Neuroevolution of Actively Controlled Virtual Characters - An Experiment for an Eight-legged Character

Svein Inge Albrigtsen, Alexander Imenes, Morten Goodwin, and Lei Jiao

Centre for Artificial Intelligence Research, University of Agder,
4879, Grimstad, Norway
{thhethssmuz, alexander.imenes}@gmail.com
{morten.goodwin, lei.jiao}@uia.no

Abstract. Physics-based character animation offers an attractive alternative for traditional animations. However, it is often strenuous for a physics-based approach to incorporate active user control of different characters. In this paper, a neuroevolutionary approach is proposed using HyperNEAT to combine individually trained neural controllers to form a control strategy for a simulated eight-legged character, which is a previously untested character morphology for this algorithm. It is aimed to evaluate the robustness and responsiveness of the control strategy that changes the controllers based on simulated user inputs. The experiment result shows that HyperNEAT is able to evolve long walking controllers for this character. In addition, it also suggests a requirement for further refinement when operated in tandem.

Keywords: Artificial intelligence · Neuroevolution· Actively Controlled Virtual Characters· Eight-legged Character.

1 Introduction

Character animation has become an important part of modern game development. The responsiveness and perceived realism of these animations play a key role for providing an immersive experience to the players. While an experienced animator is often able to create lifelike and realistic looking animations, the limitation of these animations is also obvious as they are only applicable for the purpose that the animation portrays. When the animations are utilized outside of their intended domain, however slightly, they start to fall short.

Physics-based simulation offers an attractive alternative to traditional animation techniques, wherein each motion is the direct result of a physics simulation and is therefore physically realistic by definition [5]. Physics-based animations are commonly used to simulate passive phenomena like objects, cloths, fluids and ragdolls. However, for more active animations most games still resort to kinematics-based approaches [5, 11, 7]. One of the commonly cited reasons is that physics-based simulated characters are notoriously difficult to control, as

all movement has to be controlled by the application of torques and/or other forces [5]. One way to tackle this issue is to train controllers using machine learning, a technique that has shown promising results [13, 6, 12, 14, 1, 2, 10, 15, 8, 9]. However, such controllers are usually trained for one singular purpose, or an action, and has little application outside of that purpose. While one approach could be to train multiple controllers and then switch between them, the actual implementations of this logic is sparse. It is therefore not known whether these controllers could handle the incessant switching that would be required inside a highly interactive environment like a game. In this paper, we investigate the feasibility of combining individually trained neural controllers to form a control strategy for actively controlled virtual characters. The proposed approach taken to this end utilizes neuroevolution to train a small set of neural controllers. These controllers will be utilized to generate joint torques for a physics-based simulated multi-legged character to produce motion for a corresponding set of targeted behaviors. In addition, the efficiency of these controllers can be evaluated based on how robust they perform when switching among them. The main goal of this study is twofold. Firstly, we evaluate whether HyperNEAT is able to evolve gaits for an eight-legged character, which, to the best of our knowledge, has not been studied before. Secondly, we study whether neural controllers trained for different behaviors can operate in tandem to produce robust interactive controllers.

The contributions of this study mainly reside in the approach taken to train these networks, by using HyperNEAT to generate gaits for a previously untested, highly complex, eight-legged character. We also provide insight into the efficiency of constructing high-level controllers using such evolved networks. The unabridged version[1] of this work, the source code[2] of this study and the videos[3] are all available online.

The remainder of this paper is organized as follows. In Sec. 2, the approach of the experiment is detailed. The numerical results are given in Sec. 3 before we conclude the work in the last section.

2 The HyperNEAT based Approach

This section details the approach and methods applied in this study. More specifically, the design of the character model and substrate, and most notably the setup of the physics simulation are presented.

2.1 Multi-legged Character

Before we explain the multi-legged character, the environment for our study is introduced presently. In this study, we use a custom OpenGL-based game engine that has been adapted to incorporate a physics engine and neuroevolution library in order to simulate and train the controllers. The engine itself is mainly used

[1] http://hdl.handle.net/11250/2454827.
[2] https://github.com/reewr/master.
[3] https://reewr.github.io/master.

for visualization and to control the flow of the physics-based simulation. The Bullet3 physics engine[4] is adopted to perform all rigid body simulation and collision detection required to simulate the character. To evolve the controllers, the neuroevolution library MultiNEAT[5] is utilized. The choice of MultiNEAT in particular is made due to language compatibility with the rest of the code.

Fig. 1. The arachnoid used in this study.

In the aforementioned game engine, an arachnoid with a robotic theme, featuring eight legs each of which is equipped with five different joints, is utilized in our study, as shown in Fig. 1. This model is chosen as it has a sufficient complexity to provide a realistic example of a fully featured game character. All the joints of the character are defined as hinge joints, meaning that they can only rotate in one specific axis. Furthermore, all these joints have limitations in how much they can move in their respective axises. This is to enforce realistic motions as the joints should not have a full 360-degree freedom of movement. Fig. 2 shows each joint and their limits. As seen in Fig. 2, the Trochanter is the only joint that is able to rotate forwards or backwards in relation to the sternum, whereas every other joint can only rotate up and down. The actual values of these limits can be seen in Tab. 1. The character has four additional parts that

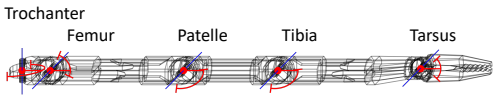

Fig. 2. Leg with joint limits.

are not mentioned in Table 1, the head, neck, hip and abdomen, which may also be rotated. However, these parts are not considered to contribute much in terms

[4] www.bulletphysics.org.
[5] www.multineat.com.

of the movement of the character and therefore have been disabled and set to a constant angle that should not interfere with the movement of the character.

The dimensions of the character are roughly 4 units from its head to the back of the abdomen and has approximately 9 units in total leg span when fully stretched out. Each part of the character has a weight of 1 mass unit, except for the sternum, abdomen and head which weighs 10 mass units, 8 mass units and 5 mass units respectively. These units are configured to make sure that the center of mass is not exactly at the center of the character, but rather slightly tilted towards the back, in order to increase the perceived realism of the character.

Table 1. Limits of the joints of the character's legs in degrees.

Joint	Upper limit	Lower limit
Trochanter	-60	60
Femur	-20	60
Patella	-100	5
Tibia	-100	5
Tarsus	-35	35

2.2 Substrate Applied in HyperNEAT

As mentioned above, HyperNEAT uses a substrate to define the coordinates for input, hidden and output nodes. Designing the substrate is an important aspect of using HyperNEAT as it contributes to how it will determine symmetries and patterns [4].

The substrate is designed to match the geometry of the character as closely as possible to make it easier for HyperNEAT to detect the symmetries of the legs. The substrate takes inspiration from previous research that has evolved gaits for Quadruped [2, 15, 8, 3], but is extended to support the additional two legs on each side and the increased number of joints. The substrate with all its layers can be seen in Fig. 3, which is defined with three two-dimensional 7×8 Cartesian grids. All unmarked inputs/outputs are un-utilized.

Each column in the substrate represents a leg, starting from the front left and going clockwise around the character. Each row, except for the two top-most rows, represents the current angle of a joint normalized from a value within [-π, π] to a value within [-1, 1]. The second row represents whether the tip of the leg is touching the floor whereas the first row includes the pitch, roll and yaw of the sternum. The first row also includes a sine and cosine wave to encourage periodic behavior. All the inputs, hidden and output coordinates of the substrate are spread uniformly in the range of [-1, 1], while trying to keep symmetry between opposite legs. To differentiate the layers, the inputs, hidden and output layers

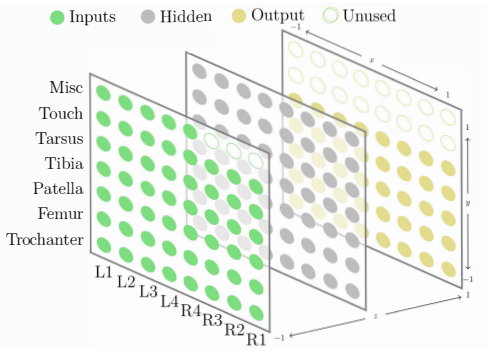

Fig. 3. The three-dimensional substrate of the character.

are placed on different z-coordinates, where the input z-coordinate is 1, hidden z-coordinate is 0 and output z-coordinate is 1.

The outputs are expected to be within the range [1, 1]. The current angle of the joint is subtracted from the output and the result of this is set as the new velocity of the joint. This simulates setting a target angle of the joint, allowing the networks to choose their desired angles of each joint.

2.3 Controllers

In order to create the control strategy, the neural controllers for each action has to be trained first.

The first controller is trained to make the character stand completely still in a standing position. While training, it will be awarded for not moving away from the initial starting position and for remaining still in a balanced pose. If the character falls at any point during the simulation, i.e. touches the ground with any vital parts, the simulation will be terminated, and the character will be rewarded for the length of time it stayed alive. This is mainly to discourage such behavior by limiting the fitness of individuals that are unable to remain upright throughout the entirety of the simulation, promoting more stable postures.

The second controller is trained to make the character walk with a stable gait. Working under the assumption that the further the character walks the more stable the gait must be. Therefore, the fitness of the character will be the furthest distance traveled in one specific axis. In this case, it is the positive z-axis, which is the axis that the character is facing at the start of the simulation. As with the first controller, if the character falls, the simulation will stop and the final fitness value will be the furthest distance traveled before it falls.

The reason for choosing these two actions is that together they form a minimum of actions required to test the control strategy. Walking gaits have been evolved for various legged creatures before and thus evolving this behavior for the character used in this project should hopefully not be much of a stretch, con-

sidering the characters heightened complexity. While it would easily be possible not to have a separate standing controller, e.g., by just locking all the joints of the character instead, this could possibly cause the character to fall over if it happens to stop in some unbalanced positions. Thus, it is required for a separate standing controller that could account for such imbalance and other residual forces from the walking controller after a transition.

Both controllers are trained via HyperNEAT using the substrate described in the previous section, and use the same input and output scheme as described above.

The simulation process will be the same for both controllers and they will only differ in the fitness functions that they use. All simulations will be run with 150 individuals with randomized weights based on a random seed for 300 generations. Each character will be allowed to run for up to 10 seconds in real time, in addition to an un-simulated process of one second where the character is positioned into a standing pose that is equal across all simulations.

Due to the complexity of HyperNEAT and its underlying NEAT algorithm, it has a variety of different parameters to set, most of which have been heavily based on previous research for similar multi-legged characters [4, 15, 3, 8]. However, these may be subject to a certain degree of trial-and-error depending on the results. A full list of the NEAT/HyperNEAT parameters may be found in the unabridged version of this work.

2.4 Control Strategy

Once these two controllers are in place, a control strategy can be evaluated by using the above controllers together. The control strategy is designed to measure the responsiveness of the controllers, which is done by using the controllers in sequence and measuring the time that it takes to transition from standing to walking and vice versa.

3 Performance Evaluations

In this section, we will evaluate our proposed approach by two examples, i.e., standing controller and walking controller independently, and then will test the controlling strategy, i.e., by testing them alternating in a tandem manner.

3.1 Standing Controller

The standing controller is evaluated based on its ability to stand still and not falling to the ground.

In early generations of the simulation the dominating factor by far is the killing of individuals who fell. In the first generations rarely does even a single individual live long enough as to complete the full simulation duration. This leads to a dramatic increase in simulation speed in the early stages as it allows the simulation to be cut short and the next generation could be started.

Seeing as the controller is directly rewarded for how long it remains alive, it is not surprising that most of the controllers managed to learn that staying upright is a good strategy. However, many controllers go beyond that and therefore often flail about frantically in order to remain upright. An example of this behavior can be seen in Fig. 4.

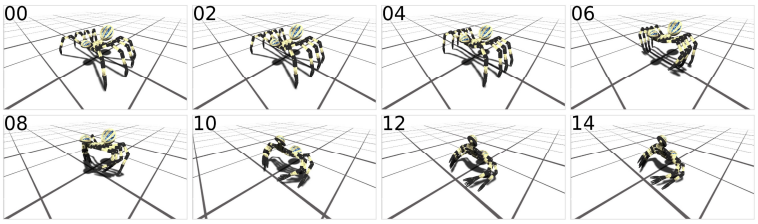

Fig. 4. Image series of the first standing controller.

This controller tries to remain upright by pulling all its legs towards its body, however, it appears to do so with an equal force across all legs. Seeing as the characters center of mass is slightly towards the back, this symmetrical application of force leads the character to tilt, eventually falling backwards.

Other controllers manage to learn that certain poses are easier to balance than others. As such, many controllers evolved various static poses that they are able to hold near indefinitely. However, some of these poses look more like a spider mannequin that has been randomly assembled by a tornado. One example of this can be seen in Fig. 5. This controller does take some time to gather itself, but eventually converges to a stable pose. While certainly amusing, it does not meet the requirements of the control strategy, as it is considered unlikely that a walking controller could naturally transition into this particular pose.

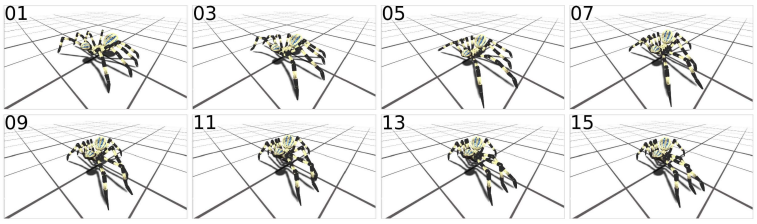

Fig. 5. Image series of another standing controller.

Among many controllers for standing, a controller is selected as the champion, and an image series of the champion controller is shown in Fig. 6. The final

controller does not achieve a perfectly balanced pose as it slowly descended towards the ground. However, it never does any drastic motions to position itself and are therefore judged to be the most compatible with other controllers in the control strategy.

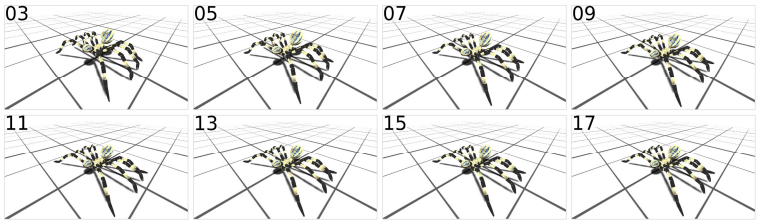

Fig. 6. Image series of the champion standing controller.

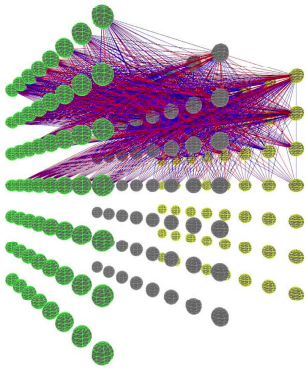

Fig. 7. The generated neural network for the champion standing controller.

Fig. 7 illustrates the generated neural network for the champion standing controller. As can be observed, the number of connections is low and they are mostly grouped around the top of the substrate. Since it was rewarded for not moving and only to keep itself from falling, it only had to use some of the joints to do exactly that. Seeing as the number of connections in a network will increase its sensitivity, this sensitivity would usually cause it to move around more. In this case, HyperNEAT discovers that reducing the number of connections in the network would make it better at standing still. An additional note is that most of the outputs are also not used. The downside of this type of network is that it is very hard for it to react to changes, unless those changes happen to hit the very few input nodes that its connected to.

3.2 Walking Controller

The walking controller was evaluated purely based on how far it traveled in one specific axis while remaining upright. While all the evolved controllers are able to move forward to some degree, classifying all of them as gaits would be a stretch.

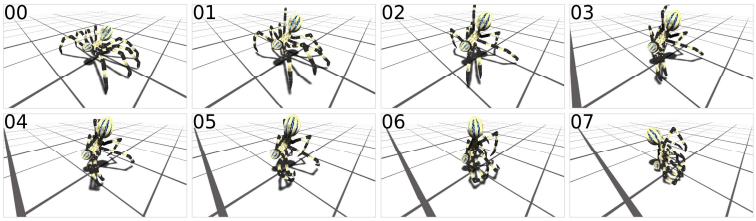

Fig. 8. Image series highlighting the gait of the first walker.

The controllers often quickly learned that standing as far up as it could and then falling forward would give it a fairly decent fitness score. In some training runs it learned that by throwing itself forward from the standing position where it started in, it could easily cover a distance of 4-5 units. If evolution discovered any of these behaviors early in the simulation, it would often outcompete other individuals and eventually dominate the population. Further evolution of these controllers would usually only get incrementally better at falling over or throwing themselves forward. An example from such a controller can be seen in Fig. 8.

After many trial of different controllers, the champion controller is presented below. This controller, which is not punished for touching the ground, displays the most purposeful and straight movement of all the controllers. The gait exhibits a strong asymmetric tendency in two distinct groups, consisting of the four front and four back legs, resembling a form of gallop. This controller is also capable of sustained walking far beyond the training period, in addition to being stable. The image series of the champion walker is shown in Fig. 9 and the final generated neural network for the champion is illustrated in Fig. 10.

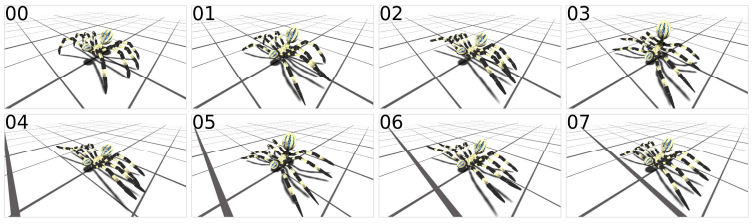

Fig. 9. Image series of the gait of the champion walker.

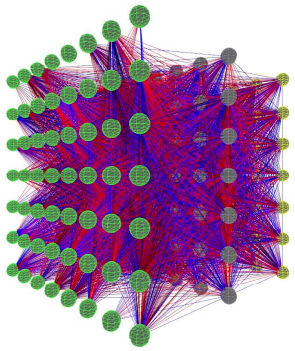

Fig. 10. The generated neural network for the champion walker.

3.3 Controller Strategy

The proposed control strategy is evaluated based on its responsiveness and robustness when switching between the two controllers. For this experiment, the walking and standing controllers are sequenced in a loop with 5 second time intervals, and the velocity of the character is recorded at each update. Fig. 11 shows the result of this experiment.

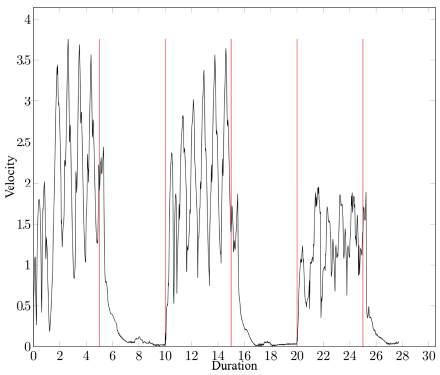

Fig. 11. Velocity of the character over time in seconds. The transitions between walking and standing are indicated by the red vertical lines.

Measuring the actual responsiveness of this experiment was more difficult than anticipated, as the standing controller never stopped completely. That is to say, its velocity never reached exactly 0 as can be seen in Fig. 11. Since the

controller never reaches the threshold value of 0, by using a larger threshold value, it becomes possible to measure the responsiveness when switching from walking to standing. Depending on how strict this threshold value is configured, the average response time is measured to 0.41s for a lenient threshold velocity of 1 and 1.54s when the threshold is set to a strict 0.1.

The same measurement is applied for the switching from the standing to the walking controller. The walking controller does not have a constant speed, but rather displays a periodically varying phase of movement and rest. However, measuring the responsiveness with a threshold value is similarly effective, even when the threshold is set higher than the low-points of the resting period of its gait. The average response time for this switch is measured at 0.09s for a lenient threshold velocity of 0.5 and 0.65s for a stricter threshold value of 1.5.

As can be seen in Fig. 11, the speed of the walking controller reduces noticeably after the second switch from the standing controller. The reason for this is that it locks one of its forelegs under two of its hind legs.

4 Conclusions

We investigate the feasibility of combining individually trained neural controllers to form a control strategy that could be used to actively control virtual physics-based characters. The HyperNEAT is adopted to evolve neural controllers for a previously untested domain of eight-legged locomotion. To create and evaluate the control strategy, two target neural controllers are trained for standing and walking. The newly trained neural controllers are combined to form the control strategy and is evaluated for its robustness and responsiveness when switching between them. The results show that HyperNEAT is able to evolve gaits for a highly complex eight-legged character. The resulting gaits showed that they are quite capable of walking long distances, even beyond the training period. However, when combined within the control strategy, the results suggest a need for further refinement as the controllers are still not robust enough to operate in tandem.

References

1. Allen, B.F., Faloutsos, P.: Evolved controllers for simulated locomotion. In: Proceedings of the 2nd International Workshop on Motion in Games. pp. 219–230. MIG '09, Springer-Verlag, Berlin, Heidelberg (2009)
2. Clune, J., Beckmann, B.E., Ofria, C., Pennock, R.T.: Evolving coordinated quadruped gaits with the hyperneat generative encoding. In: 2009 IEEE Congress on Evolutionary Computation. pp. 2764–2771 (May 2009). https://doi.org/10.1109/CEC.2009.4983289
3. Clune, J., Stanley, K.O., Pennock, R.T., Ofria, C.: On the performance of indirect encoding across the continuum of regularity. IEEE Transactions on Evolutionary Computation **15**(3), 346–367 (June 2011). https://doi.org/10.1109/TEVC.2010.2104157

4. Clune, J., Ofria, C., Pennock, R.T.: The sensitivity of HyperNEAT to different geometric representations of a problem. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation. pp. 675–682. GECCO '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1569901.1569995
5. Geijtenbeek, T., Pronost, N.: Interactive character animation using simulated physics: A state-of-the-art review. Comput. Graph. Forum **31**(8), 2492–2515 (Dec 2012), http://dx.doi.org/10.1111/j.1467-8659.2012.03189.x
6. Grzeszczuk, R., Terzopoulos, D.: Automated learning of muscle-actuated locomotion through control abstraction. In: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques. pp. 63–70. SIGGRAPH '95, ACM, New York, NY, USA (1995), http://doi.acm.org/10.1145/218380.218411
7. Hagenaars, M.: Hierarchical development of physics-based animation controllers. Master's thesis, Utrecht University (2014)
8. Lee, S., Yosinski, J., Glette, K., Lipson, H., Clune, J.: Evolving Gaits for Physical Robots with the HyperNEAT Generative Encoding: The Benefits of Simulation, pp. 540–549. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
9. Morse, G., Risi, S., Snyder, C.R., Stanley, K.O.: Single-unit pattern generators for quadruped locomotion. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation. pp. 719–726. GECCO '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2463372.2463461
10. Olson, R.S.: A step toward evolving biped walking behavior through indirect encoding. Honors in the major thesis, University of Central Florida (2010)
11. Pejsa, T., Pandzic, I.: State of the art in example-based motion synthesis for virtual characters in interactive applications. Computer Graphics Forum **29**(1), 202–226 (2010), http://dx.doi.org/10.1111/j.1467-8659.2009.01591.x
12. Reil, T., Husbands, P.: Evolution of central pattern generators for bipedal walking in a real-time physics environment. IEEE Transactions on Evolutionary Computation **6**(2), 159–168 (Apr 2002). https://doi.org/10.1109/4235.996015
13. Sims, K.: Evolving virtual creatures. In: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques. pp. 15–22. SIGGRAPH '94, ACM, New York, NY, USA (1994), http://doi.acm.org/10.1145/192161.192167
14. Valsalam, V.K., Miikkulainen, R.: Modular neuroevolution for multilegged locomotion. In: Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2008. pp. 265–272. ACM, New York, NY, USA (2008), http://nn.cs.utexas.edu/?valsalam:gecco08
15. Yosinski, J., Clune, J., Hidalgo, D., Nguyen, S., Zagal, J.C., Lipson, H.: Evolving robot gaits in hardware: the HyperNEAT generative encoding vs. parameter optimization. In: In Proceedings of the 20th European Conference on Artificial Life. pp. 890–897 (2011)