USING FOURIER ANALYSIS TO GENERATE BELIEVABLE GAIT PATTERNS FOR VIRTUAL QUADRUPEDS

A Thesis

by

SPENCER MCDOW CURETON

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Ann McNamara
Committee Members,	Tim McLaughlin
	Takis Zourntos
Head of Department,	Tim McLaughlin

May 2013

Major Subject: Visualization

ABSTRACT

Achieving a believable gait pattern for a virtual quadrupedal character requires a significant time investment from an animator. This thesis presents a prototype system for creating a foundational layer of natural-looking animation to serve as a starting point for an animator. Starting with video of an actual horse walking, joints are animated over the footage to create a rotoscoped animation. This animation represents the animal's natural motion. Joint angle values for the legs are sampled per frame of the animation and conditioned for Fourier analysis. The Fast Fourier Transform provides frequency information that is used to create mathematical descriptions of each joint's movement. A model representing the horse's overall gait pattern is created once each of the leg joints has been analyzed and defined. Lastly, a new rig for a virtual quadruped is created and its leg joints are animated using the gait pattern model derived through the analysis.

DEDICATION

To family, friends, and most importantly my loving wife, Catherine.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Ann McNamara, for her support, guidance, and patience through this research. Her help has been greatly appreciated. I would also like to thank Tim McLaughlin for allowing me to join the perception based animation research group, which exposed me to the research of quadrupedal gait generation and the influence perception has over the animation process. I should mention Ariel Chisholm, Junze Zhou, and Jorge Cereijo for their help in brainstorming and troubleshooting. I've been really fortunate to work around such intelligent and passionate people.

Dr. Takis Zourntos has been especially helpful in answering my questions over Fourier analysis and digital signal processing, not to mention just being a great friend. This project would not have made it as far as it did without everyone's support.

Thank you.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

Showcasing quadrupedal creatures (quadrupeds) is becoming increasingly popular in feature animation and live-action movies. For example, The Chronicles of Narnia [1], the Shrek and Madagascar series, and video games such as Red Dead Redemption, Cabela's African Safari, and Assassin's Creed [8] feature quadrupeds with believable gait patterns. The time invested to achieve a believable gait pattern is significant – on the order of several days. Animators spend a large amount of time and effort to produce a foundational gait pattern before they can begin directing the more complex motion and expressiveness of the character that is necessary to the performance.

Commercial software solutions exist that can procedurally generate quadrupedal gaits for virtual characters [9]. However, most of these systems output gait cycles that are less realistic and are usually perceived as mechanical and robotic [2]. There is a strong need for a more reliable system that consistently generates believable motion for quadrupeds.

Successfully generating believable motion depends on modeling natural locomotion of real creatures. This thesis describes a new methodology for extracting gait data for a walking quadruped from video and creating a model of its natural motion. The horse was the sole quadruped examined, because horses are relatively easy to film and work with. Their short hair allows the joint movements in the legs to be easily observed,

and so the front and rear leg joints were tracked to create an animation from which raw joint motion data was sampled.

Each joint in a horse's leg moves periodically and can be thought of as a periodic signal. For example, a joint's rotation could be described as angle values changing across time. Calculating the angle at each frame of the animation is equivalent to sampling the signal. Once sampled, values were analyzed with the fast Fourier transform (FFT) for each of the leg joints and their respective motion, listed in Table 1. The FFT is an efficient form of the discrete Fourier transform (DFT) used to view the integral components of the sampled signal's frequency domain. This frequency domain clearly presents the significant frequencies existing in a signal.

Table 1. Front and rear leg joint motion parameters that will be analyzed.

Front Leg Joints	Rear Leg Joints	Motion Parameter Analyzed
Shoulder	Hip	Rotation, Translation
Elbow	Stifle	Rotation
Knee	Hock	Rotation
Ankle	Ankle	Rotation

A mathematical model (expression) was derived through a form of the Fourier series, which uses the frequency components obtained from the FFT. This mathematical model describes a joint's motion across time. Due to the periodicity of each signal, this method was able to provide approximated expressions for each joint's motion (Table 1). These expressions were used to animate a virtual quadruped's skeletal rig in Autodesk's Maya. Maya is a 3D animation, visual effects, and compositing software standard in the industry [10]. The approach comprises three phases as described below.

Phase I: Sampling Locomotion Data

The locomotion data for a walking horse is sampled from actual video reference. The footage provides a sagittal (side) view of the horse, showing two of the three dimensions of movement. This perspective supplies the most information regarding each leg's joint motion [4]. Each joint's rotational motion and the leg's translational motion were observed from this perspective. Due to the symmetrical nature of the walk gait, examining the motion for one side provides enough information for all four legs.

Figure 1. Six frames from the image sequence. The sequence is imported into Maya's image plane for rotoscope creation. The horse's gait pattern is clearly seen from the side perspective.

The video was converted into an image sequence, as seen in Figure 1, and imported into Maya as the image plane for an orthogonal camera. Joints were then manually animated over the video's image sequence to create a rotoscoped animation, which is shown in Figure 2. This resulted in an animation representing the animal's natural motion.

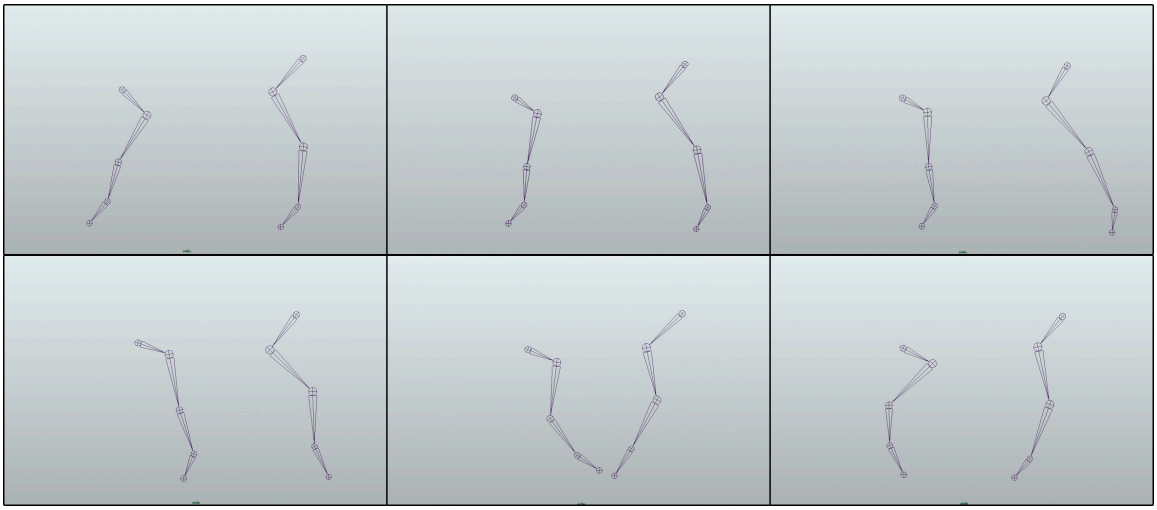

Figure 2. Six frames from the rotoscoped animation. Joints were rotoscoped in Maya 2012 over the same six frames of the reference video from Figure 1. Only the front and rear leg joints facing the camera are included in the animation.

Each joint's rotation values for the front leg and rear leg were sampled at each frame of the rotoscoped animation and exported in the form of a text file. Translation values were also sampled for the horse's shoulder and hip movement relative to a locator representing the center of mass.

These sampled signals were conditioned in preparation for Fourier analysis. Conditioning consisted of removing the bias and manually selecting values within the sampled set, which characterize the signal. The selected values must characterize one period of the sampled signal. These values were repeated to simulate a periodic extension, thus creating a new representative of the originally sampled signal. The conditioned values for each signal were referred to as the adjusted signal.

Phase II: Signal Analysis

Phase II focused on analyzing the adjusted signal for each joint's motion in order to obtain a mathematical expression.

Each joint's adjusted signal values from Phase 1 were upsampled to 32768 values to ensure coherent results from the FFT. The FFT was performed on the upsampled values in order to acquire information about the signal in the frequency domain. These results were used to better understand which frequencies had more significance in the given signal. Ultimately, the FFT results were used in a truncated form of the Fourier series to approximate the mathematical expression for the signal in question. This method was performed for the front and rear leg joints to create an overall mathematical model for the horse's gait pattern. The combination of the signals represented the believable gait pattern.

The results from this phase served as inputs to the following phase.

Phase III: Synthesizing Believable Gait Patterns

The character's motion system of hierarchical kinematic joints (referred to commonly as an animation rig) [28] was defined so that the data from Phase II could be used to generate the desired believable gait pattern. Generally, the process of rigging serves to make the animation process more intuitive through the design of the control interface. The definition of rigging from Maya's online documentation states [10]:

> Rigging a character, also known as character setup, involves creating skeletons and IK handles for your characters, binding skins to the skeletons, and setting up deformers and constraints.

In this project, the quadruped rig was scripted for efficient management. A script is a program designed to alleviate the programmer's tasks by automating functions that would otherwise be performed one-by-one. The rig consists of four legs, each with appropriate joint and control object hierarchy. The leg joint hierarchies were created by duplicating the front and rear joint chains used for the rotoscoping process. A control hierarchy was established with constraints connected to the joints for proper control over the leg's motion.

Using the expressions derived during the analysis phase, a synthetic animation was created for the four legs. The joints on the right side of the rig differed from their left counterparts by a time shift equal to half the time of the horse's stride length. Key frames were set for each joint's rotation value, specified by the abovementioned expression. The culmination of the joints produced a locomotion cycle mimicking the original gait from the reference video. Overall, the process produced four synthetically animated legs, driven by mathematical functions achieved through Fourier analysis of natural motion extracted from video reference.

The ultimate goal of this research was to provide a prototype intended to aid in the animation of quadrupedal characters for novice and experienced users by providing a foundation of a believable gait pattern. This paper provides a detailed description of the methodology along with the results and ideas toward future work.

The remainder of this thesis is structured as follows:

Chapter II: Describes the prior work in related research.

Chapter III: Details the process of data extraction to obtain samples from the natural gait pattern of the horse.

Chapter IV: Creates a mathematical model of believable gait patterns through the FFT.

Chapter V: Explains the process of creating a rig and applying the gait pattern models to create a believable animation.

Chapter VI: Draws conclusions and outlines avenues for future research.

CHAPTER II

BACKGROUND

This chapter describes prior related research in the areas of gait analysis and gait synthesis for quadrupeds. Natural gait patterns, from real quadrupeds, must be studied and analyzed before creating a synthetic gait pattern that will pass as believable.

Prior research has demonstrated that successful methods exist for analyzing and synthesizing gait patterns for virtual quadrupedal characters. The method of analysis creates the data model that represents the natural motion of the reference animal. The model is applied to a virtual character to produce a final animation.

Gait Analysis

Gait analysis is the process of breaking apart the complexity of a quadruped's gait pattern into smaller components in order to obtain a better understanding of the overall process. Several methods exist which study and model quadrupedal gait patterns including computer vision, motion capture (mocap), rotoscoping, and biomechanical analysis. The underlying goal of these different approaches is to obtain a model that represents realistic motion. This model can then be applied as a basis for animation of another, possibly new, character.

Mocap is the "process of recording a live motion event and translating it into usable mathematical terms by tracking a number of key points in space over time and combining them to obtain a single three-dimensional (3D) representation of the

performance [11].” This technology has proven to be a great tool for creating performance-based character animation. Variations of this technology have been used in films like The Lord of the Rings, Avatar, and The Adventures of Tintin. Sports video games such as Tony Hawk’s Project 8 [12] and Tiger Woods PGA Tour 13 [13] have also incorporated motion capture in order to transfer an actor’s performance to another character. Depending on the needs of the production, a single camera or multiple cameras [14] can be used to capture the desired performance. However, a single camera motion capture system is limiting since only a fraction of the 3D motion data can be represented. Also, motion capture systems tend to produce noisy data that requires filtering before it can be used for animation.

Generally, the acquired data from a mocap system describes the “mechanical aspects of rigid segments [15].” The system processes the captured footage, providing coordinate points as raw data rather than an image sequence representation. The processed motion data can then drive the animation of a virtual character similar to that of the captured performer. Polar Express implemented an advanced facial tracking system to capture Tom Hanks’ performance [14]. It can also translate to entirely different characters as seen in Happy Feet [15] and The Chronicles of Narnia [1].

However, motion capture does have its limitations for its use in movies and video games. Capturing an animal’s performance is considerably more difficult than capturing a human’s. Also, an actor’s captured performance is very actor specific and typically not generalizable. Lastly, the motion data affords very limited manipulation to modify the performance.

Commercial endeavors such as Qualisys [16] employ motion capture to study the biomechanics of horses and dogs. Domestic animals are used, as they are easy to control. It is impractical to use motion capture systems on wild animals in their natural environments. Skrba and Reveret state that the motion capture environment "can produce uncharacteristic motion – walking on grass is different to walking on a treadmill, and it is unsuitable when working with wild animals [2]."

To combat the difficulties of capturing wild creature motion, alternative strategies have been devised [2,4]. Creatures can be filmed in their natural environment and the footage can be studied and used as reference for animation. A simple, yet time-consuming method that utilizes video reference is rotoscoping. Traditionally, rotoscoping consisted of a "video projected one frame at a time onto a glass screen. A sheet of translucent paper is placed on top of the glass and the image of the video frame is traced onto the sheet [2]". Figure 3 is a drawing from Max Fleischer's patent for the original rotoscope system [26]. Technology has eased the process of rotoscoping, allowing the frames to be traced over digitally with various software packages. Rotoscoping provides a method to copy an actor's performance onto another character.

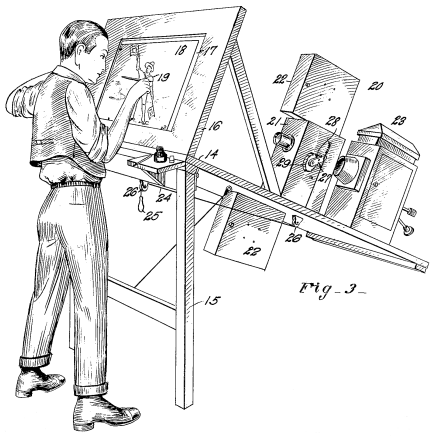

Figure 3. Max Fleischer's rotoscoping system. The drawing describes the original rotoscoping system patented in 1917. It was designed to create lifelike cartoons based on a performance captured on video. [26]

This technique has been used in many feature films including Disney's Snow White, Fantasia, and Mary Poppins [2]. Disney animators filmed actors for reference and realized that the film itself could be studied frame-by-frame "to reveal the intricacies of a living form's actions [17]." The resulting animations appeared very natural and lifelike and became examples of their animation principles [17]. However, rotoscoping only provides an animation that directly mimics the performance from the film.

Advancements in computer vision have alleviated the rotoscoping process by automatically tracking the performer's actions from video. Favreau et al. present a methodology that "allows the extraction of 3D cyclic motion of animals from arbitrary video sequences [3]." Favreau et al. were successful in their process by conducting a Principal Component Analysis (PCA) on a sequence of binary images (video frames)

and then using Radial Basis Functions (RBF) to interpolate the data, resulting in a 3D animation [3]. Figure 4 provides an overview for their process.

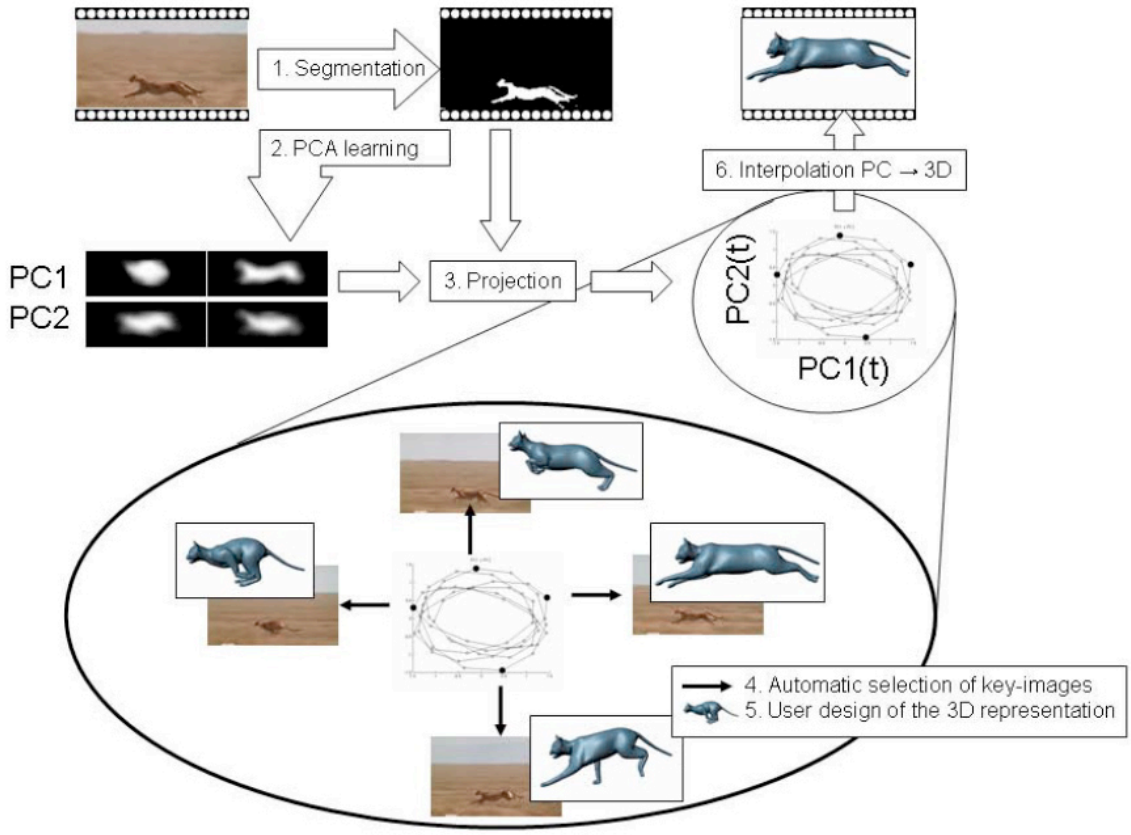

Figure 4. Overview of Favreau et al. research. Video was analyzed with PCA and a RBF interpolation technique to drive the locomotion of a 3D virtual character. [3]

Wilhelms and Van Gelder were also successful in extracting 3D "motion [data] of humans and animals from unrestricted monocular video [4]." A generic model is used along with active contours to track the motion of the animal in the video. "Active

contours are sets of connected control points associated with features (usually edges) in the underlying image [4]". They provide a means of tracking a moving object across an image sequence. However, the process is not perfect and requires a user to set the contours at different key frames throughout the sequence. A 3D model is attached to the tracked contours resulting in the animation. Few other researchers have been successful in animating a 3D model of a virtual quadrupedal character with a model derived from video analysis [3,4]. Obtaining a 3D representation of a quadruped's motion from video is nearly impossible due to the nature of the single camera perspective and its lack of 3D information. Also, the resulting animation offers little flexibility for an animator to make expressive adjustments. This method has not been adopted by the entertainment industry due to these drawbacks.

Mathematical models representing natural gait patterns have been formed within related research. Statistical analysis and learning machines are two examples of models created through gait analysis. Mocap data is leveraged and statistically analyzed to extract key elements that form a probabilistic model of the motion [5]. Researchers have used learning machines [5] and neural networks [6] to learn the pattern of motion captured data, which allowed synthesis of unique, expressive gaits. Brand and Hertzmann [5] created a statistical model able to "generate new motion sequences in a broad range of styles" by learning from motion capture samples. They produced a state-space model, built on the structure of bipedal locomotion, and incorporated a Hidden Markov Model (HMM) to generate a time-series data model. Numerous motion capture examples were analyzed with their method to create style machines specific to the

captured performance, which could then be used to animate a 3D virtual character. Unfortunately, this method is impractical to implement with quadrupeds in their natural environments due to the difficulty of establishing a controlled motion capture environment. However, other forms of mathematical analysis can be conducted with data extracted from video in order to produce a model of an animal's motion.

Frequency analysis is another method used to construct a representative model of gait motion. In Hoffmann and Düffer's research [18], Fourier analysis was conducted to create a frequency space model able to drive the controller commands for a quadrupedal robot's gait. Since "quadruped locomotion is a periodic process, describing each joint's motion over one period fully describes the motion [18]." Each joint's motion was defined with a frequency space representation, which provided a clearer understanding of the gait and better means of applying the necessary signals to generate appropriate motion.

Unuma et al. [19] describe a similar approach. Unique human figure locomotions were modeled through Fourier expansions. A mocap system provided motion data in the form of rotational joint angle values. Due to the periodicity of the motion, Fourier series expansion was used to create a functional model of the joint angle motion. The entire bipedal character was analyzed and its resulting model was used to drive a virtual character. This proved to be a very effective method for modeling bipedal motion, however the resulting locomotion appeared less than natural, especially during the transitions from run to walk. Achieving natural motion requires an understanding of a creature's physical constraints and their biomechanical limitations in the real world.

The study of biomechanics has been a focus for robotics researchers and those interested in creating physically based simulations of quadruped locomotion. Robert McNeill Alexander, a well-noted British zoologist, has written multiple books and papers covering the field of creature locomotion. His paper, “The Gaits of Bipedal and Quadrupedal Animals”, covers the definitions of stride length, gait speed, duty factor, and other parameters that contribute toward quantifying an animal’s gait [7]. Alexander “developed a hypothesis about the relationship between size, speed, mass and external forces" which states that “two bodies can be described as dynamically similar if the motion of one can be made identical to that of the other by multiplying a) all linear dimensions by some constant factor, b) all time intervals by another constant factor, and c) all forces by a third factor [2].” A solid understanding of the biomechanical properties of a real quadruped provides the background needed to construct a prototype in a physically based simulation, which will be discussed in the following section.

Typically, robotics engineers work toward creating mechanical models of four-legged robots with movements similar to quadrupedal animals. However, the actual focus lies on the control design and the robot’s ability to perform tasks wheeled robots have trouble with [20]. Quadrupedal gait patterns are analyzed since a quadruped serves as a model for the robot’s desired locomotion. The analysis provides a mathematical model used to drive the robot [18, 21].

In one form or another, gait analysis provides a mathematical model to serve as the driving component in a synthetic gait.

Gait Synthesis

Gait synthesis refers to the animation of a virtual character using the model derived from the analysis of a real quadruped's gait pattern. A few approaches exist for creating synthetic gait patterns and they depend on the resulting model from the analysis.

As discussed in the previous section, mocap techniques, rotoscoping, and computer vision are among the popular methods for gait analysis. Traditionally, animators brought life to their characters by studying the natural motion of the animal, either directly in the presence of the animal, or through video reference [17]. The following research methods describe more of an automated approach to generating a realistic animation.

The resulting animation from motion capture and rotoscoping are very similar. An actor's performance is captured and used, nearly directly, on a virtual character [1, 14]. Motion capture data might be conditioned, depending on the system, in order to control the virtual character's parameters appropriately [15]. For rotoscoping, Disney animators were successful in animating characters that appeared realistic and lifelike, as shown in Figure 5. The results were actually more life-like than expected since more of the actor's subtleties were animated due to the rotoscoping process [17].

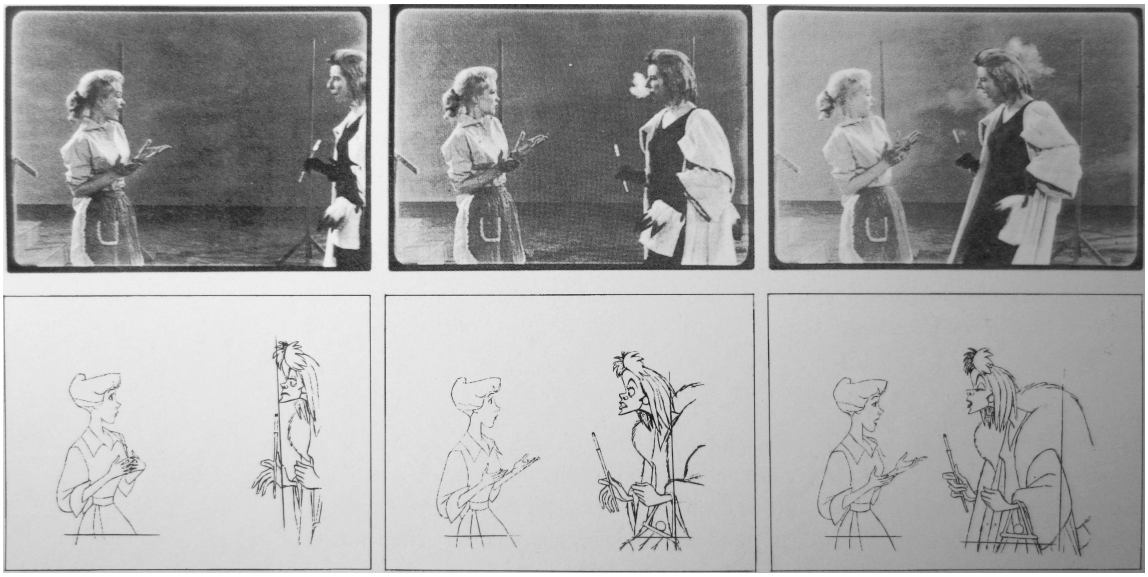

Figure 5. Example of Disney video reference. Disney animators brought their characters to life with the help of actors performing on video. Animators used this technique to include the subtleties that were usually overlooked. [17]

If the analysis resulted in a mathematical model it was applied to either the virtual character's 3D mesh or its skeletal rig [2]. A 3D mesh is animated by allowing the motion data model to manipulate the vertices of an articulated mesh, therefore producing an animated character without the need of a skeletal rig [3, 22].

Brand and Hertzmann created unique bipedal gaits by successfully applying their statistical model, obtained through PCA, on a virtual character's mesh to produce an animation [5]. Unfortunately, virtual characters animated through this, and similar methods, leave no room for adjusting the character's expressiveness [3, 22]. An animator is unable to tweak and manipulate the animation to add further detail to the performance.

Physically based simulations provide another solution for synthesizing quadrupedal locomotion. Coros et al. demonstrated a "real-time physics-based

quadruped simulation of gaits, gait transitions, sitting and standing up, targeted jumps, and jumps on-to and off-of platforms [9].” In their research, the creature’s skeletal hierarchy was created with appropriate physical constraints and properties, i.e., gravity and virtual forces and torques between connected joints. The forces, visualized in Figure 6, and torques were determined after analyzing the quadruped’s biomechanical constraints. An abstracted control system utilized proportional-derivative (PD) controls and inverse kinematics to regulate the locomotion and transitions desired by the user. Typically, PD control systems are used in robotics to ensure a fast response for features like tactile sensing. The inverse kinematics formulas evaluate the skeleton’s position for an animated character. The resulting simulated gait pattern was compared to motion capture data from a dog reference. Overall, the simulation was close to the original motion, however, there were “significant differences between the simulated motion and the captured motion [8].” The captured motion is from an actual quadruped, which provides an excellent example of natural motion. The simulated motion fails to include every muscle, tendon, and bone, which are included in the captured motion from an actual quadruped. Therefore, the simulated motion produces a less than natural animation.

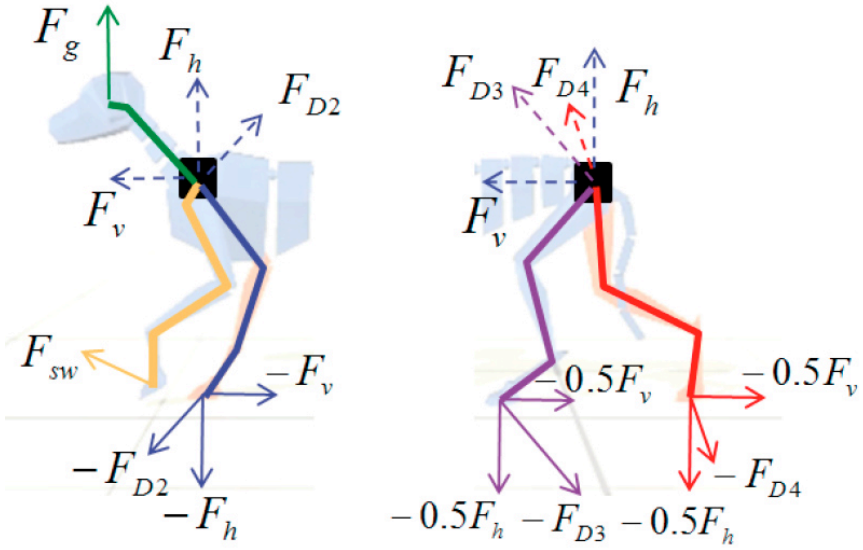

Figure 6. Illustration of internal forces described by Coros et al. A physically based simulation incorporated these internal forces for a virtual quadruped. These forces were used along with PD control algorithms and inverse kinematics to create a real-time simulation for a virtual dog. [8]

Although successful strides have been made in the research relating to quadrupedal gait analysis and synthesis, there remains a lot of potential for modeling and generating quadrupedal gaits on virtual characters. The following chapters will discuss my methodology for extracting and modeling a horse's gait from video as well as applying the model to a virtual quadruped's rig.

CHAPTER III

SAMPLING GAIT DATA

This chapter describes the process of extracting data that represents a natural gait pattern of a real horse. Video reference of a walking horse was imported into Maya 2012 [10] in the form of an image sequence. Joint objects (Maya's joint tool) were animated over the horse's joints throughout the image sequence, producing a rotoscoped animation. Figure 7 presents the scene setup for rotoscoping within Maya. Each joint's motion parameters were sampled using a Python script, which provided samples of the raw motion data. The sampled data was then conditioned for Fourier analysis.

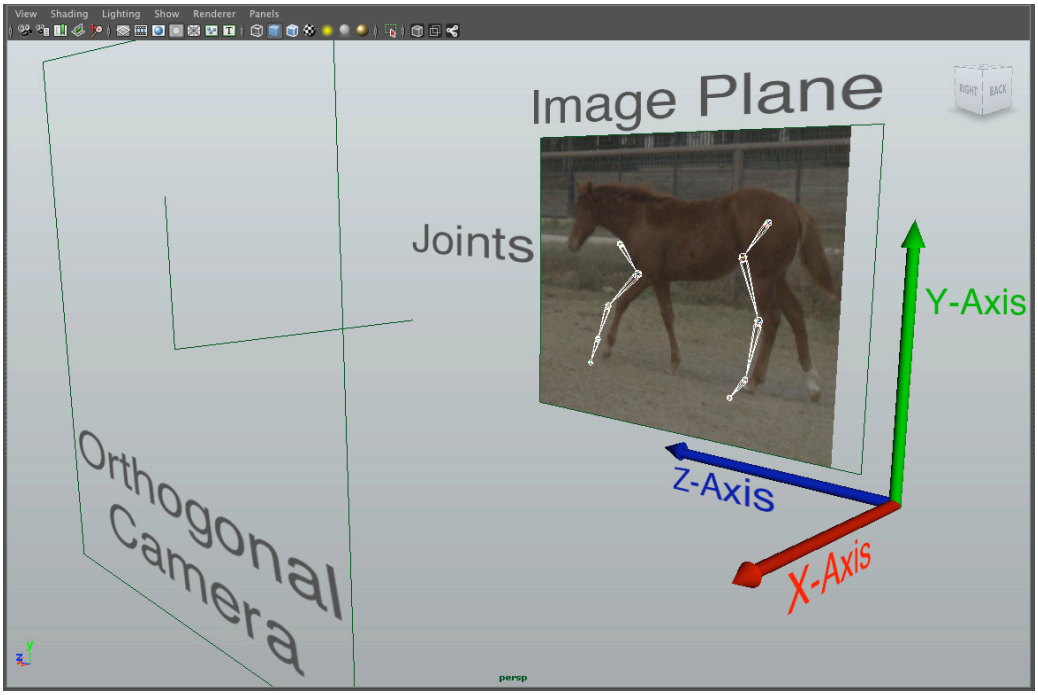

Figure 7. Scene setup in Maya 2012. Used for creating rotoscoped animation.

Video Reference

Horse video captured by the Perception Based Animation research group [27] at Texas A&M University was used for this research. A single camera recorded the side, or sagittal, view of the horse as it performed its gait. When limited to a single camera, an orthogonal perspective provides the most information about the horse's leg motion [4]. The primary rotation for each joint in the horse's front and rear legs are clearly visible from this perspective. However, the horse must maintain an orthogonal relationship to the camera. There must always be a clear picture of the true side of the horse.

Each joint's rotation values, in a given leg, along with the leg's root joint positions, contribute to the horse's natural gait pattern. Therefore, this study focuses on the data that describes the rotation of each joint in the horse's leg, along with the data that describes the position of the leg's root joint (i.e. shoulder and hip for front and rear legs) relative to the horse's center of mass.

Due to the symmetrical nature of the horse's walk gait, the front and rear leg facing the camera provide enough information for all four legs. A symmetrical gait is one where the only difference between the left and right side, for each pair of legs, is a phase difference of approximately 50% [23]. Therefore, motion data describing the front and rear leg describes the motion for all four.

Rotoscoping

The video has been converted into a sequence of still images and imported into the image plane of an orthogonal camera in Maya 2012 [10]. Using Maya's joint tool, forward kinematic joint chains were created and positioned over the horse's joints for the front and rear legs, as seen in Figure 8. In Maya, joints make up the skeletal hierarchy (skeleton) for a virtual character. A definition of the skeleton from Maya's online documentation [10]:

> Skeletons are hierarchical, articulated structures that let you pose and animate bound models. A skeleton provides a deformable model with the same underlying structure as the human skeleton gives the human body.

Figure 8. Joints animated over video frames. Front and rear leg joints were manually animated over the horse's joints to create a rotoscoped animation.

For this part of the project, the joints can be thought of as tracking objects used to create a rotoscoped animation. Throughout the image sequence, the joints were manually animated to maintain the position over the horse's joints. Figure 8 illustrates the process of matching the joints with the horse's joints. The shoulder, elbow, knee, and ankle joints were tracked for the front leg. For the rear leg, the hip, stifle, hock, and ankle were tracked. Together these animated joints represent the animal's natural motion and gait pattern, which is now separate from the original video.

Sampling

Sampling consists of extracting raw data from the rotoscoped animation that represents the horse's natural motion. At each frame of the rotoscoped animation, a joint's desired motion value was calculated and written to a text file. The joint motions include rotation and translation; rotation corresponds to the joint's angle and translation refers to the root joint's movement relative to the animal's center of mass. Sampling the appropriate values for each joint's motion was necessary in order to construct a model of a believable gait pattern. Details regarding the sampling calculations are provided in the following paragraphs.

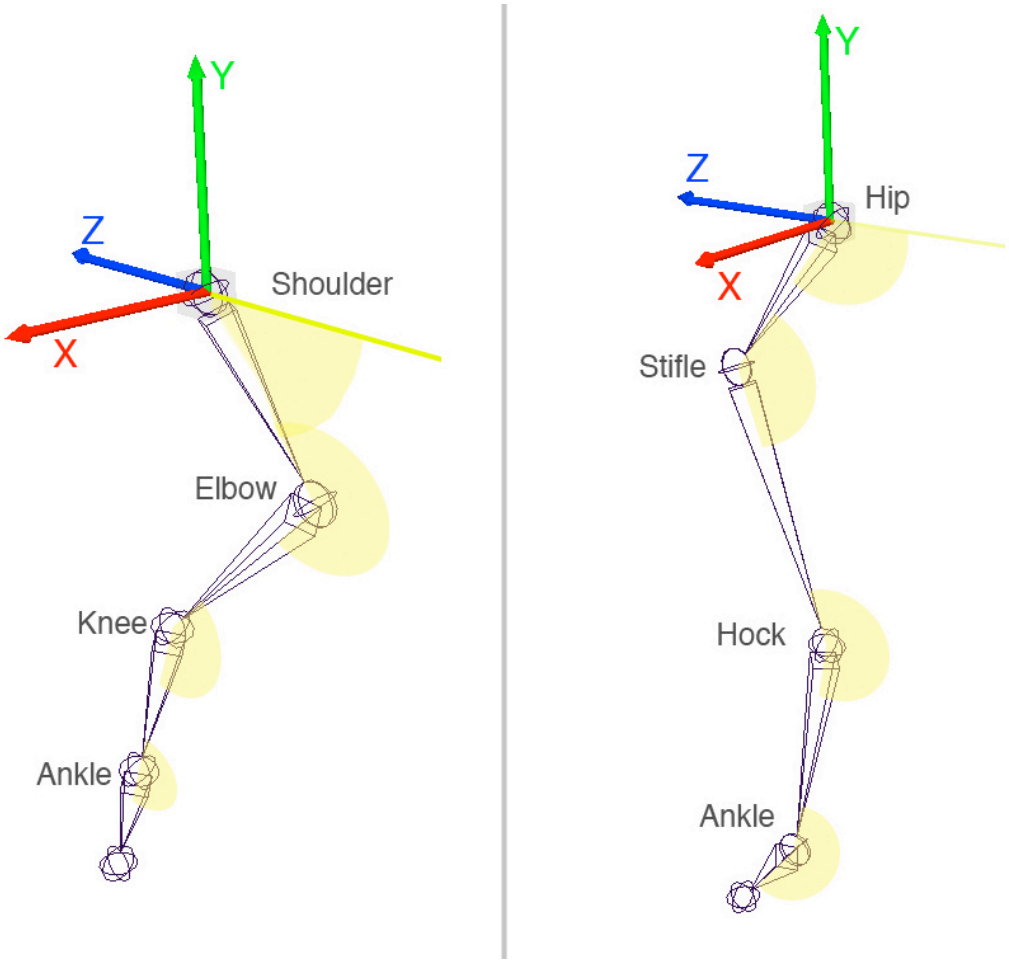

Figure 9. Desired front and rear leg joint angles. Each joint's rotational data was sampled by calculating the highlighted angle value for each frame. The shoulder and hip translation in the Y-axis and Z-axis with respect to the horse's body was also calculated.

Due to the periodicity of the horse's gait pattern, each joint's motion also performs periodically and can be thought of as a periodic signal (e.g. joint angle values changing over time to describe a joint's rotation). Extracting values representative of the joint motion at each frame is synonymous with sampling a continuous signal in an

engineering environment. The initial sampling rate is limited by the animation's frame rate of 24 frames per second (fps).

Calculating Joint Rotation Values

Sampling a joint's rotation value consists of calculating the joint's angle at a given frame. This is the angle between the two limb segments centered at the joint as seen in Figure 9. In order to accurately calculate the joint angle value, an orientation convention must be established. This also ensures consistency across different horse examples.

Maya's default world space system was used as a global control space. Therefore, when the image sequence was imported into the image plane for the "side" camera, screen "up" represents positive y-axis and screen "left" represents positive z-axis (Figure 7). Maya's environment follows the right hand rule, so that positive x-axis points out of the screen.

In order to calculate a joint's angle value, its position, in Maya's 3D world space, was queried along with its parent joint's position and its child joint's position. Two vectors were created, one pointing to the parent joint and the other pointing to the child joint, as seen in Figure 10 for the elbow joint.

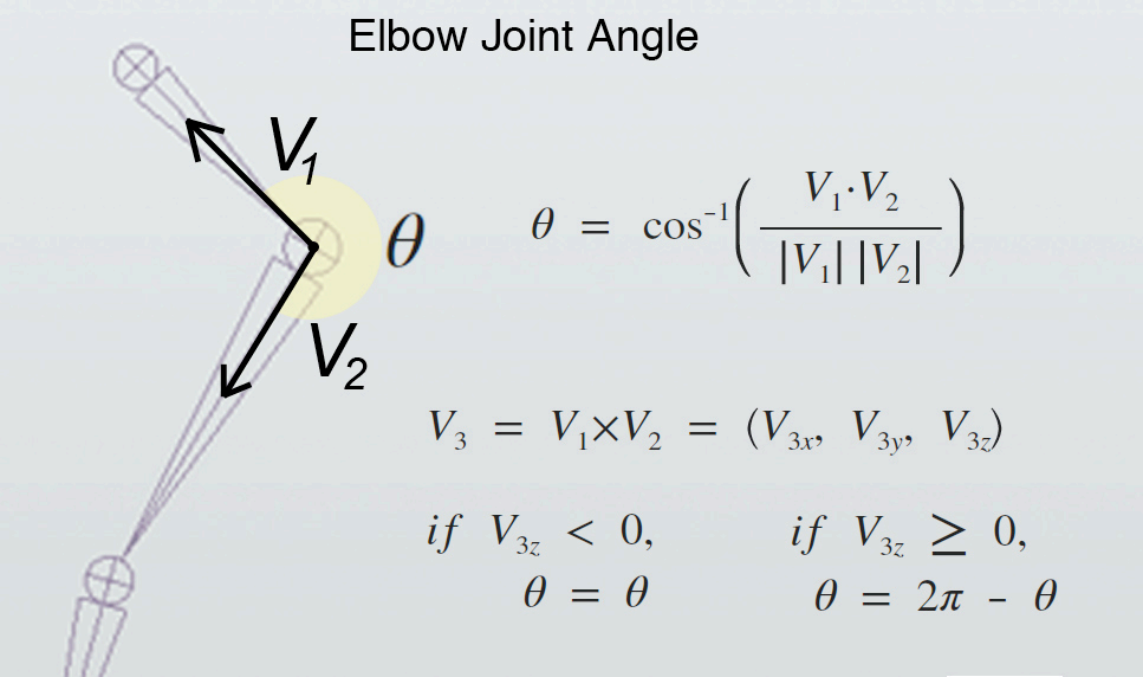

Figure 10. Elbow joint angle calculation. The dot product is used to calculate the angle between the two vectors while the cross product is used to ensure the correct side of the joint is calculated.

The dot product was used to calculate the angle between the two vectors. However, the dot product only returns angle values that are less than 180 degrees. This presented a problem for some of the joints. A solution came through the use of the cross product. By taking the cross product of the two vectors, it became clear which angle was calculated with the dot product. The cross product ensured that the appropriate angle was calculated, as shown in Figure 10. The resulting value was saved to a text file.

Sampling the shoulder and hip joint angle values was a special case. Since these joints are the root joints for their respective hierarchies, there was not a parent joint that could be used for the vector. Therefore, a vector in the negative Z-axis direction was

used as a reference vector along with a vector pointing to the joint below. The method is illustrated in Figure 11. The angle between these vectors was calculated using the same method previously described and saved to a text file.

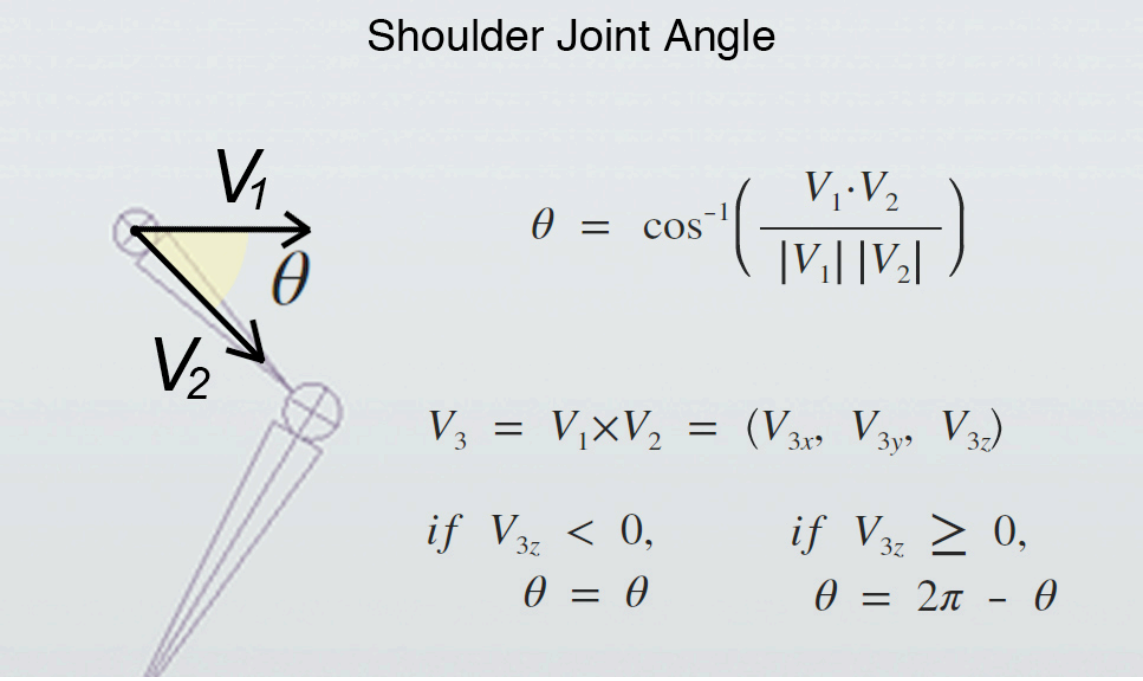

Figure 11. Shoulder joint angle calculation. The V1 vector is created along the root joint's local negative Z-axis while the V2 vector points toward its child joint.

The root joints for the front and rear legs move with respect to the horse's body. Calculating these values required an object that acted as the horse's center of mass. A locator was created whose position was determined by averaging the positions of all the joints. Units were arbitrary because the focus was on the joint's motion in the Y-axis and the Z-axis with respect to the center of mass locator. These values were calculated by

subtracting the root joint's Y component from the locator's Y component at each frame of the animation. The Z component was calculated similarly.

For an efficient sampling process, a script was created in Python where functions were defined that sampled each joint's desired motion. These samples represent the raw data extracted from the video of the horse. Each sampled signal was then conditioned in preparation for the FFT.

Conditioning

The purpose of conditioning was to create a set of samples that characterize the sampled signal and meet the requirements for the FFT. For the FFT to provide reliable information about a signal, the input signal must be periodic, unbiased, and characteristic of the signal that it is representing. The goal for conditioning the signal was to select and repeat values from the sampled set of raw data for a given joint's motion. Once conditioned, the FFT could provide further analysis of the signal. Microsoft Excel was used because it allows quick and easy data visualization.

Conditioning consists of the following steps:

1. Select values from the raw sampled data that characterizes the signal.
2. Create a periodic extension by repeating the selected values five times.
3. Select a starting and ending value from the new set of samples.
4. Remove the bias and account for the time shift.

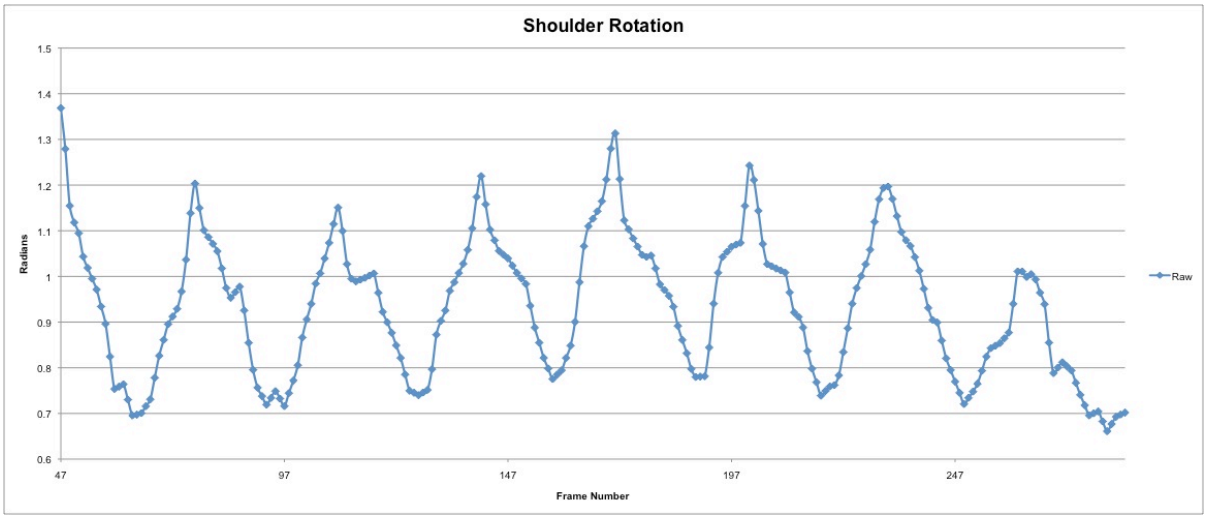

Figure 12. Sampled signal for shoulder rotation.

The sampled motion values defining the raw motion of a particular joint were imported into an Excel spreadsheet and graphed for initial viewing as shown in Figure 12. The period of each signal for a given horse was determined by the horse's stride length. The period length was consistent for each joint's motion signal. In this case, the stride length was determined manually by visually scanning each joint's signal and comparing its period with the stride length noted while rotoscoping the video. Counting the frames between peaks or troughs in a signal also works.

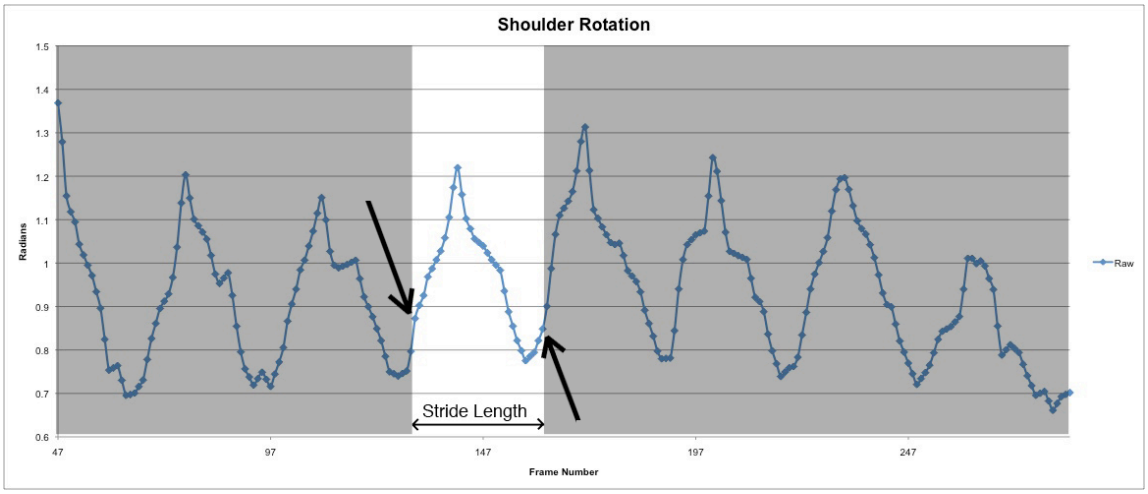

Figure 13. Selected sampled values for repeating. These create a signal characteristic of the original. The length of this set is also equal to the stride length of the horse's gait.

Because it was crucial that each adjusted signal maintained a consistent period equal to the stride length, a certain artistic license was taken in manually selecting the sample values that characterize the signal. Figure 13 provides an example of selecting values used to create a characteristic signal of the original. The goal here was to select values that correspond to a natural part of the horse's gait pattern, which was determined by comparing the section of the signal with the section of the video. The selected values were repeated to verify that the period was correct and that the repeated values created a good approximation to the originally sampled signal. Some values were altered slightly to account for errors from the sampling method. If the data was obtained through a high fidelity system like motion capture, the sampled data would have been more accurate and these modifications would not have been supported.

The set of selected sample values were copied and repeated about five times. Five was a good rule of thumb to ensure that the selected values form a signal that matches the original. However, the sample value at the beginning and end of the adjusted set must be the same. The selected samples that compose the adjusted set of values must start and stop on the same value. An explanation is provided in the following chapter.

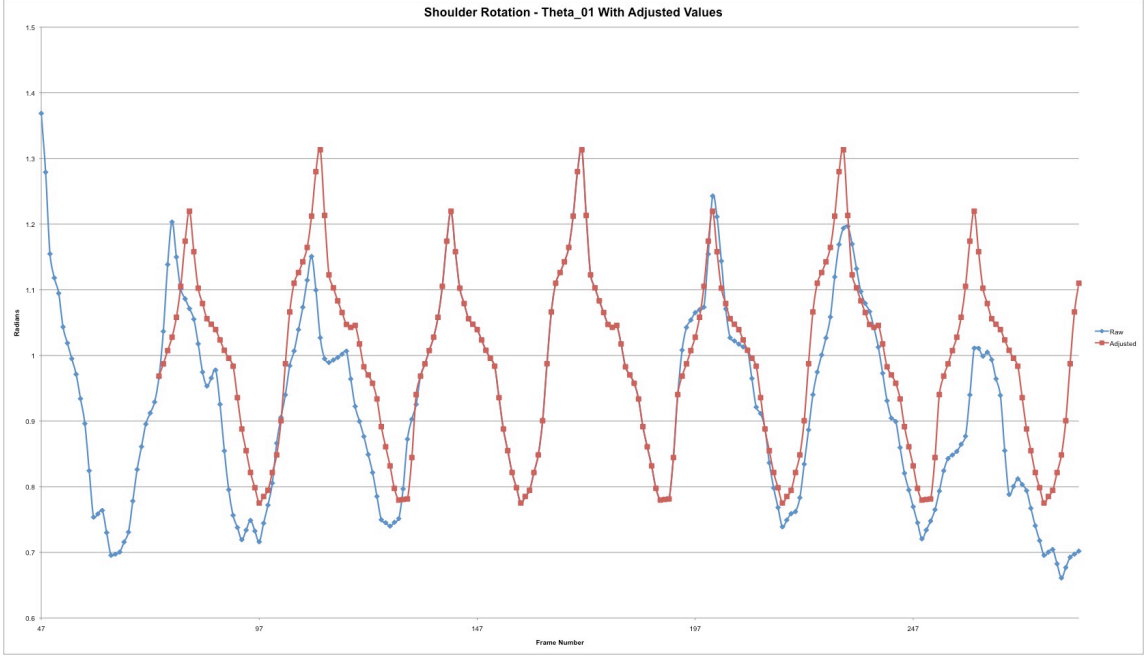

Figure 14. Characteristic signal with raw signal.

Out of the selected and repeated set of sampled values, the starting value was selected at a point in the signal where there was relatively no change in the signal's

direction. In other words, a starting value was not chose at a peak or trough in the signal as shown in Figure 13. Once the starting value has been determined, the number of frames between the starting value and the first raw sampled value must be counted and noted. This was the time shift needed for each signal to allow for synchronization during the gait synthesis.

At this point, a new version of the raw signal has been created and is shown in Figure 14 along with the raw signal. This signal is characteristic of the original, but has a bias (offset). The bias was equal to the signal's average, which is half the sum of the signal's maximum and minimum values. The offset was subtracted from each sampled value, leaving the average at zero. These values were saved to a comma separated value (csv) formatted file for ease of reading into the FFT Python script. This file represents a conditioned signal describing the motion of a given joint, in the form of discrete samples. These files were ready for FFT analysis, which is described in the following chapter.

CHAPTER IV

FOURIER ANALYSIS

This chapter details the methodology required to derive a mathematical model of a believable gait pattern through Fourier analysis. The conditioned sampled signals, for each joint's motion, were upsampled and analyzed using the fast Fourier transform (FFT). A Fourier series used the FFT results to create mathematical functions for each joint's motion. When combined, these functions create a model for a believable gait pattern. The results from this chapter serve as the mechanics for creating a synthetic gait pattern. An overview of the analysis methodology is provided in Figure 15.

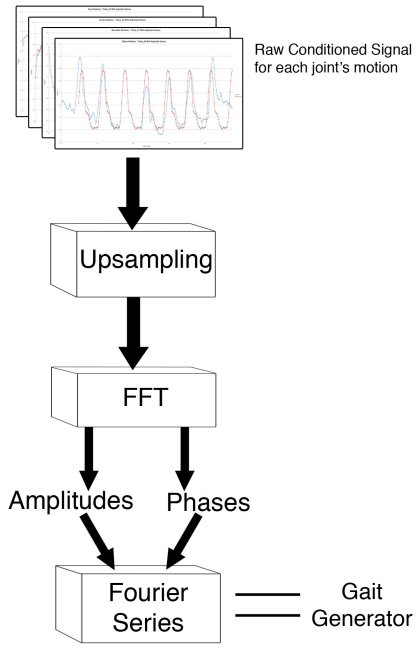

Figure 15. Overview of analysis process.

Upsampling

For the FFT to produce reliable results, its input signal must be in the form of discrete samples acquired at a high sampling rate. In this case, the initial sampling rate was limited by the frame rate of the rotoscope animation, 24 frames per second. This rate is too low for the FFT. However, retroactively upsampling alleviates the problem by creating more sample values between those already defined.

Upsampling was achieved through cubic interpolation of the original sample points. Cubic interpolation is a technique that approximates a continuous signal between the original sample points. A larger number of samples can now be acquired from the interpolated signal by gathering values at smaller discrete intervals across the signal. Results from upsampling are provided in Figure 16.

For efficiency when computing the FFT, the number of samples from the input signal should be a power of two. This is known as a radix-2 FFT [24] and is a common approach to computing the FFT. A radix-2 FFT problem computes the discrete Fourier transform through a divide and conquer method. To satisfy the radix-2, each joint's signal was ultimately upsampled to 32768 values, which is equal to 2 to the power of 15.

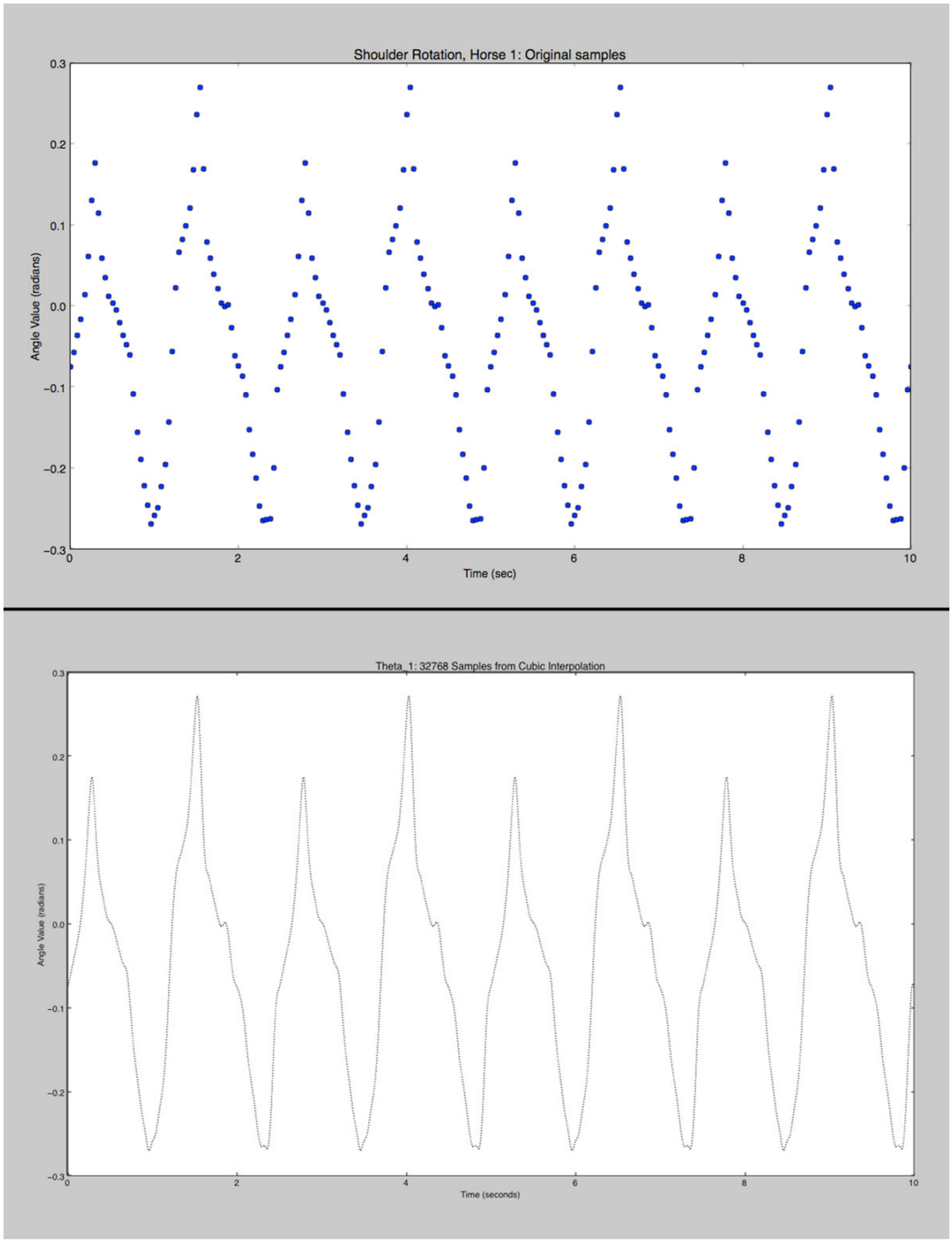

Figure 16. Original and upsampled signals. Original conditioned signal, sampled at 24 samples per second (top) and upsampled version with 32768 samples (bottom).

After each joint's motion signal has been upsampled to 32768 values, the FFT was performed, which provided insight to the signal's frequency components.

Analysis with the FFT

In order to analyze each joint's sampled signal and determine its mathematical function, a workflow environment must be established. The environment requires powerful and efficient tools to perform the analysis. Python and its packages, NumPy and SciPy, provide an environment for scientific computing. SciPy comes equipped with fftpack, which contains the FFT function.

The signal to be analyzed performs in the time domain and therefore is difficult to define mathematically if limited to the time domain. The FFT provides a more manageable analysis of the signal by transforming the sampled signal from the time domain into the frequency domain. Generally, a Fourier transform takes an expression from the time domain into the frequency domain. This allows for an easier and convenient investigation into the composition of the signal. The FFT is an efficient form of the discrete Fourier transform (DFT). Its application is discussed rather than an in-depth explanation, which would require a separate paper.

Performing the FFT on a set of sampled values from a periodic signal provides information on the frequencies present in the signal. The results render a clearer description of the signal's components from the perspective of the frequency domain.

The results from the FFT were used to calculate the amplitude and phase value for each frequency present in the signal. Amplitude values were calculated in decibel (dB) and phase values in radians as shown in Figure 17 and Figure 18. Frequencies with larger amplitudes provide more influence to the sampled signal. These values are generally viewed in a logarithmic graph (Figure 17) while the phase values are linear (Figure 18). A frequency's phase and amplitude values are known as frequency components. These components were pulled into a form of the Fourier series to construct closed form mathematical expressions for each signal.

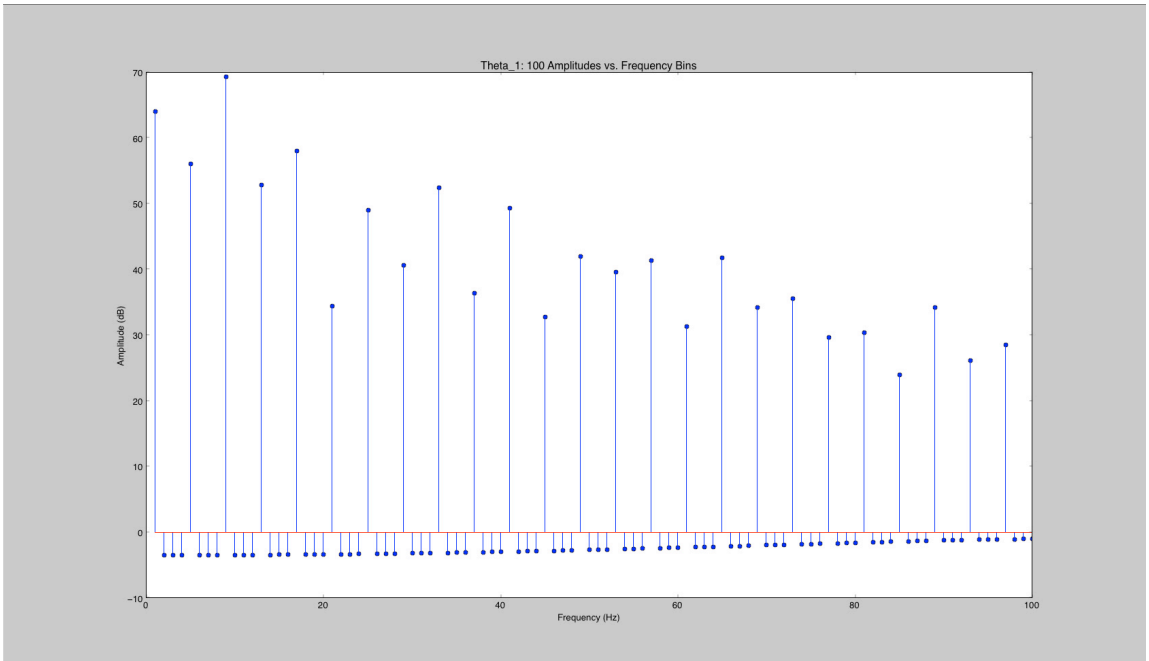

Figure 17. Amplitude values for 100 frequencies. Each stem's value represents a frequency's amplitude (in dB).

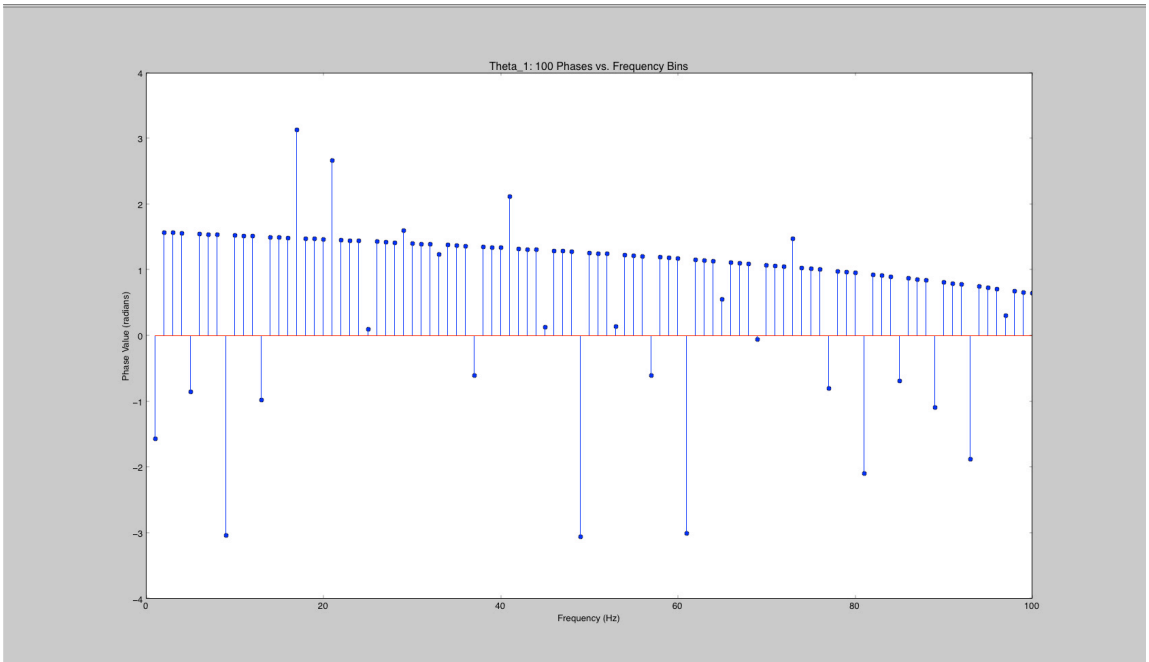

Figure 18. Phase values for 100 frequencies. Like Figure 17, each stem represents a frequency's phase value (in radians).

Fourier Series

A Fourier series decomposes an unknown periodic signal into basic sine and cosine functions by using the frequency components calculated with results from the FFT [25]. Theoretically, the series is infinite and when enough components are included in the series, creates an expression almost identical to the unknown signal.

Using the first 250 frequency components derived from the FFT, a truncated form of the Fourier series is created for each joint's motion signal. The mathematical equation is provided in Equation 1 and the Python code in Figure 19. As more frequency components were included, the approximation became more precise. 250 components provided a very close result for every signal. This process was used to create functions

describing each joint's angle values and the shoulder and hip's translational value. Figure 20 shows the result of the cubic interpolation of the original signal (top) along with the result from the truncated Fourier series with 250 frequency components (bottom).

$$\sum_{n=1}^{251} Amplitude \times \cos\left(\omega_{\Delta} \times (n) \times t + Phase\right)$$

Equation 1. Truncated form of the Fourier series that uses 250 frequency components calculated through the FFT results.

```
Y = fft(y), where y is the upsampled signal
for n in range(1, 251):
    Amplitude = abs(Y[n])
    Phase = arctan2(Y[n].imag, Y[n].real)
    y_est = y_est + Amplitude*cos(ω_Δ*(n)*t + Phase)
```

Figure 19. Implementation of Fourier series in Python.

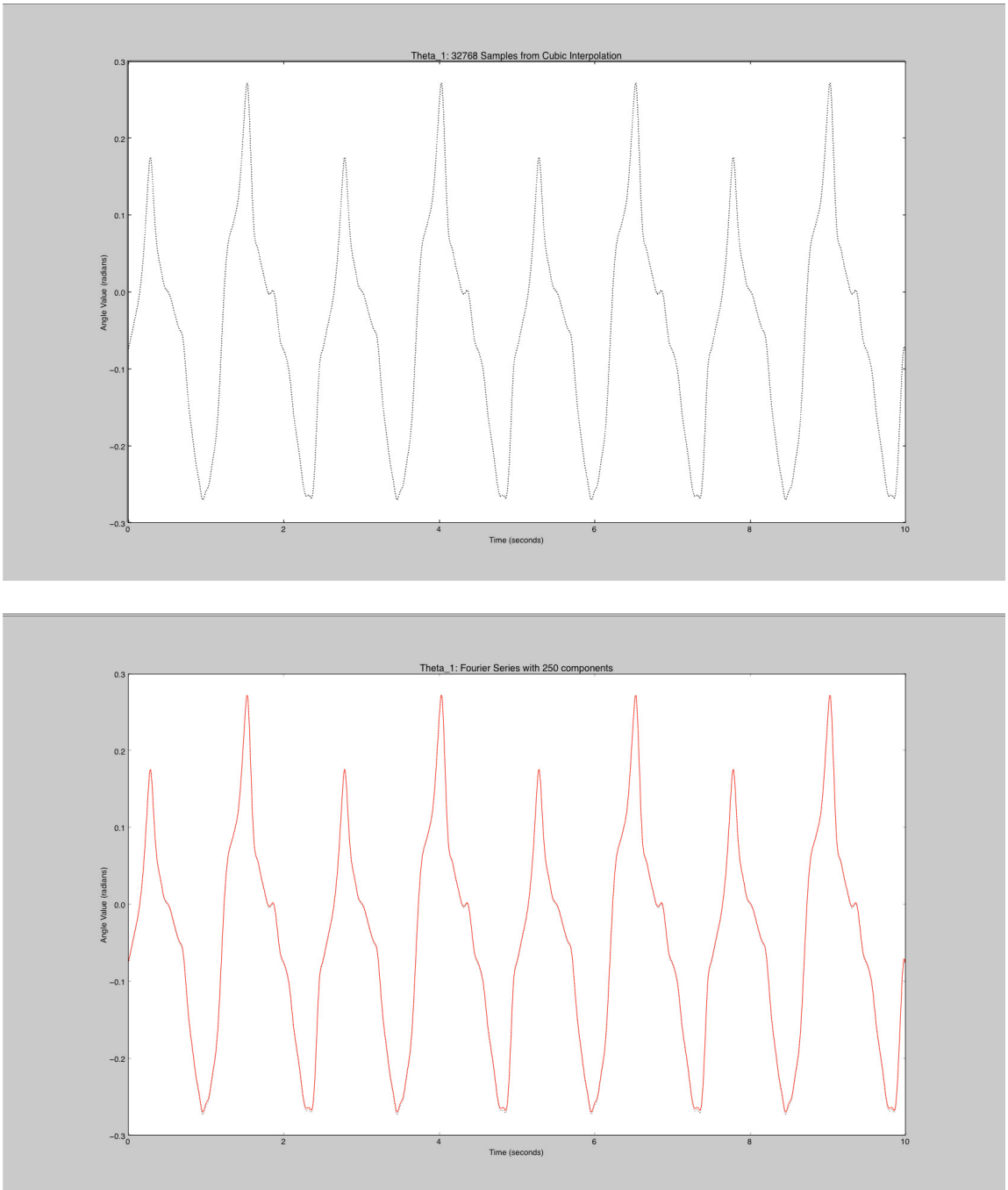

Figure 20. Comparison of cubic interpolation and Fourier series. Top graph represents the cubic interpolation of the original signal for the horse's shoulder rotation (upsampling). Bottom displays the truncated Fourier series with 250 frequency components.

When the resulting signal matched the original, the amplitude and phase values were saved to a text file for use in creating a synthetic gait pattern animation. The entire analysis process of upsampling, performing the FFT, calculating the Fourier series, exporting the values, and plotting the results was scripted in Python.

CHAPTER V

APPLICATION

In this chapter, the mathematical model of a believable gait pattern derived during the analysis chapter was used to create a synthetic animation with a new rig for a virtual character. The rig was based on the joint layout used to rotoscope the animation. An accompanying control hierarchy was established with constraints set to control the desired motion of the joints. The result was a set of four animated legs that mimicked a believable gait pattern for a quadruped.

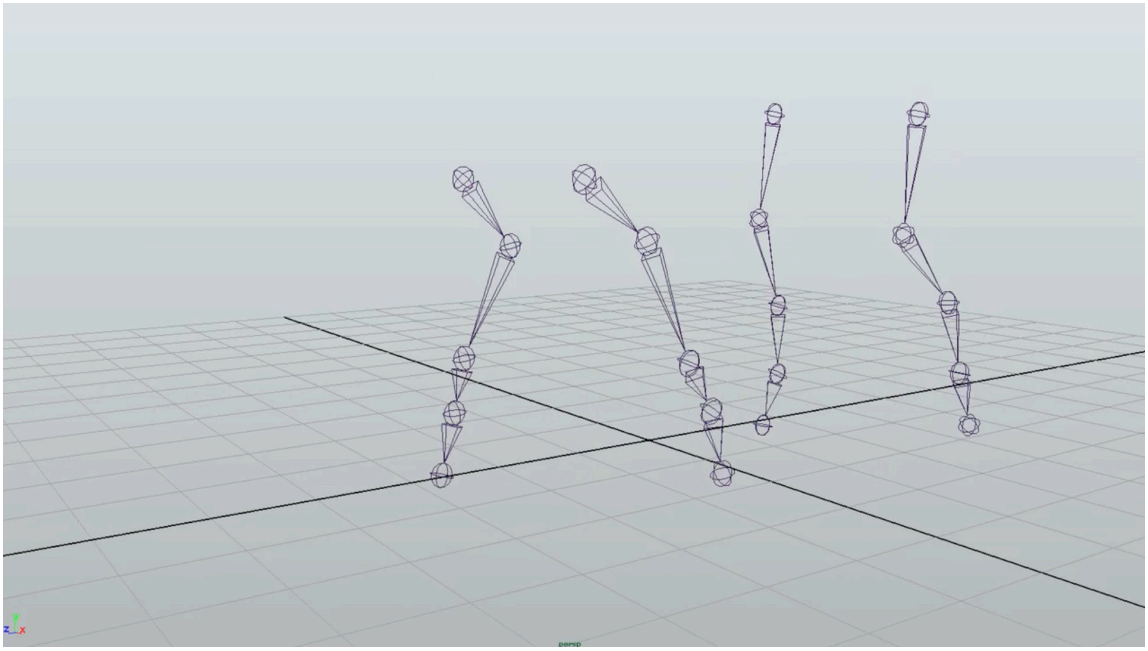

Figure 21. Rig used to create synthetic animation. The gait generator model will be used to drive the animation of the joints.

Virtual Character Rig Setup

Typically, a virtual character mesh (i.e. a virtual 3D model) must be rigged before animation can begin. Rigging consists of building the skeleton, adding inverse kinematics, if needed, and establishing manipulation handles to give the animator control over the character mesh [10]. The primary goal of this research however, is to define a mathematical model for a believable gait pattern and apply it to a new set of joints. Therefore, a quadruped mesh was not necessary. For this, the rig consisted of four legs, each comprised of a joint and control hierarchy connected by appropriate constraints, as shown in Figure 21, Figure 22, and Figure 23. Ideally this rig could later be attached to a character mesh. No manipulation handles were created either.

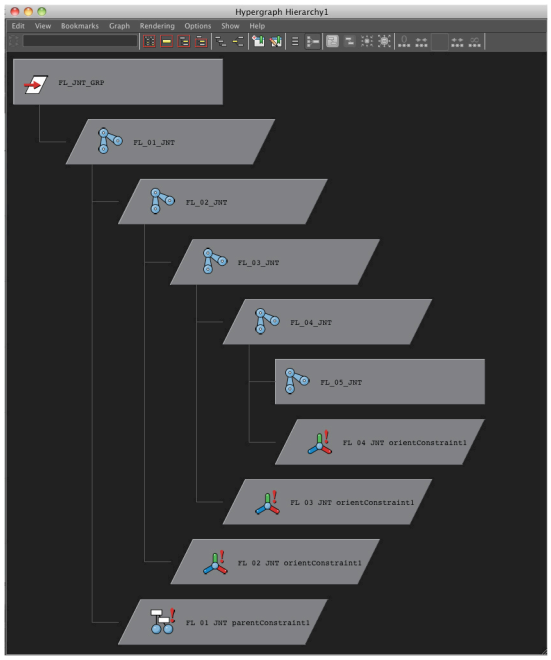

Figure 22. Joint hierarchy. This includes the constraints for the front left leg.

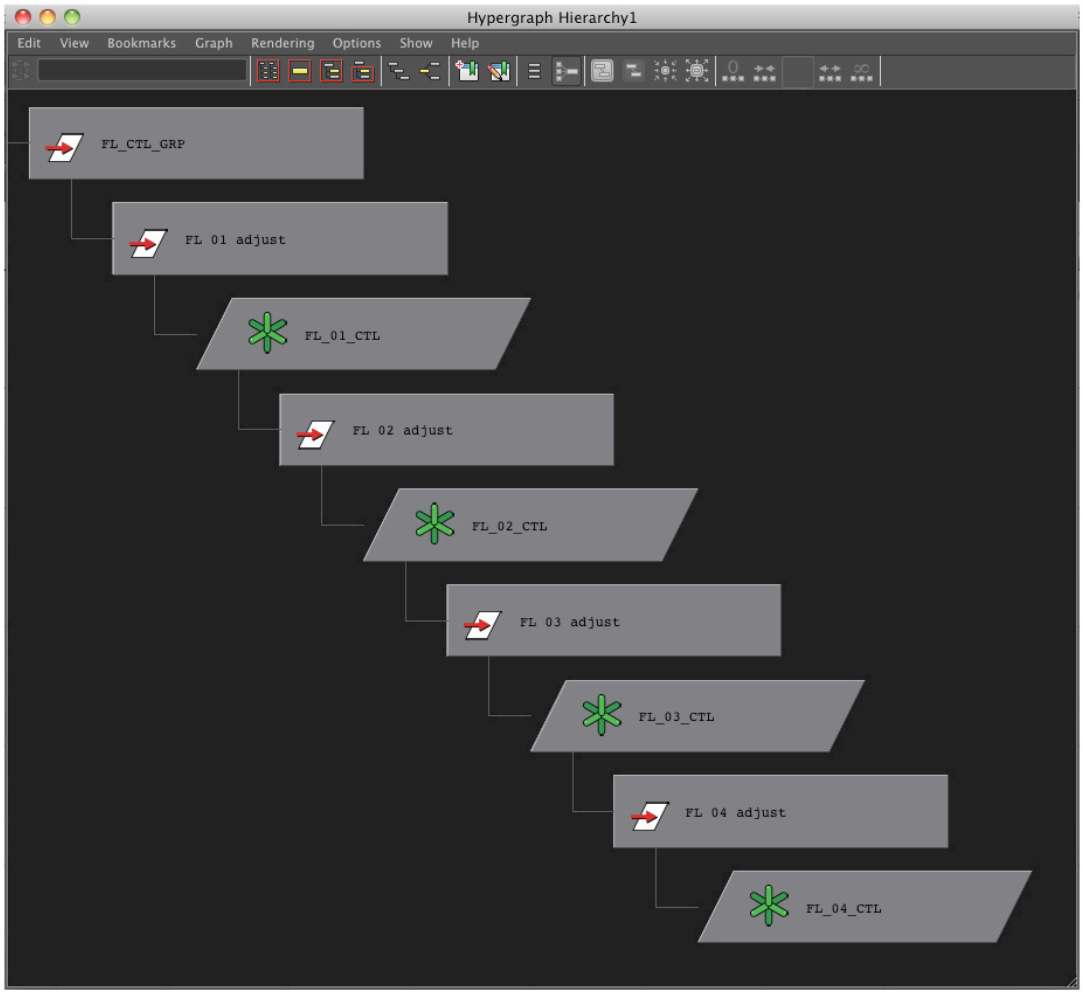

Figure 23. Front left leg control hierarchy.

For the synthetic animation to work correctly, the rig's joint distances within a leg must maintain the same proportions as the leg joint structure used during rotoscoping. If the lengths between the joints were different than the original lengths, the resulting synthetic gait pattern would appear unnatural and out of synchronization. To fulfill this requirement, the joint chains used during the rotoscope process were duplicated and renamed for use as the rig to be animated.

In most cases, a rig's control layer establishes a level of abstraction between the animator and the joint hierarchy to alleviate complexity. This level of abstraction is necessary when the rig includes manipulation handles and deformation controls. However, the focus of this rigging process was on applying mathematical functions to the rig to produce an animation. There was no need to create an interface for the animator. For the sake of clarity when applying the expressions to animate the rig, a control hierarchy was established. The control system also provides room for manipulation handles to be created for future use.

The derived functions that model the motion for a specific joint were applied to control objects in the control hierarchy. Locators were used as the control objects. Locators are markers used to indicate points in Maya's world space and are used as an aid in user interfaces. Constraints between the joints and the locators allowed the joints to become animated. As defined in Maya's online documentation [10]:

> With constraints, you can drive the position, orientation, and scale of one object with the transformation settings of another object. The object that is driven is called the constrained object, and the driver object is called the target object.
>
> Orient constraints: Orient constraints limit and control only the rotation channels of the constrained object.
>
> Parent constraints: Parent constraints cause the constrained object to inherit the transformations and global orientation of its target objects, mimicking a parent-child relationship.

In addition to the control grouping, null nodes were created as parents to each locator object. Null nodes are empty dependency nodes, which in this case, were used

only for additional adjustment of the entire signal. Therefore, if a joint's resulting rotation animation was correct, but had an offset, the adjustment node could be used to account for it through a rotation of only the adjustment node.

To simplify the process of creating the opposing leg, the root joint of the leg's joint hierarchy was grouped under a node. Similarly, the parent node of the control hierarchy was also grouped under a node. Both of these nodes were then grouped together to represent the leg's joint and control structure. Duplicating and renaming this group provided the necessary system for the opposite leg. The same approach was used to create the rear legs.

Application

This section describes the application of the mathematical models. Each joint's motion is expressed through a mathematical model, which was applied to the control object governing the joint's motion. An animation of the natural gait pattern was created when all of the expressions were applied to the appropriate control objects.

The mathematical model describes the signal of each joints' motion. However, in order to successfully apply the expression to the control object, adjustments must be made. The expression that defines a joint's rotation produced values in radians. Rotation in Maya is in units of degrees, which meant that the result from the expression was converted by multiplying the radian value by 180 and diving by pi (π).

The bias, which was removed during conditioning, and the time shift, which was noted during conditioning, were reintroduced to the resulting signal. The bias readjusts

the entire signal so that the motion looks correct in the final animation. The time shift ensured that all of the joints move in synchronization.

Frames for the synthetic animation were designated and at each frame, the truncated Fourier series calculated the appropriate value, which was adjusted to appropriately control the joint's attribute. The Fourier series used the same frequency components that were exported in the analysis phase of the project. The attribute could be rotation or translation, depending on which function was processed. Once set, a key frame was placed on the control type.

Since the horse's walk gait is symmetrical, the left leg's motion is equivalent to that of the right leg, but includes a phase difference of 50% [24]. For each joint in the right leg, the phase shift was accounted for during the Fourier series calculation as a time shift in the cosine function, which is illustrated in Equation 2. The time shift is half of the stride length. The stride length was noted during the sampling process.

$$\text{Left Leg:} \quad \sum_{n=1}^{251} Amplitude \times \cos(\omega_{\Delta} \times (n) \times t + Phase)$$

$$\text{Right Leg:} \quad \sum_{n=1}^{251} Amplitude \times \cos(\omega_{\Delta} \times (n) \times (t - t\_shift) + Phase)$$

$$t\_shift = \frac{1}{2} \times (stride\ length)$$

Equation 2. The same truncated Fourier series is calculated for the opposite side, but includes a time shift equal to half of the stride length.

For example, Horse 1 has a stride length of 30 frames. Therefore the time shift needed to separate the front left leg's motion from the front right leg's motion is 15 frames or 0.625 seconds. This time shift is included in the Fourier series calculation that determines the value for the joint's control object.

The application of the gait's model resulted in a synthetic, yet natural moving gait pattern for the virtual quadrupedal rig. Each joint's rotation was mathematically defined and applied to the appropriate control object governing a specific joint in the rig, while the root joint's translational motion was similarly modeled and applied.

Results

Through the application process described in this chapter, a synthetic animation was created. The animation used the model for a believable gait pattern found in the analysis chapter to create the new rig based on the joint layout used to rotoscope the animation during the motion data-sampling chapter. An accompanying control hierarchy was established with constraints set to control the desired motion of the joints. The result was a set of four animated legs that mimicked a believable gait pattern for a quadruped.

Three different walking horses were analyzed in order to form a mathematical model for the specific walk. These models were successfully applied to a rig for a new virtual quadruped, producing a synthetic gait pattern. The results are included as videos with this thesis.

CHAPTER VI

CONCLUSION AND FUTURE WORK

This thesis provides a prototype system for creating a believable quadrupedal gait pattern generator for virtual quadrupedal characters. Gait analysis of a horse's walk, from a rotoscoped animation based on video reference, provided a means of constructing a mathematical model of the horse's natural gait pattern. This model was applied to a virtual character's rig to produce synthetic locomotion. Ideally, this process alleviates the animator's workload by identifying the quadruped's gait pattern from video and creating a generator that produces the gait pattern, thus allowing the animator to focus more attention on achieving the desired performance of the character.

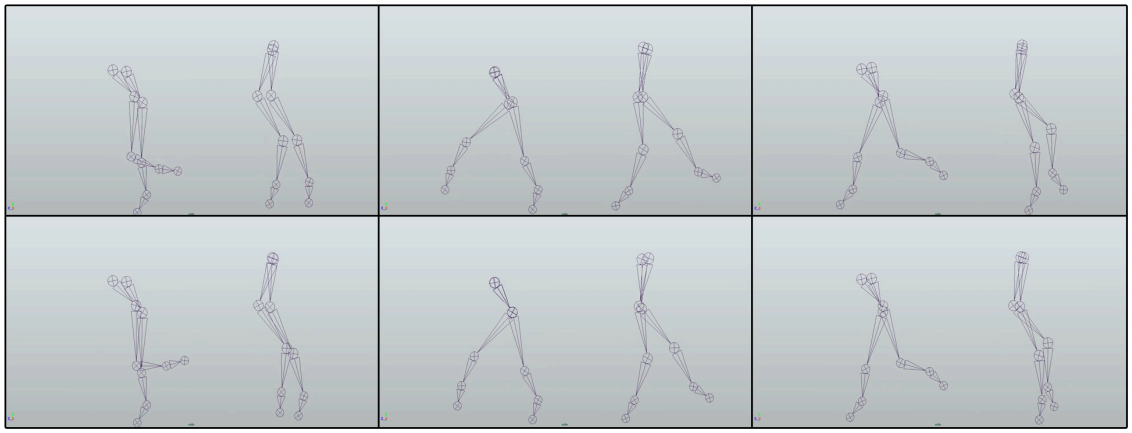

Figure 24. Frames from the final synthesized gait animation.

Gait analysis was conducted by analyzing a real horse's walk from video. Raw motion data was separated for sampling through the rotoscoping process. Sampling obtained motion data values for each joint in the front and rear leg facing the camera. Each joint's rotational motion and the front and rear leg root joint translational motion were the primary focus in this research. Due to the periodicity of the gait pattern, each joint's motion data was conditioned and analyzed through Fourier analysis. The analysis included performing the FFT on each joint's sampled values and using the results to form a closed-form expression through a truncated version of the Fourier series. The combination of each joint's expression formed a complete model representative of the natural gait pattern mimicking that of the horse from the video reference.

A new rig for a virtual quadrupedal character was created, which included a joint hierarchy for four legs along with a control hierarchy with appropriate constraints. The front and rear leg joints on one side of the rig were animated through the application of the derived expressions on the control objects. The leg joints on the opposite side were animated using the same expressions but included a time shift equal to half of the stride length. Frames from the final animation are shown in Figure 24. Overall, this process produced a quadrupedal gait pattern generator, which drove the locomotion of a virtual quadruped's leg joints.

The gait generator produced realistic animation for a virtual quadrupedal character, but in a limited environment. Limited meaning that full 3D motion was unable to be achieved due to the nature of the mathematical model of the quadrupedal gait pattern derived from the single camera capture method. The gait analysis method created

a gait generator able to drive each joint's rotation around one degree of freedom (i.e. the joint's local x-axis) rather than all three degrees of freedom (i.e. motion around the joint's local x, y, and z axes). Even though the resulting animation will not appear completely believable in a virtual 3D environment, the research provides a proof of concept for creating mathematical models of periodic motion from video.

Other limitations surfaced through researching the methodology. The process for sampling joint motion data uses video of a horse walking orthogonally to the camera. Unfortunately, it is extremely difficult to maintain this relationship with the animal as it naturally performs its gait. As mentioned previously, a single camera can only capture a limited amount of information about the horse's movement in 3D space. Therefore, when examining the rotation motion for each joint, user discretion is required to verify whether or not the side of the horse was clear throughout the selected video frames for analysis. Even though the method relies on imperfect video, the results were promising and shed light on many possibilities for future work.

Future Work

This thesis offers many opportunities for future research in the area of generating synthetic gait patterns for virtual quadrupedal characters. Further frequency analysis of each joint's motion signal would provide better insight when modeling a gait pattern. Comparison among signals through a given leg or even a given joint across multiple examples might produce a unifying factor throughout each signal. This factor could aid in creating a general expression for a horse's walk pattern based on size of the virtual

character's limbs and eliminate hours of repetitive work. Once analysis of different gait patterns for a given animal have been performed, interpolation techniques could be employed to transition between gait patterns as was demonstrated in [18].

Optimizing the current process would accelerate the creation of the gait generator for a given quadruped. This includes using a windowing technique to examine only a section of each joint's raw motion signal. Windowing is a common technique in signal processing applications where only a set range of a signal is examined while everything outside of the range is set to zero. This technique would aid in automating and simplifying the conditioning process of the sampled raw signal. Automating the signal conditioning section would save time and reduce explanation for the user.

Code optimization along with a conversion to C++ would benefit by increasing the program's efficiency and possibly allow for real-time calculation for a gait pattern. C++ can perform much faster than Python and can be used within Maya through Maya's C++ API as a Plug-in. A graphical user interface (GUI) would also benefit the current setup along with an implementation through C++. Through a well-designed GUI, the animator would be able to tweak and adjust each joint's motion signal in real time.

Fortunately, this research will continue through the Perception Based Animation research group in the Department of Visualization at Texas A&M University. One of the major goals for the group is to define and implement expressive characteristics for a quadruped's gait. Methods similar to those described in this thesis will be used to pursue the future research.

REFERENCES

[1] Wright, Dean, Bill Westenhofer, Jim Berney, and Scott Farrar. "The Visual Effects of the Chronicles of Narnia: The Lion, the Witch and the Wardrobe." *Computers in Entertainment.* 4, no. 2 (2006): 4.

[2] Skrba, Ljiljana, Lionel Reveret, Franck Hetroy, Marie-Paule Cani, and Carol O'Sullivan. "Animating Quadrupeds: Methods and Applications." *Computer Graphics - New York - Association for Computing Machinery – Forum.* 28 (2009).

[3] Favreau, Laurent, Lionel Reveret, Christine Depraz, and Marie-Paule Cani. "Animal Gaits from Video." In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 277-86. Grenoble, France: Eurographics Association, 2004.

[4] Wilhelms, Jane, and Allen Van Gelder. "Interactive Video-Based Motion Capture for Character Animation." In *Proceedings of IASTED Conference on Computer Graphics and Imaging*. Kauai, Hawaii, 2002.

[5] Brand, Matthew, and Aaron Hertzmann. "Style Machines." In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, 183-92. New Orleans, LA: ACM Press/Addison-Wesley Publishing Co., 2000.

[6] Cappelletto, Jose, Pablo Estevez, Wilfredis Medina, Leonardo Fermin, Juan M. Bogado, Juan C. Grieco, and Gerardo Fernandez-Lopez. "Gait Synthesis and Modulation for Quadruped Robot Locomotion Using a Simple Feed-Forward Network." In *Proceedings of the 8th International Conference on Artificial Intelligence and Soft Computing*, edited by Ryszard Tadeusiewicz Leszek Rutkowski, Lotfi A. Zadeh, 731-39. ICAISC '06: Springer-Verlag, 2006.

[7] Alexander, Robert McNeill. "The Gaits of Bipedal and Quadrupedal Animals." *The International Journal of Robotics Research*. 3, (June 1984): 49-59.

[8] Coros, Stelian, Andrej Karpathy, Ben Jones, Lionel Reveret, and Michiel van de Panne. "Locomotion Skills for Simulated Quadrupeds." In *Proceedings of ACM Transactions on Graphics* 30 (2011): 1-12. Vancouver, BC: ACM, 2011.

[9] Mixamo. "Mixamo, Animation in Seconds." Accessed August 9, 2012. http://www.mixamo.com/c/company.

[10] Autodesk, "Autodesk Maya Online Help," 2012. [Online]. Available: http://download.autodesk.com/global/docs/maya2012/en_us/index.html

[11] Parent, Rick. *Computer Animation Complete*. Burlington, MA: Morgan Kaufmann, 2009.

[12] Neversoft. *Tony Hawk's Project 8*. (Activision). Xbox 360, 2006.

[13] Electronic Arts. *Tiger Woods PGA Tour 13*. (Electronic Arts). Xbox 360, 2012.

[14] Bennett, David. "The Faces of "The Polar Express"." In *ACM SIGGRAPH 2005 Courses*, 6. Los Angeles, California: ACM, 2005.

[15] Menache, Alberto. *Understanding Motion Capture for Computer Animation, Second Edition.* Burlington, MA: Morgan Kaufmann, 2010.

[16] Qualisys. "Motion Capture Technology." Accessed August 9, 2012. http://www.qualisys.com/about/motion-capture-technology/.

[17] Thomas, Frank, and Ollie Johnston. *The Illusion of Life*. New York: Hyperion, 1981.

[18] Hoffmann, Jan, and Uwe Duffert. "Frequency Space Representation and Transitions of Quadruped Robot Gaits." In *Proceedings of the 27th Australasian Conference on Computer Science - Volume 26*, 275-78. Dunedin, New Zealand: Australian Computer Society, Inc., 2004.

[19] Unuma, Munetoshi, Ken Anjyo, and Ryozo Takeuchi. "Fourier Principles for Emotion-Based Human Figure Animation." In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, 91-96. New York, NY: ACM, 1995.

[20] Duffert, Uwe, and Jan Hoffmann. "Reliable and Precise Gait Modeling for a Quadruped Robot." In *Robocup 2005*, edited by Bredenfeld Ansgar, Jacoff Adam, Noda Itsuki and Takahashi Yasutake. 49-58: Springer-Verlag, 2006.

[21] Yu, Lianqing. "Gait Analysis and Implementation of a Simple Quadruped Robot." In *IEEE*, 431-34. Wuhan University, China: IEEE, 2010.

[22] James, Doug L., and Christopher D. Twigg. "Skinning Mesh Animations." In *Proceedings of ACM Transactions on Graphics* 24 (2005): 399-407. Los Angeles, California: ACM, 2005.

[23] Griffin, Timothy, Russell Main, and Claire Farley. "Biomechanics of Quadrupedal Walking: How Do Four-Legged Animals Achieve Inverted Pendulum-Like Movements?". *The Journal of Experimental Biology*, no. 207 (2004): 3545-58.

[24] Yamamoto, Susumu. "Effective Implementations of Multi-Dimensional Radix-2 FFT." *Computer Physics Communications* 125, no. 1–3 (2000): 1-7.

[25] Siebert, William Mc C. *Circuits, Signals, And Systems*. Illustrated ed. Vol. 2: Cambridge, MA: MIT Press, 1986.

[26] Fleischer, Max. "Method of Producing Moving-Picture Cartoons." edited by United States Patent Office. United States of America, 1917.

[27] McLaughlin, Tim, and Ann McNamara. Perception Based Animation. "Generating Animal Avatar Animation with Specific Identifiable Traits Based Upon Viewer Perception of Real Animals." Accessed on November 1, 2012. http://www.viz.tamu.edu/research/pbanimation/

[28] O'Neill, Rob. *Digital Character Development: Theory and Practice*. Burlington, MA: Morgan Kaufmann Publishers, 2008.

APPENDIX A

DESCRIPTION OF PYTHON SCRIPT FILES

The following are descriptions of the Python scripts included with this thesis.

Extraction.py – Extracts the joint motion data from the rotoscoped animation in Maya.

FFT.py – In a virtual environment for Python, using NumPy and SciPy, the sampled signal is upsampled and the FFT is performed. The FFT results are used in a form of the Fourier series to create a mathematical expression for a given joint's motion.

Synthesize.py – In Maya, this script uses the results from the FFT.py script to drive each joint's motion. Separate functions were created for each joint's motion.

SynthRig.py – Creates quadrupedal rig that will be used for creating the synthetic animation.

APPENDIX B

DESCRIPTION OF RESULTING VIDEO FILES

The following are descriptions of the videos included with this thesis.

Horse_01_persp_FINAL.mov – Perspective (3/4) view of the synthetic animation for the "Horse_01" example.

Horse_01_side_FINAL.mov – Side view of the synthetic animation for the "Horse_01" example.

Horse_01_walk_original.mp4 – Original video of "Horse_01" walking orthogonally to the camera.

Horse_02_persp_FINAL.mov – Perspective view of the synthetic animation for the "Horse_02" example.

Horse_02_side_FINAL.mov – Side view of the synthetic animation for the "Horse_02" example.

Horse_02_walk_original.mp4 – Original video of "Horse_02" walking orthogonally to the camera. This video shows black dots over the horse's joints, which was used as a point light display for research in the Perception Based Animation research group.

Horse_04_persp_FINAL.mov – Perspective view of the synthetic animation for the "Horse_04" example.

Horse_04_side_FINAL.mov – Side view of the synthetic animation for the "Horse_04" example.

Horse_04_walk_original.mp4 – Original video of "Horse_04" walking orthogonally to the camera.