Quadruped Animation: Skeleton Fitting and Real-Time Adaptation Using Control Parameters

Ariana Allsopp

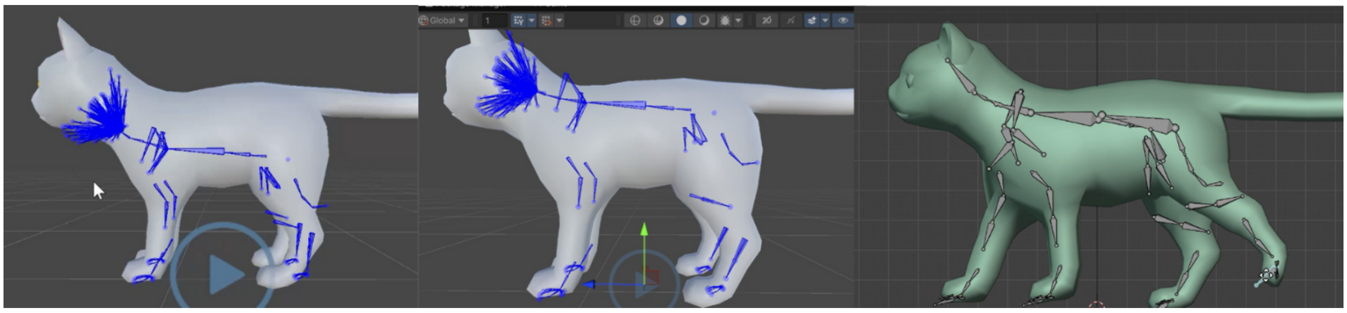

Figure 1: *The skeleton fitting and animation process. The left image is the skeleton in the mesh before altering the parameters, the second image is the same skeleton after altering the parameters to fit the mesh, and the third image is the skeleton being animated.*

Abstract

Quadruped animation presents significant challenges in computer graphics due to anatomical diversity and movement complexity. This project introduces a simplified skeleton-based animation system designed for real-time applications. Using predefined skeletons and three user-controlled parameters—pelvis-to-floor height, neck rotation, and foot configuration—the system achieves smooth and adaptable animations. Developed with Unity and Blender, it incorporates skeleton fitting, basic skinning, and inverse kinematics, ensuring intuitive control and efficient performance. Evaluations highlight strengths in usability, adaptability for species like cats and dogs, and joint accuracy. However, limitations remain in foot placement consistency and anatomical fitting for atypical quadrupeds such as horses.

Building on control parameters extracted from Principal Component Analysis (PCA) and existing animation frameworks, this system balances simplicity and functionality, offering a practical alternative to more complex solutions. Real-time parameter adjustment enables precise and user-friendly control, while the integration of inverse kinematics ensures motion fidelity. Despite challenges in stance width and foot placement accuracy for certain models, the approach provides a robust foundation for efficient and accessible quadruped animation.

This work contributes a streamlined, computationally efficient framework for animating quadrupeds, addressing a less-explored area compared to biped systems. By leveraging optimized control parameters and intuitive design, the system delivers high usability and adaptability while maintaining a focus on accuracy and efficiency. Future efforts can refine anatomical fitting and foot placement for broader applicability, further enhancing its potential for diverse animation tasks.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ *Computer graphics; Animation; Procedural animation; Inverse kinematics;* • ***Software*** $\rightarrow$ *Software development techniques; Frameworks;*

1. Introduction

Accurate and efficient animation of quadruped systems is a fundamental problem in computer animation, particularly in applications such as video games, virtual reality, and animated films. Quadrupeds pose unique challenges compared to their bipedal counterparts due to their diverse anatomy, complex movement, and variability across species. Unlike bipeds, quadrupeds rely on coor-

(12/2024)

dinated limb movements and varying stances, which require precise modeling and animation to achieve realistic motion. This complexity is extended by the anatomical differences between species, such as the elongated body of a horse versus the compact frame of a dog. As a result, a one-size-fits-all solution is impractical, and the lack of standardized tools or techniques has hindered the advancement of quadruped animation systems.

In recent years, the field has seen limited progress in developing tools specifically designed for quadruped animation. Existing methods often emphasize bipedal systems, leaving quadrupeds underrepresented. Accurate quadruped animation requires solving two interdependent problems: fitting a skeleton to the target model and generating realistic motion. Both tasks are challenging due to the anatomy and motion of quadrupeds. Furthermore, many current approaches involve highly complex systems that require significant computational resources or expertise, making them unsuitable for broader applications.

Previous works have addressed these challenges using both manual and automated methods. For instance, Reveret et al. (2005) introduced morphable models and principal component analysis (PCA) for skeleton fitting, while Baran & Popović (2007) developed automated skeleton fitting techniques within 3D meshes. Although these methods provide powerful tools for animating quadrupeds, their complexity often limits their accessibility and usability in practical environments. Moreover, these systems are frequently tailored to specific use cases, making them less adaptable to diverse quadruped models.

This project simplifies quadruped animation by introducing a user-friendly system that employs a predefined skeleton structure and three intuitive control parameters: pelvis-to-floor height, neck rotation, and foot configuration. These parameters allow real-time adjustments to adapt the skeleton to various quadruped models without the need for extensive manual intervention. By leveraging PCA-optimized parameters and integrating techniques like inverse kinematics and basic skinning, the system strikes a balance between simplicity and accuracy.

This paper details the development and implementation of the proposed system, emphasizing its adaptability, computational efficiency, and ease of use. Through qualitative and quantitative evaluations, the system demonstrates its effectiveness in addressing the challenges of quadruped animation, particularly for species like cats and dogs. By providing a streamlined approach to skeleton fitting and animation, this project contributes a practical framework for advancing quadruped animation systems in computer graphics.

2. Related Work

The development of adaptable skeleton-based animation systems for quadrupeds requires building upon existing methods in skeleton fitting, animation, and inverse kinematics. This section reviews key contributions in these areas, highlighting their relevance to this project and the specific methodologies that inspired its approach.

2.1. Skeleton Fitting

Reveret et al. (2005) created morphable models for quadruped skeletons [RFDC05], which is strongly related to the systems proposed in this project. They tested multiple rotation methods and coordinate systems, highlighting the advantages of quaternion-based rotations for interpolating between joint positions. In addition, they found that global coordinate systems prevent errors when dealing with hierarchical skeletal systems. Furthermore, Reveret et al. used three skeleton control parameters extracted from their Principal Component Analysis (PCA), which reduced the complexity of the structures. This project builds on these findings, using the same control parameters and emphasizing quaternion-based rotations to ensure joint accuracy and reduce animation errors. However, this project diverges by adapting the reference skeleton directly by fitting it into 3D models and prioritizing local coordinates for practical implementation. Moreover, the method used in this project streamlines parameter application for a simplified skeleton rather than relying on extensive models.

Baran & Popović (2007) applied a predefined skeleton model within a mesh, achieving an efficient method to rig 3D characters [BP07]. These concepts are similar to those in this project to simplify character rigging methods. Baran & Popović automate their skeleton fitting process, which differs from the use of control parameters to adjust the skeleton to the mesh in this project. However, both methods result in a minimal need for manual skeleton adjustment. This project integrates their principle of minimal manual adjustments by emphasizing parameterized control. However, it avoids their reliance on penalty functions and weighted vertex attachments, opting for simple control parameters and traditional skinning methods. They also used unique skin attachment methods by weighting mesh vertices and trained a penalty function to position the skeleton optimally within meshes. These methods increase the efficiency and accuracy of the fitting but are beyond the scope of the project. Baran & Popović included quadruped examples in their work, but primarily focused on human examples. This project adapts their efficiency principles for parameter control.

Wade and Parent (2000) and Reveret et al. (2005) used automatic skeleton generation for animation [WP00, RFDC05]. Wade and Parent used input parameters to convert models to voxelized forms, while Reveret et al. used joint placements to generate skeletons. These methods are too complex for this project, but follow similar principles of adapting skeletons for model animation. This project employs input parameters in both methods; however, it diverges from them by emphasizing a balance between manual control and automation, avoiding direct joint placement while maintaining simplicity.

2.2. Animation

Wilhelms & Van Gelder (1997) animated models with a complex anatomical animal modeling approach, including skin, muscles, and tissues [WVG97], and was one of the first methods of this kind. This system is too complex to implement in this project; however, it is the root of this animation system and inspired many other methods. This project draws inspiration from this work, but simplifies the anatomical approach to focus on skeleton fitting and animation. Specifically, their use of bounding boxes for anatomical modeling influenced the approach that this project took to skeleton fitting, although it concentrates solely on skeletal adjustments.

Inverse kinematics eliminates the need for key-framing without sacrificing smooth, continuous movement. Grochow et al. (2004) proposed a style-based inverse kinematics system based on a learning model of human motion capture data [GMHP04], which inspired the use of inverse kinematics in this project. However, this method is complex, so this system will use a solver to compute the joint angles. Avoiding style-based complexities enables this project to emphasize practical IK applications suitable for quadrupeds. More recently, Yang et al. (2021) used inverse kinematics to animate their model [YWM*21], which is the method that this project will use. They also applied the animations to the skeleton. The rest of the methodologies used by Yang et al. are far more complex and involve neural fields and networks that predict 3D shapes, joint positions, and skinning weights. Yang et al. use advanced predictive methods which are beyond the scope of this project, which adopts only the foundational IK solver for simpler and more focused implementation.

This project builds on many of these foundational methods by leveraging PCA-derived control parameters for skeleton fitting while avoiding the complexities of neural networks, voxelized forms, or complex anatomical systems. By focusing on quadrupeds and combining efficient skeleton fitting with practical IK solvers, this project provides a streamlined approach to quadruped rigging and animation that balances manual control and automation.

3. Overview

This project implements a skeleton-based animation system for quadruped animals. The system employs a predefined dog skeleton as the foundational structure and uses three control parameters to alter the skeleton. These parameters - height from the pelvis to the floor, neck rotation, and foot configuration (hoofed vs. plantigrade) - are adapted from the study by Reveret et al. (2005) [RFDC05], where they were optimized using Principal Component Analysis (PCA). After the control parameters are adjusted to fit, quadruped meshes, primarily cats and dogs, are applied and bound to the skeleton through skinning. Subsequently, inverse kinematics (IK) are applied to the skeleton joints, enabling smooth and fluid animations for the modified skeletons. This work emphasizes adaptability and user interaction during parameter adjustments.

3.1. Control Parameters

The animation system centers on three control parameters derived from Reveret et al. (2005) [RFDC05]. These parameters were chosen for their simplicity, effective control, and computational efficiency. Figure 2 depicts each of the control parameters against the reference skeleton:

1. **Pelvis-to-Floor Height** (h): The height from the top of the pelvis to the floor, governs the vertical scaling of the skeleton's leg segments.

2. **Neck Rotation** (θ): Alters the rotation of the upper spine and neck segments to adjust head direction.

3. **Hoofed vs. Plantigrade Configuration** (ϕ): Modifies the angle between the floor and the rear leg to reflect different quadrupedal stances.

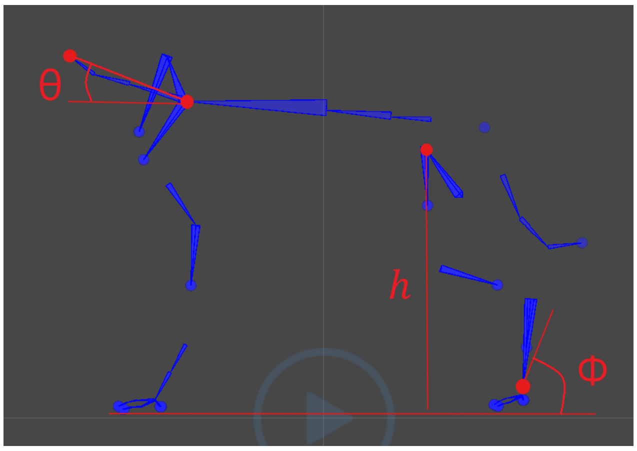

Figure 2: *The three control parameters, pelvis-to-floor height (h), neck rotation (θ), and ankle rotation (ϕ)*

The choice of these parameters is justified by their effectiveness in controlling a quadruped skeleton while maintaining computational simplicity, as validated by PCA optimization in Reveret et al.'s study. Each parameter influences the skeleton through a series of transformations. The control parameters were implemented in Unity.

3.1.1. Pelvis Height

The pelvis height parameter h is implemented by modifying the positions and lengths of the skeletal components, including the pelvis, upper body, and leg segments. The vertical position of each segment is adjusted to reflect the new height difference Δh.

1. **Pelvis and Upper Body Adjustment:** The pelvis position is updated as:

$$z'_{\text{pelvis}} = z_{\text{pelvis}} + \Delta h \tag{1}$$

where z_{pelvis} is the current pelvis height and $\Delta h = h - h_{last}$ is the height difference. Similarly, the upper body is adjusted to maintain consistency:

$$z'_{\text{upper}} = z_{\text{upper}} + \Delta h \tag{2}$$

2. **Leg Segment Scaling:** Each leg segment is scaled proportionally to half the height difference:

$$z'_{\text{leg-top,i}} = z_{\text{leg-top,i}} + 0.5 \cdot \Delta h, \quad i \in \{FL, FR, BL, BR\} \tag{3}$$

Each leg is scaled to ensure realistic proportions. The scaling is distributed across the upper and lower legs using linear interpolation between the height of the pelvis and upper half of each leg with $\alpha = 0.5$.

3.1.2. Vertebral Bend Adjustment

The spine bend angle θ modifies the orientation of various segments of the spine and neck, and is implemented as a rotation about the local x-axis of the vertebral column. The rotation angle is given by θ, and the corresponding quaternion Q_{bend} is calculated using the formula for rotation of the axis angle quaternion:

$$Q_{\text{bend}} = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right) \cdot (x\mathbf{i} + y\mathbf{j} + z\mathbf{k}), \tag{4}$$

where:

- θ is the spine bend angle in radians,
- (x, y, z) is the normalized rotation axis (for the spine bend, $\mathbf{v}_{\text{axis}} = (1, 0, 0)$).

This simplifies to:

$$Q_{\text{bend}} = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\mathbf{i}. \tag{5}$$

The transformation is applied to the local rotation of the upper neck, lower neck, and front upper body:

$$\mathbf{p}' = Q_{\text{bend}} \cdot \mathbf{p} \cdot Q_{\text{bend}}^{-1}, \tag{6}$$

where:

- $\mathbf{p}$ is the position of the point being transformed in quaternion form,
- Q_{bend}^{-1} is the conjugate of Q_{bend}, given by:

$$Q_{\text{bend}}^{-1} = \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right)\mathbf{i}. \tag{7}$$

This rotation is applied to three segments in the neck to ensure smooth rotation.

3.1.3. Ankle Angle Adjustment

The ankle angle adjustment modifies the lower leg rotation to achieve the desired stance (hoofed vs. plantigrade).

1. **Ankle Rotation:** The rotation angle ϕ is applied about the local x-axis. The corresponding quaternion Q_{ankle} is defined the same as Equation 4. For the ankle adjustment, the axis $\mathbf{v}_{\text{axis}} = (1, 0, 0)$, so:

$$Q_{\text{bend}} = \cos\left(\frac{\phi}{2}\right) + \sin\left(\frac{\phi}{2}\right)\mathbf{i}. \tag{8}$$

This rotation is applied to the lower leg's local transformation:

$$\mathbf{p}' = Q_{\text{ankle}} \cdot \mathbf{p} \cdot Q_{\text{ankle}}^{-1}, \tag{9}$$

2. **Foot Translation:** The foot's original position before the rotation was at $\mathbf{p}_{\text{foot,orig}}$. After rotation, the new foot position $\mathbf{p}_{\text{foot,rot}}$ may no longer be aligned with the leg. To correct this, we calculate a compensatory translation $\Delta\mathbf{p}$ to shift the foot to its intended location. The translation vector is given by:

$$\Delta\mathbf{p} = \mathbf{p}_{\text{foot,orig}} - \mathbf{p}_{\text{foot,rot}}, \tag{10}$$

This ensures that the foot remains stationary on the ground despite the rotation of the lower leg. The final position of the foot $\mathbf{p}_{\text{foot,new}}$ is:

$$\mathbf{p}_{\text{foot,new}} = \mathbf{p}_{\text{foot,rot}} + \Delta\mathbf{p}. \tag{11}$$

The use of control parameters for skeleton adjustments prioritizes simplicity and user interaction, making the system accessible to non-experts. However, this approach assumes that the input meshes align with the predefined skeleton anatomically. Significant variations in limb proportions or posture may reduce the system's adaptability. Extending the parameter set could address this limitation in subsequent developments.

3.2. Skeleton Implementation and Skinning

A predefined dog skeleton was selected from Unity's Asset Store for its accessibility and suitability. The skeleton chosen is from the pack "3D Stylized Animated Dogs Kit" created by Bublisher [Bub24]. Meshes were chosen for their availability and compatibility with the skeletal structure, and were primarily sourced from "Animals FREE - Low Poly Asset Pack" by ithappy on the Unity Asset Store [ith24].

After the skeletons were adjusted using the control parameters in Unity, they were exported and imported into Blender for further processing. This step ensures that the skeletons are properly positioned and bound to the corresponding quadruped meshes, finalizing their readiness for animation.

3.2.1. Skeleton Positioning and Adjustment

Once imported into Blender, the skeletons were carefully positioned within the meshes they were previously scaled and adjusted for. This step ensures that the skeleton aligns accurately with the anatomical structure of the mesh. Any discrepancies between the skeleton and the mesh, such as misaligned joints or improper bone scaling, would typically be corrected at this stage. However, for this project, no manual adjustments were made to the joints or bones beyond those applied through the control parameters. This decision was made to maintain consistency and to evaluate the effectiveness of the control parameters alone in adapting the skeletons. If future projects were to incorporate models with more complex anatomy, this step could involve detailed realignment or even additional bone creation to fit unique anatomical structures.

The decision to avoid manual adjustments in Blender is based on the goal of evaluating the effectiveness of the control parameters in adapting the skeletons to the meshes. This should be demonstrated in the animations, to fully display the capabilities of the system. While manual adjustments could correct minor fitting issues, relying solely on parameter-based transformations ensures consistency across different models and highlights the efficiency of the approach. However, this limitation may result in slightly less precise fits for more complex models, which could be addressed in future work by integrating semi-automated adjustment tools.

3.2.2. Binding Skeletons to Meshes

After alignment, the skeletons were bound to the meshes using Blender's automatic weighting feature. This process calculates the influence each bone has over nearby vertices, allowing the mesh to deform naturally when the skeleton moves. The use of automatic weighting was particularly successful due to the low-poly nature of the models. Automatic weighting was utilized to reduce complications, as the main focus of this project is the control parameters of the skeleton. This solution was significantly faster to implement, and reqired little adjustment for the low poly meshes.

3.3. Animation Creation

Once the skeletons were skinned to the meshes, animations were created in Blender. The animation process relied on a combination of:

Inverse Kinematics (IK): Blender's IK solver was used to create natural joint movements by specifying the desired positions of end effectors (e.g., paws). The IK solver computed the intermediate joint angles, ensuring smooth and realistic motion. The incorporation of IK follows the principles outlined by Grochow et al. (2004) and Yang et al. (2021) [GMHP04, YWM*21], emphasizing computational efficiency and natural motion.

Keyframing: The target positions for the IK solver were keyframed to define specific poses over time. This technique allowed for precise control over the timing and flow of animations.

3.4. Optimization Framework (proposed)

The control parameter values are dependent on the user's adjustment and visual comparison between the mesh and the skeleton. This process can be optimized by setting the control parameters to values that minimize the distance from the ideal joint positions within the mesh:

$$\text{minimize} \ \ f(c) = \sum_{j=1}^{m} ||\mathbf{p}_j(c) - \mathbf{p}_j^*||^2 \tag{12}$$

where:

- $\mathbf{p}_j(c)$ represents the position of the j-th joint as a function of the control parameters c,
- $\mathbf{p}_j^*$ is the target position of the j-th joint within the mesh,
- m is the total number of joints,
- $||\cdot||$ denotes the Euclidean norm.

The control parameter vector $\mathbf{c} = [h, \theta, \phi]$ contains the control parameters to be optimized:

- h: pelvis to floor height,
- θ: vertebral bend angle,
- ϕ: ankle rotation angle.

The optimization would be performed using an external library or optimization package, which provides robust algorithms such as gradient descent, L-BFGS, or other numerical solvers. The external package would handle the iterative adjustment of c to minimize $f(c)$. Examples of suitable packages include SciPy's optimize.minimize or machine learning frameworks like PyTorch or TensorFlow.

As the scope of this project does not include implementing the optimization algorithm, prebuilt tools would be used to set the control parameter values to minimize joint distances.

Although the optimization is currently proposed rather than implemented, it outlines a path for automating control parameter adjustments. By minimizing the distance between actual and target joint positions, this functionality would reduce user dependency for parameter adjustment. However, the computational cost of real-time optimization is a challenge that would need to be addressed. Efficient optimization algorithms, such as gradient-based methods or evolutionary algorithms, could be explored to balance accuracy and performance.

4. Evaluation

The evaluation of this project focuses on measuring the system's ability to adapt and animate quadruped models through both qualitative and quantitative measures. The skeleton-based animation system is analyzed for system performance, the control parameter joint position error, and the inverse kinematics solver quality. This section highlights the strengths and weaknesses of the system, as well as various computational limitations.

4.1. Qualitative Measures

The qualitative evaluation of this project focuses on demonstrating the system's ability to fit and animate quadruped models effectively, highlighting both its strengths and limitations. By showcasing the skeleton fitting process, animation quality, and real-time control adjustments, this section provides insight into the system's functionality and areas for improvement.

4.1.1. Skeleton Fitting

The skeleton-fitting process effectively adjusted most components to fit similar quadruped models, such as cats and dogs. The control parameters, which are height from pelvis to floor, neck rotation, and leg positioning, successfully allowed the skeleton to adapt to tested cat meshes. Specifically, the height and neck adjustments allowed for alignment within the torso and head of the meshes.

However, certain skeleton components without attached control parameters, such as tail rotation and length, caused issues with the skeleton fitting. Additionally, the width of the stance of the skeleton—where the front legs are positioned closer together than the back legs—made it difficult to fully position the skeleton within meshes that do not share this characteristic. These issues highlight areas for potential refinement of the system.

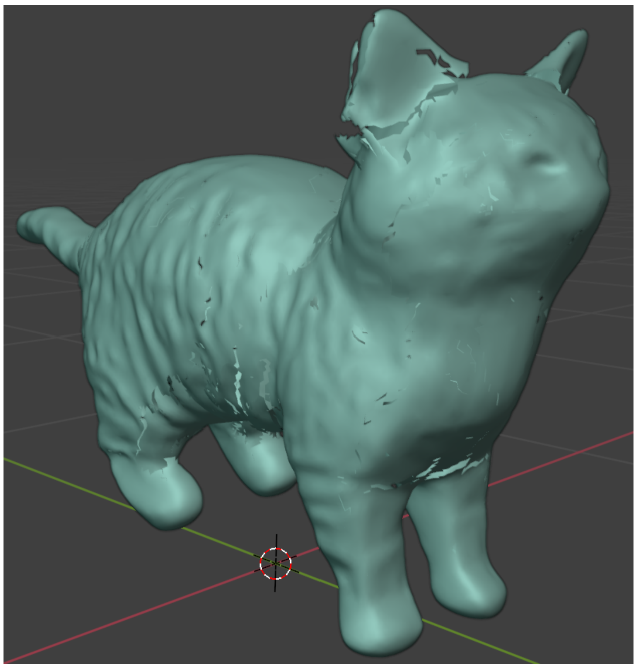

Figure 3: *The mesh of a cat, skinned on the skeleton with automatic weights. Due to the texture in the fur, the weighting is off and results in holes in the mesh*

4.1.2. Animation Quality

The animations depict smooth joint movements, without much visible error or incorrect rotations. Due to the use of automatic weights, certain meshes have some artifacts during the animations. This is more prevalent on one of the cat meshes due to the texture in the fur, as shown in Figure 3.

4.1.3. System Usability

Adjusting the control parameters in real time proved to be an easy and intuitive method for fitting the skeleton to various meshes. This approach significantly reduced the complexity of manual skeleton adjustments by tying significant bone and joint controls into three parameters. However, the application of foot placement and rotation presented challenges. The rotation of the leg segment for the hoofed vs. plantigrade parameter required adjustments to the foot position to maintain connection at the ankle. This adjustment introduced slight inconsistencies in foot placement, especially after large rotations, where the foot would not return exactly to its original position, as shown in Figure 4.

4.2. Quantitative Measures

The quantitative evaluation analyzes the system's performance through objective metrics, including frame rate consistency, joint position accuracy, and computational efficiency. This section provides a detailed description of how well the system achieves its goals under various configurations and identifies areas for optimization.

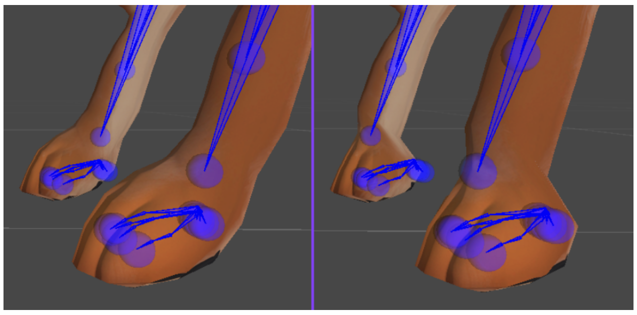

Figure 4: *The ankle configuration with the control parameter for the ankle angle set at the same value of -110°. The image on the left depicts the initial configuration, while the image on the right shows the result after large parameter changes and subsequent adjustments. This comparison highlights the slight inconsistencies in foot placement caused by extensive rotations, illustrating an area for potential refinement in parameter interaction handling*

4.2.1. System Performance

The system maintained an average of 100 FPS when control parameters were not being altered, and dropped to an average of 80 FPS while altering control parameters, with an FPS as low as 65 for extreme parameter adjustments. This remained consistent across all tested meshes and control parameter configurations, only slowing down momentarily when the parameters are rapidly changed to extreme values.

During animations, the system maintained an average framerate of 30 FPS, and demonstrated consistent across all tested meshes and control parameter configurations.

4.2.2. Joint Accuracy

The data for skeleton fitting is derived from a mesh that aligns well with the skeleton, ensuring that no joints or bones require significant repositioning for the skeleton to function minimally within the model. This data is largely consistent across models of this type, with each joint requiring only minor adjustments to reach its ideal position. Table 1 contains all of the optimal joint positions $\mathbf{p}_i^*$ and actual joint positions $\mathbf{p}_i$ of the skeleton fit to a cat mesh, and the euclidean distance error e_i for each joint, as given by:

$$e_i = ||\mathbf{p}_i - \mathbf{p}_i^*|| = \sqrt{(x_i - x_i^*) + (y_i - y_i^*) + (y_i - y_i^*)} \qquad (13)$$

where:

- (x_i, y_i, z_i): Coordinates of the actual position of the i-th joint,
- (x_i^*, y_i^*, z_i^*): Coordinates of the optimal position of the i-th joint.

For the entire skeleton, the Euclidean distance error is:

$$E = \sqrt{\sum_{i=1}^{n} ||\mathbf{p}_i - \mathbf{p}_i^*||^2} \qquad (14)$$

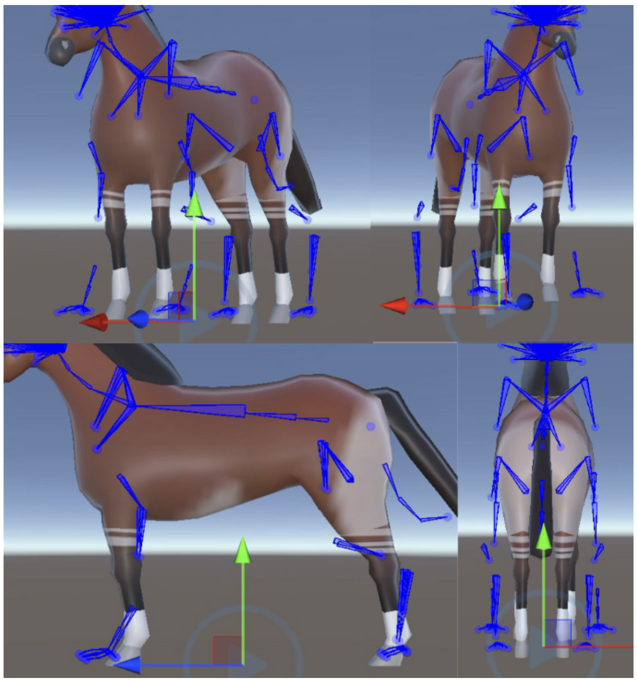

Figure 5: *The skeleton adjusted to fit the mesh of a horse model. Due to the control parameters' inability to adjust the width of the skeleton's stance or the torso length, the skeleton appears too wide for the horse's narrower body and elongated proportions. This demonstrates a limitation of the current system when working with anatomically distinct quadrupeds, such as horses, compared to cats or dogs.*

The total error E aggregates these individual errors across all joints. This metric provides a quantitative measure of how well the skeleton aligns with the desired configuration. The total error for the cat mesh is 0.2714, measured in meters, the default unit in Unity. For the system, this error is relatively small, and depicts the joint locations as being accurate overall.

Models that do not function minimally within the model exhibit significantly reduced joint accuracy and necessitate extensive bone repositioning to achieve adequate alignment. This limitation arises because certain aspects of the skeleton, such as the width between the legs and the body length, are not influenced by the control parameters. For example, models like horses, which have elongated bodies and narrower stances, cannot accommodate the skeleton, even with adjustments to the control parameters. Figure 5 depicts the skeleton adjusted to the mesh of a horse, and demonstrates how the stance of the horse is too narrow for the skeleton when it is sized to match the torso length.

4.2.3. Computational Limits

While the system showed consistent performance overall, some computational inefficiencies were observed in scenarios with complex animations or large parameter adjustments. These inefficiencies are tied to the adjustments for foot placement, which introduced slight inaccuracies in maintaining the original positions during extensive rotations. These limitations underscore the need for further optimization of parameter interactions and foot positioning algorithms.

Table 1: *Comparison of ideal and actual joint locations, with computed positional errors.*

Joint Locations (in m)			
	Ideal (x, y, z)	Actual (x, y, z)	Error
Head	0 0.031000 0.032000	$-2.4447 \cdot 10^{-9}$ 0.047556 $-5.9754 \cdot 10^{-9}$	0.036029
Neck Base	0 0.22600 0.037000	$-1.0100 \cdot 10^{-13}$ 0.22861 $5.5879 \cdot 10^{-9}$	0.037092
Shoulder (L)	−0.091000 0.043000 −0.036000	−0.091355 0.048607 −0.061434	0.026047
Pelvis Base (L)	−0.036000 −0.011000 0.11000	−0.036166 0.020024 0.11560	0.031525
Hip (L)	−0.0040000 0.098000 0.024000	$-3.7253 \cdot 10^{-9}$ 0.096253 $-3.2596 \cdot 10^{-9}$	0.024394
Front Ankle (L)	−0.025100 0.051100 −0.00070000	$2.7940 \cdot 10^{-9}$ 0.051228 $-1.4435 \cdot 10^{-8}$	0.025110
Front Knee (L)	−0.020000 0.12900 0.011000	$-1.0245 \cdot 10^{-8}$ 0.096954 $-1.1176 \cdot 10^{-8}$	0.039344
Back Ankle (L)	−0.13200 0.25354 0.036000	−0.10524 0.25354 0.036000	0.026763
Back Knee (L)	−0.13390 0.33600 0.19300	−0.10876 0.33600 0.19300	0.025142

All of the quantitative measurements of the animation system were measured on a standard machine (Intel Core i7-12700H CPU @ 2.30 GHz, 16GB RAM, Intel® Iris® Xe Graphics, Windows 11).

4.3. Discussion

The evaluation demonstrates that the skeleton-based animation system effectively fits and animates quadruped models. Key strengths include the intuitive real-time adjustment of control parameters, smooth joint animations, and consistent system performance. However, areas for improvement include addressing limitations in fitting components without control parameters, such as tail length and stance width, and optimizing foot positioning algorithms to minimize placement inconsistencies after large rotations. Future improvements could involve refining the parameterization of the skeleton to include additional controls for currently unsupported components, as well as enhancing the computational efficiency of foot adjustments. Expanding the range of tested quadruped models will further validate the system's adaptability and robustness.

5. Conclusion

This project successfully developed a simplified skeleton-based animation system for quadruped models, focusing on adaptability and computational efficiency. By using pre-existing skeleton structures and control parameters optimized through Principal Component Analysis (PCA), the system provides a streamlined approach to fitting and animating quadrupeds. However, several challenges and limitations emerged during implementation.

The skeleton chosen for this project presented issues that impacted its performance. It included an excessive number of face bones and lacked proper connections between certain leg bones, requiring manual adjustments in Blender to address parenting and connectivity issues. Although these adjustments ensured functional skeleton movement, they highlighted the limitations of the predefined skeleton used in the system.

The system demonstrated strong performance with low-poly models, where the reduced vertex count minimized the risk of incorrect weight assignments, resulting in clean deformations and smooth animations. However, high-poly meshes posed significant challenges, including overlapping vertices, distortion during movement, and artifacts caused by the automatic weighting process. Addressing these issues for high-poly models would require manual weight painting to refine deformations, especially around joints like shoulders and hips.

Future work should focus on addressing these limitations. Incorporating advanced weight painting techniques can improve the accuracy and visual fidelity of animations, particularly for complex models. Additionally, implementing animation retargeting tools would enable efficient reuse of animations across models with similar skeletal structures, improving the system's scalability. Automating the parameter optimization process and expanding the system to support a broader range of quadruped species with diverse anatomical structures would further enhance its robustness and applicability across various domains.

References

[BP07] BARAN I., POPOVIĆ J.: Automatic rigging and animation of 3d characters. In *ACM SIGGRAPH 2007 Papers* (2007), p. 72. URL: https://doi.org/10.1145/1275808.1276467, doi:10.1145/1275808.1276467.

[Bub24] BUBLISHER: 3D Stylized Animated Dogs Kit. *Unity Asset Store*, 2024. Accessed: November 2024. Available at: https://assetstore.unity.com/packages/3d/characters/animals/mammals/3d-stylized-animated-dogs-kit-284699.

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIĆ Z.: Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3 (2004), 522–531. URL: https://doi.org/10.1145/1015706.1015755, doi:10.1145/1015706.1015755.

[ith24] ITHAPPY: Animals FREE - Low Poly Asset Pack. *Unity Asset Store*, 2024. Accessed: November 2024. Available at: https://assetstore.unity.com/packages/3d/characters/animals/animals-free-low-poly-asset-pack-by-ithappy-260727.

[RFDC05] REVERET L., FAVREAU L., DEPRAZ C., CANI M.-P.: Morphable model of quadrupeds skeletons for animating 3d animals. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 135–142. URL: https://doi.org/10.1145/1073368.1073386, doi:10.1145/1073368.1073386.

[WP00] WADE L., PARENT R. E.: Fast, fully-automated generation of control skeletons for use in animation. In *Proceedings of Computer Animation 2000* (2000), pp. 164–169. URL: https://doi.org/10.1109/CA.2000.889075, doi:10.1109/CA.2000.889075.

[WVG97] WILHELMS J., VAN GELDER A.: Anatomically based modeling. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '97* (1997), pp. 173–180. URL: https://doi.org/10.1145/258734.258833, doi:10.1145/258734.258833.

[YWM*21] YANG Z., WANG S., MANIVASAGAM S., HUANG Z., MA W.-C., YAN X., YUMER E., URTASUN R.: S3: Neural shape, skeleton, and skinning fields for 3d human modeling. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021). URL: https://doi.org/10.1109/cvpr46437.2021.01308, doi:10.1109/cvpr46437.2021.01308.