INDIVIDUAL ASSIGNMENT

RESEARCH METHODS FOR COMPUTING AND TECHNOLOGY

Student Name	Cheng Yi Heng
TP Number	TP058994
Intake Code	APU2F2305CGD
Module Code	CT09832RMCT
Lecturer Name	Assoc. Prof. Ts. Dr. Tan Chin Ike
Hand Out Date	7th November 2023
Hand In Date	19th February 2024

Exploring Deep Learning Approaches for Real-Time Interactive Character Animation

Yi Heng Cheng

voxell.technologies@gmail.com

1. Literature Review

1.1. Introduction

Animation has evolved significantly from its origins to become a cornerstone of entertainment and communication. This review delves into 3 three key areas. The first section will discuss about the evolution of animation. History is extremely important when it comes to understanding a field. This section will bring about the advancement in animation over the past decades, providing a timeline and the potential future trajectory. The following section will be a deep dive into the research that has been done to create the technologies needed to make interactive character animation possible. Understanding the underlying technologies that made interactive animation possible is key to new innovations. The final section will be about the recent advancements of deep learning that could propel the interactive character animation standards. This section explores what had been done and the potential future of the interactive characater animation industry. The type of sources that will be used in this chapter will primarily come from research articles.

1.2. Evolution of Animation

Animation in its simplest form is a sequence of actions that when played in a sequential manner, produces an illusion of movement. In the beginning, all animations are offline or pre-recorded in some form before displaying it to its audience. As oppose to many real-time animations today, especially in games, offline animations are not interactable, and thus only fits the purpose of the film industry.

Animation production started off with hand-drawn animations. From the 1940s to the 1980s, hand-drawn animations was the main mode of output in the animation industry (Lamotte, 2022). An animation is produced frame-by-frame, requiring prodigious quanitites of labor for the construction of a 24 frames per second film (Baecker, 1969). It was slow and ineffective, but it was the only choice given the state of the technology at that time.

The emergence of computer-assisted animation started gaining popularity during the 1970s (Lamotte, 2022). Computer graphics systems strive to create a better experience to replace the drawing and painting process, widely known as the “Ink and Paint” process at that time. TicTacToon was a method that proposes a paperless 2D animation production line (Fekete et al., 1995). Motion capture was also introduced using potentiometers to track the movement of the human body (Gleicher, 1999; Sturman, 1994) .

Starting around 2000s, purely computer generated images (CGI) has started to become possible. Computer graphics systems had evolved to be able to render 3D scenes. A tremendous improvement in CGI can be seen from the film *Tron: Legacy* which was released in December 2010 and in production since 2009. There were also released 3D games with 3D interactive animations like *Halo: Combat Evolved*, *Gears of War*, and *Half-Life*. This marks a significant change in the animation industry.

1.3. Interactive Character Animation

Interactive character animation is made up of multiple underlying technologies. It is a subset of animation where characters are typically animated using a rig which deforms a mesh made up of triangles that is rendered onto the screen in real-time. The end goal is to create a system that is capable of providing visual feedback of character movements for users in real-time applications.

1.3.1. Mesh Skinning

Skinning is the process of performing mesh deformation according to a function of skeletal poses (Rumman & Fratarcangeli, 2016). In character animation, it is important to adopt a skinning method that is high in fidelity and performance. This section will explore the various methods of skinning that had been developed over the years as well as their pros and cons.

Linear Blend Skinning (LBS) is a commonly used method in character animation where each vertex of the character mesh is influenced by a weighted sum of the transformations of nearby bones (Lander, 1998). It is being used in *AAA* game engines like Unity3D and Unreal Engine. LBS is known for its fast and simple algorithm that maps advantageously to the graphics hardware.

Spherical Blend Skinning (SBS) is another form of skinning method that employs spherical interpolation to smoothly blend between bone transformations (Kavan & Žára, 2005). SBS aims to solve the "lost of volume" artifact that LBS brings despite its efficient algorithm.

To solve the computational and memory overhead that SBS brings, Kavan et al. (2007) propose Dual Quaternion Blending (DQB). DQB uses dual quaternions to represent both translation and rotation, allowing for more accurate and natural deformations of the character mesh. Unlike SBS, it does not require additional memory to cache rotation centers. The DQB method is also extremely efficient. However, DBQ comes with a limitation, it only supports rigid transformation and is not suitable for scaling or shearing effects.

1.3.2. Inverse Kinematics

Inverse kinematics (IK) is widely used in video games and robotics to create realistic poses within a defined constraint. In short, the ultimate goal of IK is to determine an appropriate joint configuration that allow the end effectors to reach a target position (Aristidou et al., 2018).

One use case of IK is to perform animation retargeting to map movements between characters with different proportions (Molla et al., 2017). In the context of interactive applications like games, IK can also be used to perform secondary motions on top of an already playing animation (Ruuskanen, 2018). For example, turning the head towards an interest point, or moving the hand towards a target position.

At its current state, there is a total of 4 main categories towards IK:

1. **Analytical**

   Analytical IK solvers aim to determine all potential solutions based on mechanism lengths, initial posture, and rotation constraints. They often rely on assumptions to compute a single solution.

2. **Numerical**

   Numerical methods often require a set of iterations to achieve a satisfactory approximation by minimizing a predefined cost function.

3. **Data-Driven**

Data-driven methods relies on large accurate animation databases. Most data-driven methods employs some kind of machine learning algorithms to learn from the dataset.

4. **Hybrid**

   The hybrid method is simply a way of combining 2 or more different IK methods into a single solution.

1.3.3. Physics Based Character Animation

Physics based character animation offers a completely new solution for developers to prioritize physics accuracy over animation precision. It forces characters to obey the laws of physics like preventing collisions between collidable objects and interacting with external forces such as gravity, pressure, etc (Ye Hu, 2016).

Authoring physics based character animation can be extremely hard. This is due to the unpredictability of the physical world. For example, a character might accidentally get hit by a physical object during runtime, resulting in unexpected movements or behaviors that can disrupt the intended animation sequence.

A major limitation of physics based animation is the inability to precisely control the artistic intent for achieving specific visual effect. Additionally, ensuring computational efficiency while simulating complex physical interactions adds another layer of challenge to the authoring process.

1.3.4. Animation System

Multiple animation clips are normally used in interactive environments to create a variety of dynamic motions. In a conference talk by Holden (2018), he mentioned that Assassin's Creed Origins had around 15,000 animations in the game. These animations are needed to be handled by an animation system to systematically select the correct animation clips to sample depending on the current scenario.

Game engines like Unity3D uses a hierarchical state machine (HSM) graph, shown in Figure 1. It controls the sampling of animation clips and the transition between them. This allows developers to divide complex systems into smaller isolated modules (Berg, 2023). During runtime, the animation system will traverse the state machine graph and subsequently transition to the animation clip it reaches.

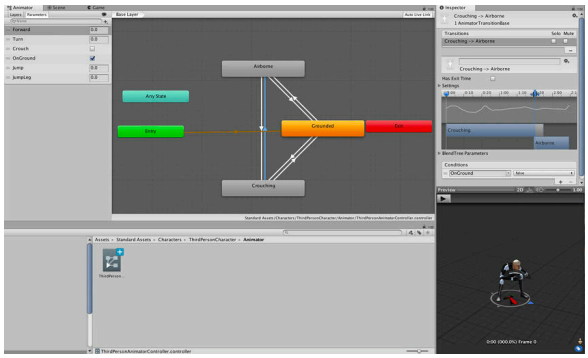

Figure 1: Unity's Mechanim

In some cases, animators would also like to mix and match different animation clips together. For example, an in between animation of walking and jogging to produce a slow jog. This can be achieved using a method called blend trees (Berg, 2023).

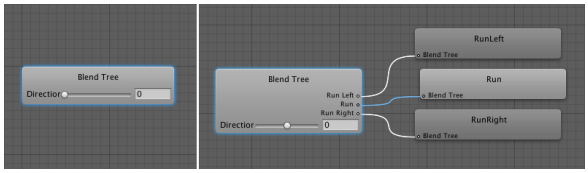

Figure 2: Blend tree.

Another method known as motion blending is also used to apply motion trajectories onto the rig, based on a weighted sum of multiple animation clips (Ménardais et al., 2004). This can create interesting motion dynamics like a walking animation clip towards the lower body part and a punching animation towards the upper body part.

1.4. Deep Learning in Animation

1.4.1. A Brief History of Deep Learning

In 1943, neurophysiologist Warren McCulloch and mathematician Walter Pitts published an article titled *"A Logical Calculus of the Ideas Immanent in Nervous Activity"* (McCulloch & Pitts, 1943). In it, they described how neurons in brain might work and modeled a simple neural network using electrical circuits. It was an attempt to understand how the human brain works and more importantly, how it learns.

The first artificial neural network (ANN) called Perceptron was invented by Frank Rosenblatt (Rosenblatt, 1958). Perceptron was able to model functions that are determined by linearly separable data. Activation functions were used to introduce non-linearity into the network, e.g. Sigmoid, ReLU, and Leaky ReLU (Sharma et al., 2017).

Deep neural networks can solve many hard computational tasks like image recognition using convolutional neural networks (CNN) (LeCun et al., 1998). Recurrent neural networks (RNN) were also introduced to tackle sequential data (Rumelhart et al., 1985). An improved version of RNN known as the Long Short-Term Memory (LSTM) was proposed to solve more complex and longer sequential tasks (Hochreiter & Schmidhuber, 2010).

In an article titled *"Attention Is All You Need"*, the authors revolutionized the deep learning industry by introducing the Transformer model (Vaswani et al., 2017). The Transformer model is capable of performing all kinds of tasks from learning sequential data for human conversation like Llama 2 to speech recognition like Whisper (Radford et al., 2023; Touvron et al., 2023) .

1.4.2. Using Deep Learning to Drive Character Animation

1.4.2.1. Motion Matching

Traditional HSM methods tightly couples the animation data with the states (Holden, 2018). A better approach to this problem is to store all animation data into a database and tag them with their related traits, e.g. walk, idle, run. etc. Immediately, all of the animation data now form a relation based on similar tags they share. Getting an animation from the database can be done by querying the specifc traits that is needed.

To improve this system further, character data and important animation state can also be incorporated as traits in the animation database, e.g. velocity of the character, location of the hip bone, etc. This way, getting a specific animation clip becomes more like a matching system rather than a query system. Developers can now find the best match animation based on the current and desired character state alone, without the need of any state machines. This method is widely known as motion matching, a data-driven approach towards character animation (Büttner & Clavet, 2015).

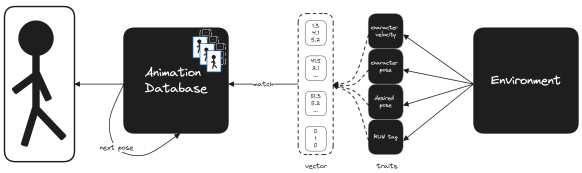

Figure 3: Motion matching.

1.4.2.2. Deep Learning

The key idea of using neural networks is to attempt to generalize the problem and solve the scalability issue of many traditional animation systems (Holden, 2018). Other than scalability, neural networks also proof to be fast and memory efficient. No animation data is required during runtime when inferencing a sufficiently trained neural network.

Phase-Functioned Neural Network (PFNN) introduces the idea of using a phase function to generate weights for a regression network which in turn generates the animation (Holden et al., 2017). Not all animations have a phase, in addition, labeling the phase variable to each animation can be a laborious task. To solve this, a new neural network architecture called Mode-Adaptive Neural Networks (MANN) is proposed to remove the phase label and replace it with a gating network (Zhang et al., 2018). Local Motion Phases (LMP) removes the dependency of a global phase variable in favor of multiple independent local phases for each bone (Starke et al., 2020). The local phase is defined based on the contact between each bone and the environment. Instead of defining phases manually, Starke et al. (2022) proposes Deep Phase, a periodic autoencoders for learning motion phase manifolds automatically. The authors stated that the learned motion phase can also potentially be used for motion matching and reinforcement learning. Learned motion matching presents a learned alternative towards the highly flexbily and low pre-processing time method, motion matching. It promises to retain high quality animation data and quick iteration time while preserving the scalability of a neural network approach (Holden et al., 2020).

Deep learning also tremendously benefit the world of physics based animation. Deep Mimic uses deep reinforcement learning (RL) to learn control policies to imitate a variety of animation clips in a fully physics simulated scenario (Peng et al., 2018). Peng et al. (2021) proposes a fully automated adversarial RL system for physics based character animation to imitate behaviors from unstructured dataset. In the following year, a better approach was introduced to allow physically simulated characters to learn reusable skill embeddings from large dataset of unstructured motion clips (Peng et al., 2022).

1.5. Conclusion

Creating scalable interactive character animation system is still a challenging and on going research topic. While deep learning solves a lot of the scalability issue, every little change to the animation data would require an update towards the network. Compared to systems like HSM and motion matching, this update is not instantaneous and might require a huge amount of computational power. Besides, using new systems like motion matching and neural networks require animators to adapt and learn a new set of skills. Further studies is needed to create highly scalable animation systems that can retain rich animation motion while shortening or even removing the retraining process of neural networks.

2. Methodology

2.1. Introduction

This chapter will discuss about the target users that are suitable for this research as well as providing insights into the strategies used to sample and collect data, and ultimately draw meaningful conclusions regarding the efficacy of deep learning for real-time interactive character animation.

2.2. Target user

The target users for this study encompass three primary groups: game developers, animators, and gamers. Understanding the perspectives and requirements of each group is essential for designing deep learning approaches that are effective, user-friendly, and capable of enhancing the interactive character animation experience across various platforms.

1. **Game Developers**: These professionals play a crucial role in integrating

character animation into interactive gaming environments. Their insights are invaluable for understanding technical constraints, performance requirements, and integration challenges associated with deploying deep learning techniques in real-time scenarios.

2. **Animators**: Animators possess expertise in crafting compelling character animations that resonate with audiences. Their input is essential for evaluating the artistic quality, expressiveness, and fidelity of animations generated using deep learning methods.
3. **AI Researchers**: Creating effective deep neural network architectures is not a simple task. Their expertise can provide valuable insights into the latest advancements, methodologies, and challenges in applying deep learning techniques to character animation.
4. **Gamers**: Ultimately, the success of interactive character animation lies in its reception by gamers (or interactive application users). Understanding their preferences, expectations, and experiences with character animations can provide valuable feedback on the effectiveness and immersion of deep learning-driven animations.

2.3. Sampling method

We primarily focuses on using purposive sampling method so that participants can be selected based on their expertise and involvement in the field of character animation and interactive applications. This method ensures that only individuals with relevant knowledge and experience are included in the study. A total of 40 participants will be sampled, comprising 10 game developers, 10 animators, 10 AI researchers, and 10 gamers. This sampling approach ensures that diverse perspectives are represented, enhancing the richness and depth of the data collected.

2.4. Data collection method

The data collection method used invovles qualitative interviews. In-depth interviews will be conducted with each participant, utilizing a semi-structured approach to explore their perspectives, challenges, and expectations regarding real-time interactive character animation and the potential role of deep learning techniques. During these interviews, it is also allow for the flexibility to delve into more specifc topic of interest, providing rich insights regarding the area of expertise of the interviewee.

2.5. Conclusion

By employing purposive sampling and qualitative data collection techniques, this chapter aims to gather comprehensive insights from game developers, animators, and gamers. These insights will inform the development and refinement of deep learning techniques, ultimately enhancing the quality, realism, and interactivity of character animations in gaming and interactive media environments.

3. References

Aristidou, A., Lasenby, J., Chrysanthou, Y., & Shamir, A. (2018). Inverse kinematics techniques in computer graphics: A survey. *Computer Graphics Forum*, *6*, 35–58.

Baecker, R. M. (1969). Picture-driven animation. *Proceedings of the May 14-16, 1969, Spring Joint Computer Conference*, 273–288.

Berg, J. (2023). Animation Graph. *Lulea University of Technology*.

Büttner, M., & Clavet, S. (2015). Motion matching-the road to next gen animation. *Proc. Of Nucl. Ai*, *2015*, 2.

Clavet, S. (2016). Motion matching and the road to next-gen animation. *Proc. Of GDC*, *4*, 9.

Fekete, J.-D., Bizouarn, É., Cournarie, É., Galas, T., & Taillefer, F. (1995). TicTacToon: A paperless system for professional 2D animation. *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, 79–90.

Gleicher, M. (1999). Animation from observation: Motion capture and motion editing. *ACM SIGGRAPH Computer Graphics*, *4*, 51–54.

Harvey, F. G., Yurick, M., Nowrouzezahrai, D., & Pal, C. (2020). Robust motion in-betweening. *ACM Transactions on Graphics (TOG)*, *4*, 60–1.

Hochreiter, S., & Schmidhuber, J. (2010). Long short-term memory. *Neural Computation*, *8*, 1735–1780.

Holden, D. (2018). Character control with neural networks and machine learning. *Proc. Of GDC 2018*, 2.

Holden, D., Kanoun, O., Perepichka, M., & Popa, T. (2020). Learned motion matching. *ACM Transactions on Graphics (TOG)*, *4*, 53–1.

Holden, D., Komura, T., & Saito, J. (2017). Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)*, *4*, 1–13.

Kavan, L., & Žára, J. (2005). Spherical blend skinning: a real-time deformation of articulated models. *Proceedings of the 2005 Symposium on Interactive 3d Graphics and Games*, 9–16.

Kavan, L., Collins, S., Žára, J., & O'Sullivan, C. (2007). Skinning with dual quaternions. *Proceedings of the 2007 Symposium on Interactive 3d Graphics and Games*, 39–46.

Lamotte, C. (2022). Discovering Animation Manuals: Their Place and Role in the History of Animation. *Animation*, *1*, 127–143.

Lander, J. (1998). Skin them bones: Game programming for the web generation. *Game Developer Magazine*, *1*, 10–18.

Lasseter, J. (1998). Principles of traditional animation applied to 3D computer animation. *Seminal Graphics: Pioneering Efforts That Shaped the Field*, 263–272.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *11*, 2278–2324.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 115–133.

Menolotto, M., Komaris, D.-S., Tedesco, S., O'Flynn, B., & Walsh, M. (2020). Motion capture technology in industrial applications: A systematic review. *Sensors*, *19*, 5687.

Molla, E., Debarba, H. G., & Boulic, R. (2017). Egocentric mapping of body surface constraints. *IEEE Transactions on Visualization and Computer Graphics*, *7*, 2089–2102.

Ménardais, S., Multon, F., Kulpa, R., & Arnaldi, B. (2004). Motion blending for real-time animation while accounting for the environment. *Proceedings Computer Graphics International, 2004.*, 156–159.

Peng, X. B., Abbeel, P., Levine, S., & Panne, M. Van de. (2018). Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, *4*, 1–14.

Peng, X. B., Guo, Y., Halper, L., Levine, S., & Fidler, S. (2022). Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions on Graphics (TOG)*, *4*, 1–17.

Peng, X. B., Ma, Z., Abbeel, P., Levine, S., & Kanazawa, A. (2021). Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (Tog)*, *4*, 1–20.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023). Robust speech recognition via large-scale weak supervision. *International Conference on Machine Learning*, 28492–28518.

Rose III, C. F., Sloan, P.-P. J., & Cohen, M. F. (2001). Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum*, *3*, 239–250.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, *6*, 386.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation. *Institute for Cognitive Science, University of California, San Diego La~*.

Rumman, N. A., & Fratarcangeli, M. (2016). State of the art in skinning techniques for articulated deformable characters. *International Conference on Computer Graphics Theory and Applications*, 200–212.

Ruuskanen, A. (2018). *Inverse Kinematics in Game Character Animation*.

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. *Towards Data Sci*, *12*, 310–316.

Starke, S., Mason, I., & Komura, T. (2022). DeepPhase: Periodic autoencoders for learning motion phase manifolds. *ACM Transactions on Graphics (TOG)*, *4*, 1–13.

Starke, S., Zhao, Y., Komura, T., & Zaman, K. (2020). Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics (TOG)*, *4*, 54–1.

Sturman, D. J. (1994). A brief history of motion capture for computer character animation. *Siggraph94, Course9*.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., & others. (2023). Llama 2: Open foundation and fine-tuned chat models. *Arxiv Preprint Arxiv:2307.09288*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

Ye Hu, L. (2016). Physics-based character animation. *Universitat PolitèCnica De Catalunya*.

Zhang, H., Starke, S., Komura, T., & Saito, J. (2018). Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)*, *4*, 1–11.