [10]. It offsets the swing leg and root joint target angle according to the distance from the COM to the stance foot.

The inputs to the network are the offset between the COM and the stance foot trajectory in the previous frame as well as the COM velocity. This is multiplied by a gain parameter and passed into the offset parameter of the swing leg. The swing leg is determined by a switch node that has as input the stance state step function.

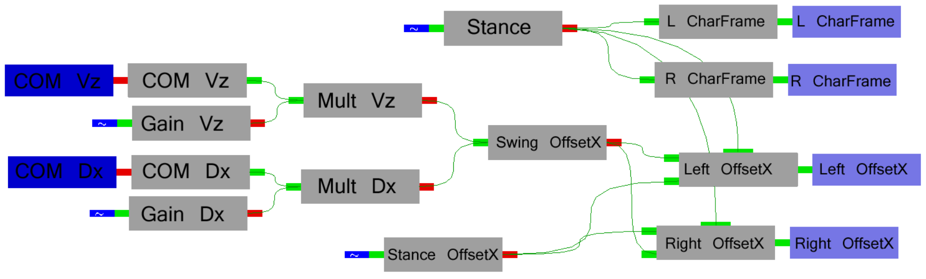

Fig. 8. The network to implement the Simbicon feedback rules.

To achieve jumping (see Figure 9) the objective was to achieve static balance within 5 seconds and to travel at least 0.5 meters. The root joint position and orientation in the sagittal plane was sampled while the feet were stationary and solved for with IK. Both feet were connected to the same trajectory for the rotation in the sagittal plane to achieve desired toe off in the jump. Also, the upper arms were connected to a single trajectory. Figure 12 shows an example of two separate jumps that were created from the same network.

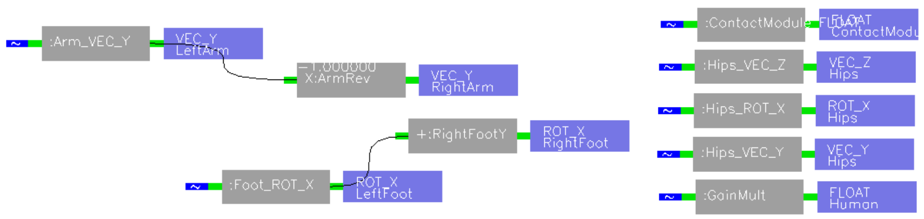

Fig. 9. The network to generate jumps.

Figure 11 shows an example where we use the network in Figure 10 to modify the step height of the motion with a single trajectory. We hope to use this in future work to achieve physically simulated characters following motion capture. The goal is to produce swing foot offsets that will raise the foot independent of the step phase of the motion capture data. The nodes in the center of Figure 10 have as inputs trajectories extracted from the motion capture data (in this case the y position of each foot). These are added to a swing trajectory (far left) that is modulated based on the stance state. Several of these results are presented in the accompanying video.

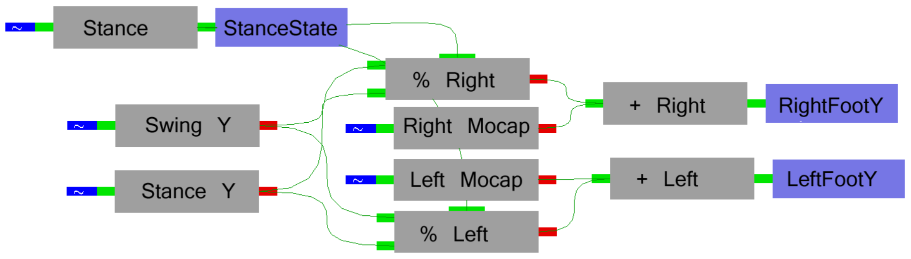

Fig. 10. The network to transform the motion.

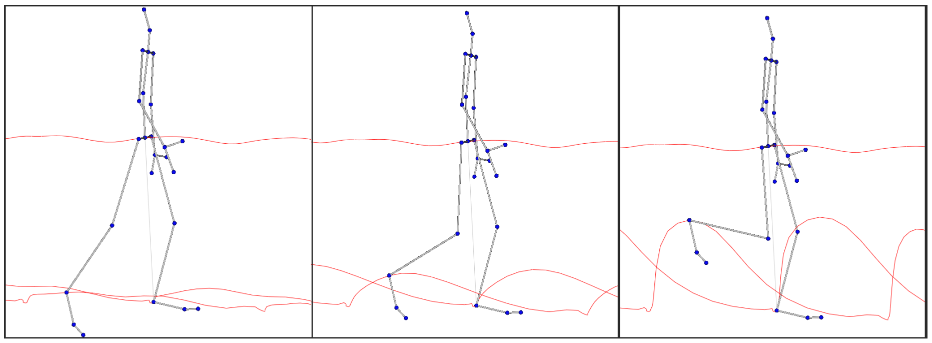

Fig. 11. Three values for the swing foot offset curve.

9 Conclusions and Final Remarks

We presented a system that allows users to create controllers for physically simulated characters without low-level programming. Our system introduces a new methodology to explore, edit and create parameterized physics controllers that can be later used in real-time applications. We are starting to use our system for the rapid prototyping and customization of physics behaviors designed to improve game-based therapeutic applications and the results obtained so far are promising.

Acknowledgments This work was partially supported by CITRIS grant 128.

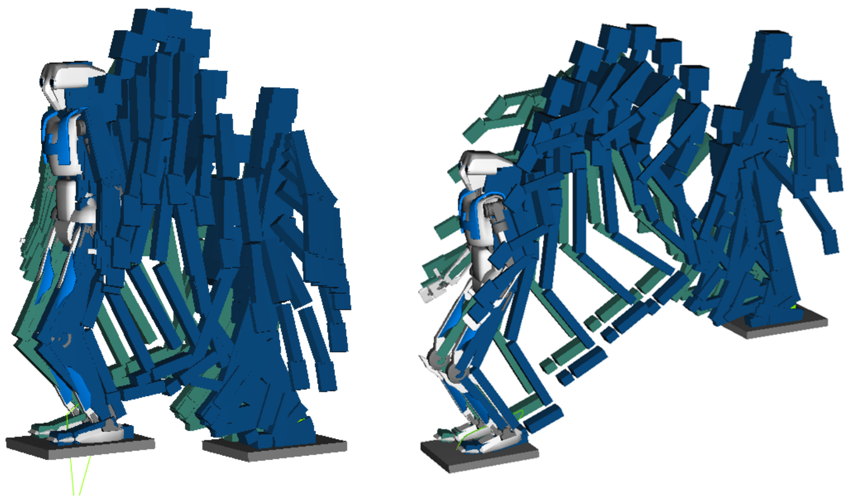

Fig. 12. Two jumps created from the same motion network.

References

1. Coros, S., Beaudoin, P., van de Panne, M.: Generalized biped walking control. ACM Transactions on Graphics (Proceedings of SIGGRAPH) **29**(4) (2010)
2. Hodgins, J.K., Pollard, N.S.: Adapting simulated behaviors for new characters. In: ACM Transactions on Graphics (Proceedings of SIGGRAPH). (1997) 153–162
3. Liu, L., Yin, K., van de Panne, M., Shao, T., Xu, W.: Sampling-based contact-rich motion control. ACM Transactions on Graphics (Proceedings of SIGGRAPH) **29**(4) (2010)
4. Levine, S., Wang, J.M., Haraux, A., Popović, Z., Koltun, V.: Continuous character control with low-dimensional embeddings. ACM Transactions on Graphics (Proceedings of SIGGRAPH) **31**(4) (2012)
5. de Lasa, M., Mordatch, I., Hertzmann, A.: Feature-based locomotion controllers. ACM Transactions on Graphics (Proceedings of SIGGRAPH) **29**(3) (2010)
6. Popovic, Z., Witkin, A.P.: Physically based motion transformation. ACM Transactions on Graphics (Proceedings of SIGGRAPH) (1999)
7. Zordan, V., Macchietto, A., Medina, J., Soriano, M., chih Wu, C.: Interactive dynamic response for games. In: Proceedings of the ACM SIGGRAPH symposium on video games. (2007)
8. Yoonsang Lee, S.K., Lee, J.: Data-driven biped control. ACM Transactions on Graphics (Proceedings of SIGGRAPH) **29**(4) (2010)
9. Macchietto, A., Zordan, V., Shelton, C.R.: Momentum control for balance. ACM Transactions on Graphics (Proceedings of SIGGRAPH) **28**(3) (2009)
10. Yin, K., Loken, K., van de Panne, M.: Simbicon: Simple biped locomotion control. ACM Transactions on Graphics (Proceedings of SIGGRAPH) **26**(3) (2007)
11. Pratt, J., Chew, C.M., Torres, A., Dilworth, P., Pratt, G.: Virtual model control: An intuitive approach for bipedal locomotion. The International Journal of Robotics Research **20**(2) (2001) 129–143