D5.9 Tools for synthesizing animation without a rig

Grant Agreement nr	780470
Project acronym	SAUCE
Project start date (duration)	January 1st 2018 (36 months)
Document due:	June 30th 2020
Actual delivery date	June 30th 2020
Leader	DNEG
Reply to	Mungo Pay - mungo@dneg.com
Document status	Submission Version

Project funded by H2020 from the European Commission

Project ref. no.	780470
Project acronym	SAUCE
Project full title	**S**mart **A**sset re-**U**se in **C**reative **E**nvironments
Document name	D5.9 Tools for synthesizing animation without a rig
Security (distribution level)	Public
Contractual date of delivery	June 30th 2020
Actual date of delivery	June 30th 2020
Deliverable name	Tools for synthesizing animation without a rig
Type	Demonstration
Status & version	Submission Version
Number of pages	33
WP / Task responsible	DNEG
Other contributors	Trinity College Dublin (TCD)
Author(s)	Mungo Pay - DNEG, Ewan Rice - DNEG, David Reeves - DNEG, Pisut Wisessing - TCD
EC Project Officer	Ms Adelina Cornelia Dinu - adelina-cornelia.dinu@ec.europa.eu
Abstract	Animation re-use in the VFX industry is critical when attempting to reduce the costs associated with asset production. The creation of bespoke animations will be handled by either the Animation or Motion Capture departments, but relying on them entirely to produce all required animation is impractical, especially when minor edits to previously created animation could be used. This deliverable investigates two use cases where simple animation editing is desirable without the requirement of the full rig being available. The use cases are that of footstep cleanup and terrain adaptation for crowd scenes, and pre-roll generation for CFX tasks.
Keywords	Inverse Kinematics, Animation, Shape interpolation, Pre-roll
Sent to peer reviewer	Yes
Peer review completed	Yes
Circulated to partners	No
Read by partners	No
Mgt. Board approval	No

Document History

Version and date	Reason for Change
1.0 05-05-20	Document created by Dr Mungo Pay
1.1 25-06-20	Version for internal review
1.2 30-06-20	Revisions in response to review: final version submitted to Commission

Table of Contents

1 EXECUTIVE SUMMARY

This document provides details of the work done for deliverable D5.9 titled "Tools for synthesising animation without a rig" within work package 5 (WP5) of the SAUCE project. The work in this document was carried out by two consortium partners, DNEG and TCD and focuses on two areas of research, namely an animation adaptation tool for footstep cleanup and terrain adaptation, and a pre-roll animation generation tool.

Section 2 provides some background of the document and how it relates to previous deliverables.

Section 3 gives some background to the problems being tackled in this deliverable so that the reader is clearly informed as to the reasoning why this work is being undertaken.

Section 4 details the work done for the terrain adaptation and footstep cleanup in the context of crowd simulation.

Section 5 sets out the pre-roll generation problem, and then describes two alternative solutions to the problem.

Sections 6, 7, 8, and 9 cover conclusions, references, web references and acronyms and abbreviations.

2 BACKGROUND

This deliverable continues the work done for work package 5 on asset transformation, focussing on animation. Whilst the implementations and problem spaces covered range for the different use cases, the three sections of this deliverable aim to address a common problem within the VFX industry, that of animation reuse.

Chapter 4 of this deliverable builds on the work done in deliverables D5.4 and D5.5 which focussed on animation synthesis. Chapter 5 explores a new problem of pre-roll generation and investigates the feasibility of two different techniques. The first attempts to generate a skeleton from the mesh so that standard skeletal interpolation techniques can be applied. The second uses shape interpolation techniques to maintain properties of the mesh whilst interpolating between shapes.

3 INTRODUCTION

The animation department, in any VFX or feature animation studio, is fundamental to producing the realistic looking motion that breathes life into the characters on screen and allows cinemagoers to connect with the once static 3D geometry. They are able to do this work because of the work done by the rigging department, who provide animation controls that allow the animator to move the character in a "believable" way. A standard rig comprises two key parts, the kinematic model, and the deformation model. The kinematic model determines how the movement of the animation controls will change the position of the character's skeleton. The deformation model determines how the mesh will be deformed based on this skeleton and other inputs. There are many techniques and tricks that go into the creation of a high quality animation rig, but these are the two fundamental concepts.

When creating a rig, riggers will balance quality and complexity to produce the best results for the job at hand. As the number of modules in a rig increases, the animator is likely to get more realistic or controllable results, but this will likely be at the cost of computational complexity and therefore speed. If a character is going to be very close to screen, perhaps a more complex rig is required. If many characters need to be on screen at once, but they're further from the screen, perhaps performance is the greater requirement so the rig complexity would be reduced.

Whilst this goal of bringing characters to life is fundamental to the success of any project, an animator's work also provides a starting point for many other departments in the 3D pipeline. Along with the mocap department, they produce keyframed skeletal animation that can be used by departments where having a full rig in scene no longer makes sense. This is true for crowd simulations where there can be a requirement to have thousands or perhaps even hundreds of thousands of characters in a scene. This kind of scale is not practical for complex rig setups.

There are also situations where the representation of a rig is tied to a specific DCC, so there is no good translation mechanism into another DCC where other work is to be done. At DNEG, this is the case for CFX work done in SideFX Houdini™ because our rig representation is tied to a scene representation in Autodesk's Maya™. In this situation, the final animated character mesh is exported with no skeleton nor animation controls with which the animation can be adjusted. In the case of this CFX work, pre-roll animation is often required to allow for a buffer for either a hair or cloth simulation to settle in a reasonable state from an initial start position.

These two examples highlight a bottleneck in the production pipeline that can require many days of iterations to get appropriate animation for the task at hand. The work done described in this deliverable attempts to allow artists to modify the animation they will use for their task without a requirement to kick the work back to the animation department. The tooling described focuses on two areas: Terrain adaptation and footstep cleanup using IK techniques on pre-baked skeletal animation, and an automatic pre-roll generation tool to allow CFX artists to create this pre-roll themselves from a deforming mesh. The pre-roll work is split into two different implementations to evaluate different methodologies. The first method attempts to generate a skeleton from the geometry which artists can then pose using keyframes to generate their pre-roll, the second uses shape interpolation techniques to edit the mesh directly.

The hope for these tools is to reduce the extra artist time requiring data from other departments, and to speed up the turnaround of shots.

3.1 Main objectives and goals

- Produce a framework that allows crowd artists to ingest keyframed animation and generate new animation that adapts the character footsteps to an input terrain.
- Create a tool that allows artists to ingest an animated mesh, and generate pre-roll animation for CFX work.

3.2 Methodology

For the work in Chapter 4 on footstep cleanup, the methodology was to extend the work done in deliverables D5.4 and D5.5 where a C++ animation datatype was developed. This datatype allows us to edit the animation in an efficient and predictable way. Using this datatype, footstep detection techniques could be used to determine when a character had their feet on the ground, and using inverse kinematics, the animation could be adapted to follow an input terrain.

For the pre-roll tool, two approaches were attempted, as detailed in Chapter 5. The first, looked to automatically generate a skeleton and skinning weights based on the input mesh. If this skeleton can be reliably generated, pre-roll animation could be created by simply posing the skeleton at particular keyframes, and using the skinning to generate appropriate mesh deformation. It would also add the potential that some of the other path editing tooling developed in D5.5 could be used to produce richer and more complex pre-roll animations if ever required. This tooling was developed using Houdini, which is the target application for the majority of CFX at DNEG.

The second approach to the pre-roll problem, also detailed in Chapter 6, attempts to use shape interpolation techniques to be able to interpolate between the mesh posed in different positions. Whilst linear vertex blending techniques will often produce scaling and skewing artifacts, by using laplacian minimisation techniques, these artifacts can be ameliorated. The tooling was developed in C++ as a standalone tool, and then wrapped in a Houdini node so that it could be used by CFX artists at DNEG.

3.3 Convention

In this deliverable will use *italics* for emphasis and `monospace` for code and pseudo code.

3.4 Relation to the Self-Assessment Plan (D1.2)

As this work package was devised and added to the project when it was underway, there is no mention of this work package or its deliverable in the self assessment plan. However, since it is similar in nature to the work described in WP5T2 and WP5T3, it feels appropriate that it should have a similar testing plan, namely to devise some standard tests, and evaluate the speed of turnaround of shots when using the tooling and when using older techniques. This evaluation will take place in months M25-36, as part of work packet 8 in deliverable D8.3.

4 Footstep cleanup and end effector handling for crowds

At DNEG, for reasons of efficiency, crowd artists use keyframed skeletal animations to drive the agents. This animation data generally comes from mo-cap sessions. As crowds require a large number of characters potentially all following their own unique paths and performing different behaviours, it is not feasible for animators to author bespoke animations for every character. Even crowds with a low number of agents where this may be possible, the animations will often need to be edited when the environment is updated, the layout of the crowd is altered, or when someone isn't satisfied with the animation of particular agents. Going back to an animator for every edit is often a very slow process, so a crowd artist needs to be able to edit animation within their scene. As the rigs animators use to manipulate a skeleton are often very complex to process and dependant on a particular DCC, it's not possible to use them in a live crowd scene. Instead, animations are baked as transforms for joints for each frame of animation, which can then be imported into Houdini for crowd work to begin. With baked skeleton data we can fulfill many crowd requirements by looping, trimming, blending and layering animations, which are all relatively simple operations to perform and can rely on the joint transforms of the source data when producing any new poses.

Situations where a joint needs to be transformed a large distance, such as with terrain adaptation, require an Inverse Kinematics (IK) system in order to calculate realistic transforms for all the joints in the chain of the transformed joint.

4.1 Terrain Adaptation

Crowd agents may need to walk over bumpy, sloped and continuously changing terrain that are subject to change throughout a production. We cannot capture mo-cap or author animations for any variation in an environment and so we must adapt our existing animations to be able to realistically traverse their environment.

To achieve this we project the root of an animation to smoothly follow the height of the ground that the agent is traversing. To maintain appropriate foot positions, we derive transforms for the ankle joints that keep the foot in contact with the ground when it is not lifted, and preserve the roll of the foot as it steps on the ground.

As well as adapting to uneven terrain, this can be used to walk up and down stairs. Stairs are generally a more difficult geometry for which to produce a realistic pose due to the discrete change in the height of each step. Additionally, this technique is useful to eliminate foot sliding by locking the positions of the foot joints while they are in contact with the ground.

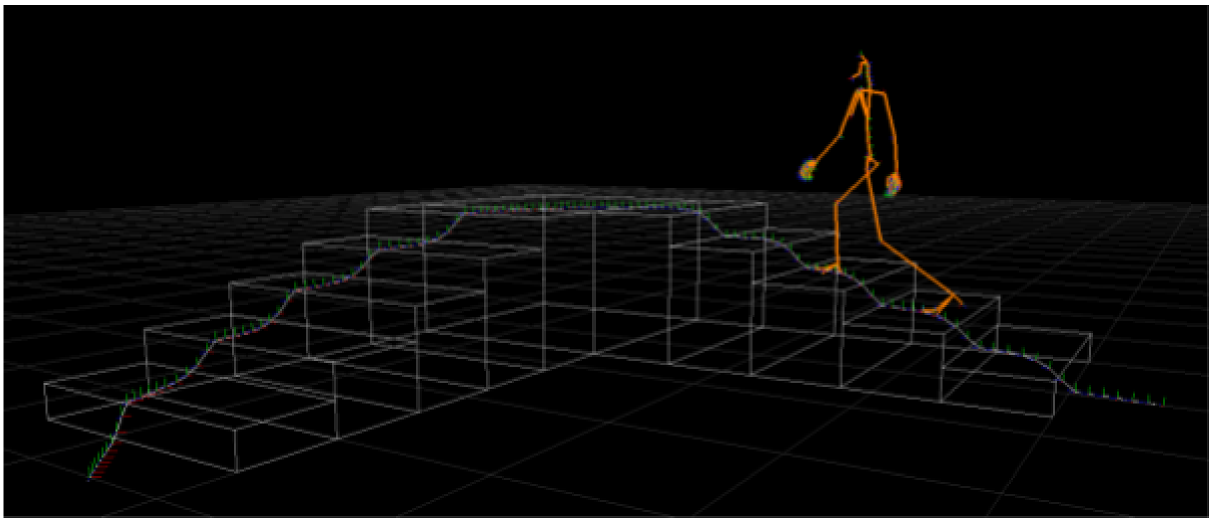


Figure 4.1 Flat walk animation traversing stairs.

4.1.1 Projecting to the Ground and Smoothing Root Projection

The first step in the terrain adaptation process is to project the root joint transform of the animation onto the ground. We also use ground projection in detecting footsteps and calculating the target ankle transforms.

From any joint position we find a projected point on the ground by shooting a ray from that point to find an intersection with the ground geometry. The direction of the ray is downwards (-y) by default, but this is editable if required.

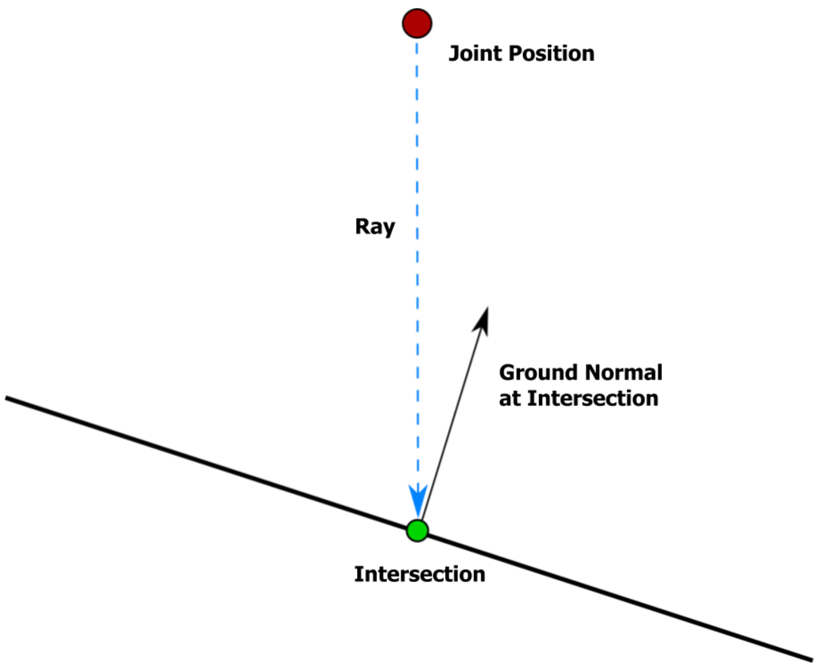


Figure 4.2 Intersection of a ray, shot from a joint position, with the ground geometry.

For continuously smooth terrain, this projected point can be used as the root joint position, keeping the animation at a consistent height above the ground. However, for stairs or bumpy terrain we need to smooth out the projected points over the neighbouring frames to avoid a discontinuous change in height of the characters animation. Using walking upstairs as an example, for frame n if the projected position on frame n+1 intersects with the next step, we do not want the animation to suddenly jump up by the height of the step. We avoid this by calculating the projected root position for each frame, and applying a damped spring filter to each [Clavet 2016].

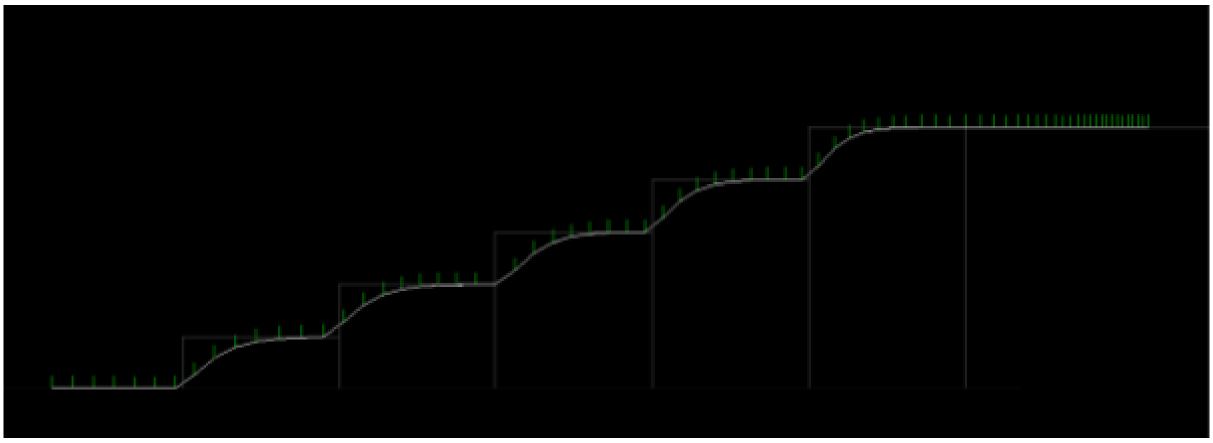


Figure 4.3 Spring damped root projection over stairs.

4.1.2 Detecting Footsteps

In order to clean up footsteps we must first detect when the foot joints are in contact with the ground. The foot of the DNEG archetype biped crowd skeleton generally comprises three joints: the heel and ball joints, both parented to the ankle, and a toe joint parented to the ball. To detect the foot down points, we analyse these joints at every frame, and tag them if it is found to be down. This provides a frame range for each joint of when it is in contact with the ground.

A simple metric is used to automatically detect when the joints are down, combining joint speed and height [Pražák 2012]. For each frame n of an animation the speed and height of the animation are derived as follows —

$$speed_n = \sqrt{(jointPosition_{n+1} - jointPosition_{n-1})^2}$$ *Figure 4.4*

$$height_n = jointPosition.y_n - groundProjection.y_n$$ *Figure 4.5*

The ground projection is found by firing a ray from the joint to the ground along a user defined up vector.

If both of these values fall below user defined thresholds for speed and height for a given joint, it is determined to be in contact with the ground. In order to smooth out the results and eliminate any anomalies, a median filter is run over the result of each joint with their neighbouring frames, and the joint is tagged for any frames that return true.

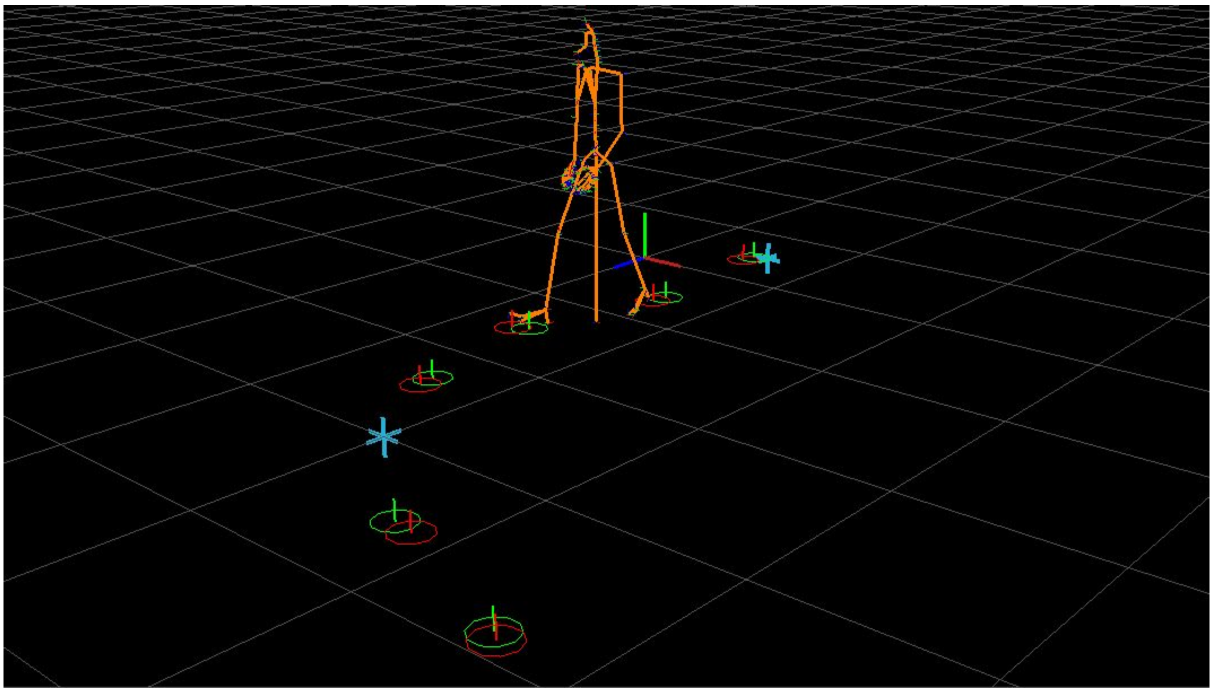


Figure 4.6 Footsteps - Green circles show when the heel is down, red are when it lifts up.

This technique can miss foot downs in noisy or irregular animations so a mechanism to manually tag foot downs is also provided. These tags are treated as constraints that keep a joint from moving while it is in contact with the ground.

4.1.3 Ankle Targets

Due to the way that ankle joints pivot when a locomotive animation is playing, it is necessary to treat them as a special case. For each frame of the animation, and for both ankles, a target transform must be found which ensures that the foot is above the ground when no joints are constrained. When joints are tagged as down, this transform must keep the foot from sliding, but allow for rotation as the foot rolls.

4.1.3.1 Foot Step Phases

Foot Up
When there are no foot down constraints, the foot is maintained at at least a minimum distance above the ground to ensure that the foot doesn't penetrate the terrain. This is done by projecting each foot joint position to the ground and then translating the ankle joint upwards by the distance from the joint to the ground.

Into Foot Down
When no joints are constrained, the animation is analysed to find the first upcoming constrained frame. If the constrained frame is within a user defined number of frames, the target ankle transform is found at the constrained frame. The joint transformations are then linearly interpolated from the current ankle target to the constrained target. As the ankle target at the constrained frame has been projected to the ground, this interpolation is necessary to smooth out any discontinuous change in the ankle transforms [Kovar et al. 2002].

Foot Down
To handle frames where there are constraints present, the solver needs information about which joints are constrained and at which frame did the constraint start. The first joint to contact the ground acts as the start point for calculating the current target transform. This start target transform is found by translating the ankle transform at the start frame by the projection distance to the ground

from the first constrained joint position, thus placing the joint on the ground. Subsequent frames take the start transform and rotate it around the projected joint position, until the joint lifts off the ground.

The next constrained joint can then be used as the new pivot point for the ankle. The rotation amount is determined by forming a plane with the joint positions of the foot, and finding the rotation from the start transform plane and the current plane. If a constrained joint is not touching the ground, its position is projected to the ground and an interpolated position is used towards the projection.

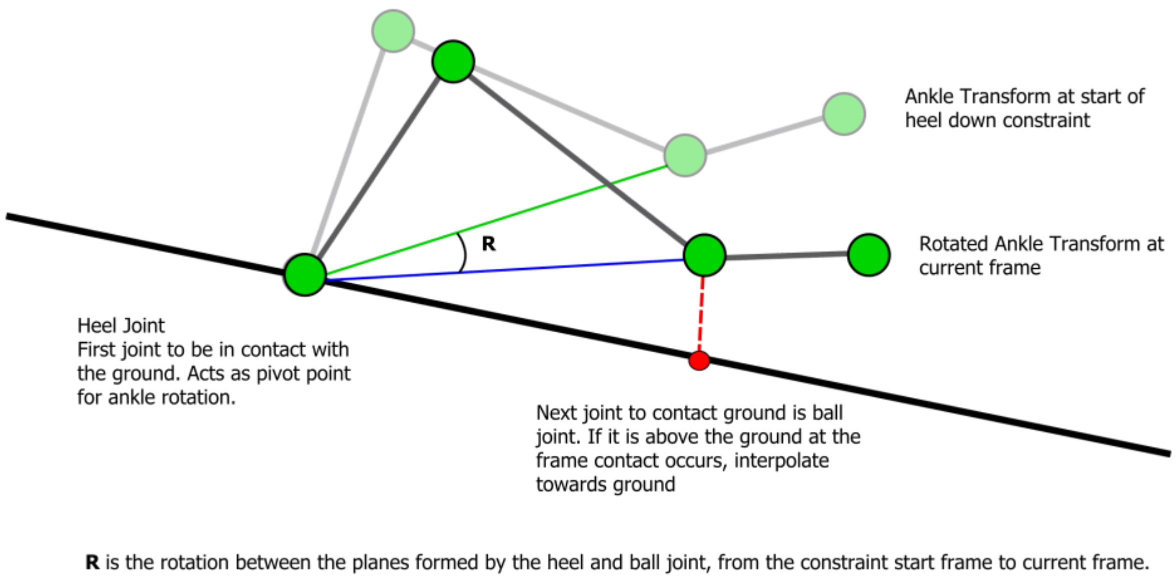

Ankle Transform at start of heel down constraint
R
Rotated Ankle Transform at current frame
Heel Joint
First joint to be in contact with the ground. Acts as pivot point for ankle rotation.
Next joint to contact ground is ball joint. If it is above the ground at the frame contact occurs, interpolate towards ground
R is the rotation between the planes formed by the heel and ball joint, from the constraint start frame to current frame.

Figure 4.7 Ankle Target Transform.

Out of Foot Down
As with the 'into' phase of the foot step, a linear blend is performed from the last constrained ankle target to the current target ankle transform over a user defined blend duration [Kovar et al. 2002].

4.1.4 Hip/Pelvis Target

In the DNEG crowd skeleton hierarchy a single joint, named "hip" is used as a root joint for the whole skeleton. Once the ankle targets have been determined for each leg the hip joint needs to be evaluated to see whether it needs to be translated. This is required if any of the targets are a greater distance away than the length of that leg.

The length of each leg is determined as the sum of the distances from each joint to its child, until the ankle joint is reached, starting from the root of that leg. These length results can be used as the radius of a sphere from which the target ankle position is centred. The intersection of these spheres provides positions where the hip is within range of all ankle targets. If the hip falls outside an ankle sphere its position is projected to the intersection in order to find the minimal translation needed to move the hip within range [Kovar et al. 2002].

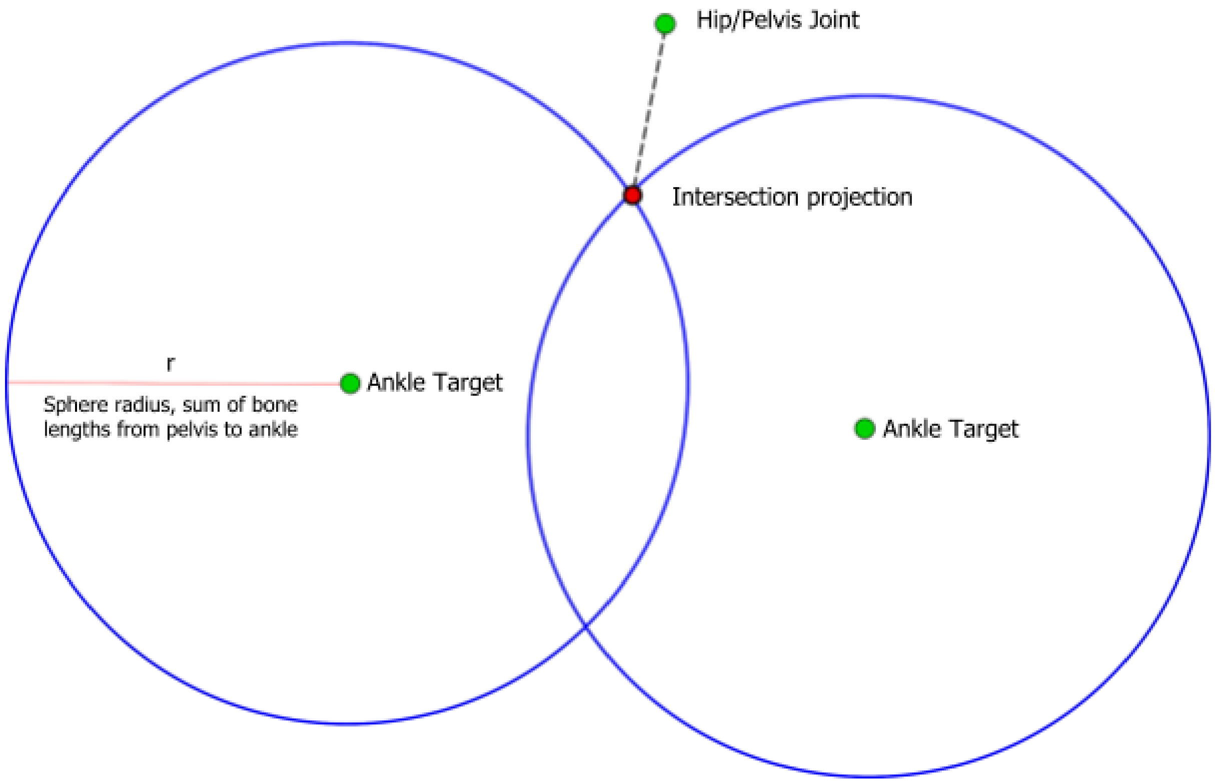


Figure 4.8 Hip/Pelvis Target.

4.1.5 Stretching Leg Bone

In order to avoid knee-popping and to allow for situations where there are large distances from the ankle targets to the hip [Kovar et al. 2002], the user has access to exposed parameters which allow leg lengthening and shortening. These parameters are utilised when calculating the leg lengths for the previous step. The maximum and minimum length of the legs are added as constraints for the IK solver, which handles the lengthening and shortening of the bones.

4.1.6 Applying IK

At this stage of the process the animation root has been projected to the ground, the target transforms for the ankles have been found and the hip joint has been translated to be within reach of the targets. The final process is to move the ankles to their desired targets. To do this, an IK solver is used in order to create a realistic pose given the new transforms of the ankles. Every joint in the chain of an ankle needs a new transform that will be within the constraints of a skeletal hierarchy.

The IK solve is the final step in the terrain adaptation, outputting a new pose with feet correctly placed on the terrain, whilst the body maintains a realistic pose that should closely resemble the input source pose.

4.2 IK

Manipulating skeletons and controlling the configuration of each joint is a well covered topic in animation, games and robotics. Inverse Kinematics allows us to move an end effector joint, and calculate consistent transforms for each joint in the chain of the end effector. The IK solve must be fast and produce smooth and realistic poses. Many techniques exist for performing IK, some of the more widely used in games and animation technology being CCD (Cyclic Coordinate Descent), the Jacobian Method, and FABRIK (Forward and Backward Reaching Inverse Kinematics) which was the chosen method implemented for our system.

4.2.1 FABRIK System

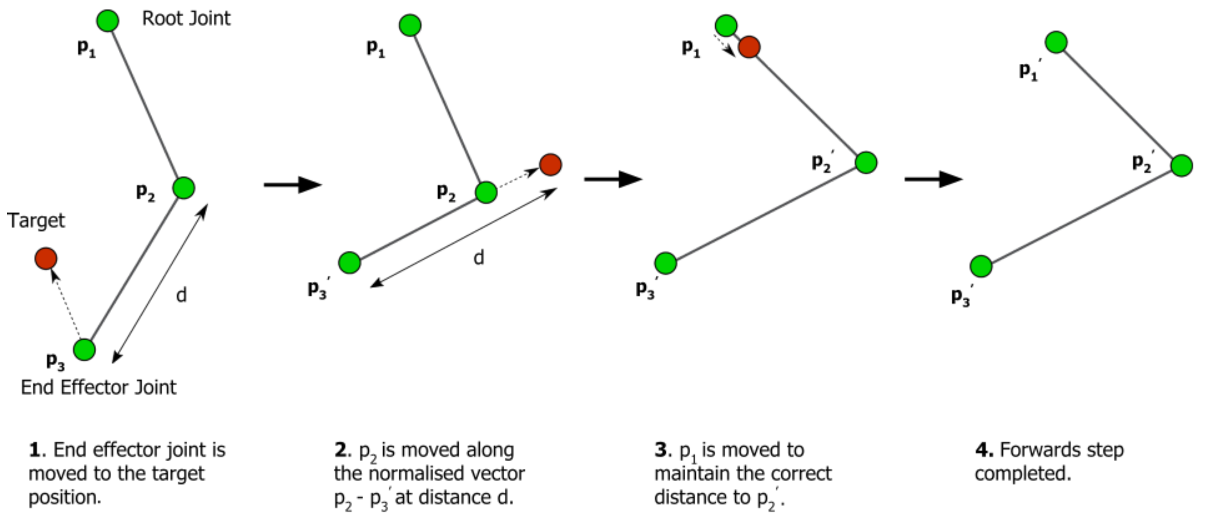
FABRIK is an iterative solution, fast enough to be used in real-time applications, relatively simple to implement, and can be used for full-body IK, where multiple end effectors can have a combined effect on shared parent joints.

With each iteration of the FABRIK algorithm the end effector joint should move closer to its desired target transform. When all end effectors are within a user-defined distance tolerance to their target, the algorithm is halted as a solution has been found.

4.2.1.1 Forwards and Backwards Solve

The algorithm gets its name by starting at the end effector, applying translations to each parent joint until reaching a root joint, then applying a second pass from the root to the end effector joints [Aristidou and Lasenby 2011].

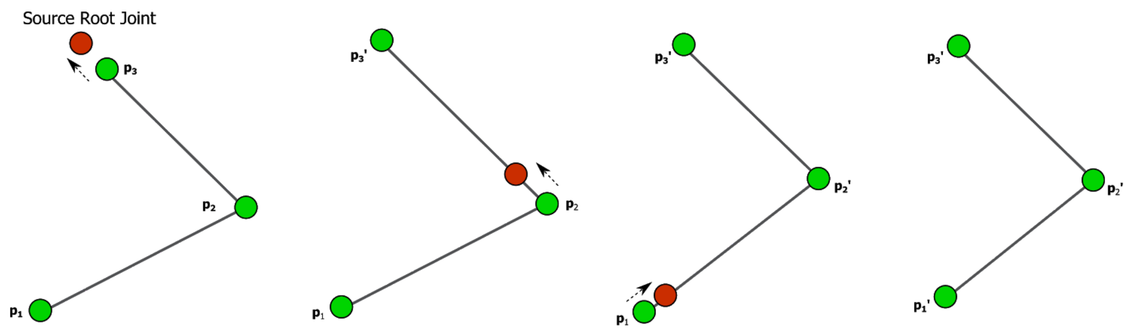
Forwards
First the end effector joint j_n's transform is set to the input target. Then, a position for the parent joint j_{n-1} is found along the vector $j_n' - j_{n-1}$ at a distance equal to the source distance, from j_n to its parent. This is repeated until the root joint is found.



Root Joint
p1
p2
Target
d
p3
End Effector Joint
1. End effector joint is moved to the target position.
p1
p2
p3'
d
2. p2 is moved along the normalised vector p2 - p3' at distance d.
p1
p2'
p3'
3. p1 is moved to maintain the correct distance to p2'.
p1'
p2'
p3'
4. Forwards step completed.

Figure 4.9 FABRIK forwards step.

Backwards
The now translated root joint is moved back to its source position and we continue to move each child joint along the vector from child to parent to maintain their source distance until the end effectors are reached.



1. The root joint p3 is moved back to its source position.

2. p2 is moved along the normalised vector p3' - p2, maintaining the distance of vector p3 - p2.

3. p1 moved to maintain the distance to p2'.

4. One iteration of the algorithm completed. We then repeat the forwards and backwards steps until the end effector joint is within a tolerable distance to its target.

Figure 4.10 FABRIK backwards step.

A maximum number of iterations is also defined as an additional exit condition if no solution has been found.

4.2.1.2 Multiple End Effectors

During the forward step, if a joint is reached that has more than one child joint, which we call a sub-base joint, every proposed transform of that joint is first found for each of the child joint chains it parents. An average of these transforms is then used as this sub-base joint's transform and the algorithm continues until reaching either another sub-base joint or the root joint of the skeleton [Aristidou and Lasenby 2011].

4.2.2 Constraints

The rotations of joints must be constrained in order to produce a realistic pose. Without them, joints may be bent the wrong way or twist beyond what is possible in a body. A joint has 3 degrees of freedom, which may be decomposed into two rotations, a swing and twist. The twist describes the rotation along the direction vector, i.e, where the joint is pointing. In our model, for a joint with only one child, the swing gives us the direction of the bone connecting the joint to its child. As the rotation can be decomposed into swing and twist components, each rotation is constrained separately [Aristidou and Lasenby 2011], where the swing of a joint is dependent on the result of the twist constraint.

The FABRIK algorithm provides the target positions of joints, but the joint rotations must also be calculated in order to correctly apply constraints. When a child joint is moved the rotation can be found using the vector from the parent to the previous child position, to the vector from the parent to the target child position. This rotation is then applied to the parent. In cases where a parent has more than one child, the average target position of all the children is found and the joint is rotated towards that point.

Although a skeleton may contain a number of different joint types, our implementation simply treats every joint as a ball and socket joint. Hinge joints such as the knee or elbow are treated no differently in the solver, but are constrained to only allow movement along a single plane.

4.2.2.1 Constraint Types

Three constraint types are used on each joint, applied at every step of the algorithm, on the forwards and backwards phase [Aristidou and Lasenby 2011].

Twist
The twist constraint for a joint is defined as two angles representing the minimum and maximum twist angle in which a joint rotation can lie in relation to its parent. When determining the twist rotation of a child joint in relation to its parent we first find its local rotation and decompose to find the twist along the axis we know points down the bone. The twist quaternion is then clamped to stay within the constraint angle range [van den Bergen 2016]. The difference in the current twist and the clamped twist produces the quaternion **R** needed to rotate the child joint in order to stay within its limits.

When carrying out the forward step of the algorithm the parent must be rotated so that the child adheres to the constraints, and on the backwards step the child is rotated.

Swing
The swing constraint of a joint is represented with 4 angles which are used to form 4 radii of an irregular ellipse. Given a target position for a child joint, we project this point onto the ellipse to ensure it stays within the limits of the constraint joint. The normal of the ellipse is the direction vector of the previous bone, and the distance from the constraint joint is the projection of the target child position onto the normal. Given the distance from the constraint joint, the ellipse radii can then be calculated. The twist/orientation of the ellipse is the same as that of the constraint joint.

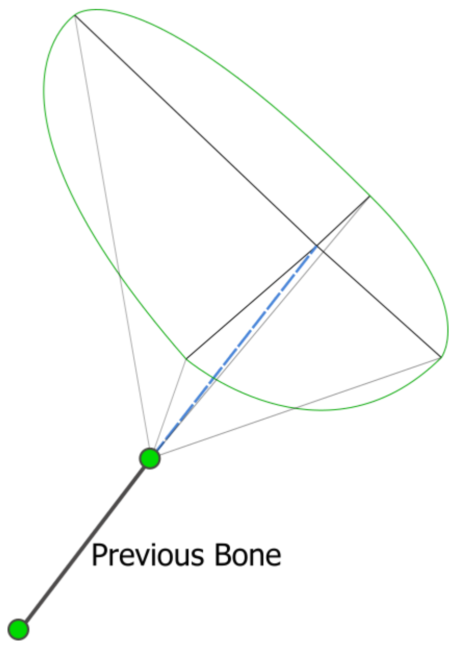


Figure 4.11 Swing Constraint - 4 angles of constraint form 4 radii for each axis of the ellipse.

By finding the position of the target local to the ellipse we can find the quadrant the target point lies in, giving us the two radii with which we can form a regular ellipse. By finding the closest point to that regular ellipse we have the constrained position of the target [Aristidou and Lasenby 2011].

Bone length
As mentioned previously in section 4.1.5 we allow for bone length stretching and squashing. This constraint is represented as a minimum and maximum scale range for a distance that a joint is allowed to be from its parent. This is applied during the FABRIK algorithm when finding new target positions for joints. The min/max bone lengths are calculated as the rest bone length scaled by the min/max values. The distance from the target to its parent is clamped to be within this constrained range.

4.2.3 Resolving Deadlock

An issue with the FABRIK algorithm is that deadlock can occur where the end effector joint of a constrained chain is outside the acceptable distance for a solution, and does not get any closer with subsequent iterations, even when a potential solution exists. Constraints are not aware of any other constraints in the chain which limits the potential solutions the solver can produce [Aristidou et al. 2016].

When a deadlock situation occurs, the solve restarts from the first iteration, but with a small rotation of the chain applied towards the target. This step is repeated until either a solution is found or we have rotated 360^o back to the start point.

Resolving deadlock can harm the performance of the solver due to the number of times the algorithm may need to be repeated. Because of this, the user is given control over the amount of rotation to perform after deadlock is found. A larger rotation can resolve deadlock faster at the risk of larger delta of joint positions from one frame to the next.

5 Pre-roll generation from deforming geometries

It is sometimes desirable to edit the animation of a deforming character geometry, either to fix artifacts or alter some aspect of the animation. Whilst it would technically be possible to edit each vertex of a polygon mesh individually, such a procedure would be incredibly cumbersome and time-consuming, and is therefore impractical for anything other than the simplest of edits. As described in previous chapters, the standard approach is to use a set of hierarchical joints to influence the motion of larger and more complex geometries using some skinning algorithm. The process of virtual bone placement and setting up the controlling relation between the bones and the polygon mesh is normally handled during the rig creation stage.

In a typical production environment, a rigger will add a skeleton and animation controls for each animated character, balancing high-fidelity deformation and performance. Whilst rigging does not necessarily need to be tied to specific functionality of a DCC, practically, making a DCC agnostic rigging system is time consuming and technically difficult, so many studios have built their rigging systems using DCC specific functionality. This is true of the DNEG character pipeline which uses Maya as its standard application for character animation. Cases where there are multiple studios working on a single film, and assets need to be shared, having a single rig representation that all companies can interpret is impractical, since each company will have developed their rigging system separately. As a result, when an animation is handed over from one company to another, it is usually cached as geometry and the rig is discarded.

5.1 The Pre-roll Problem

When bidding for work on a VFX show, a studio will split the bid into a number of shots, and the tasks required for each shot, where a shot is defined as a section between camera cuts. A sequence is normally defined as a series of shots that take place in the same environment or a succession of related actions. Whilst it might seem logical to animate characters as a continuous animation for the entire sequence, practically this rarely works as different camera positions might highlight animation artifacts that are obscured from another camera angle. As a result, in the vast majority of cases, animation is done at a per shot level.

This per-shot methodology often introduces an issue later down the pipeline since animation clips created for a shot will be bound by the first and last frame of that shot. This means that they contain no information of what occurred before the first frame. The event preceding shot start is commonly referred to as the *pre-roll* (Figure 5.1). The pre-roll is crucial to the realism and continuity of certain character effects such as cloth and hair simulation which fall to the creature effects department (CFX). For this reason, for shots where CFX work is required, animators are generally asked to add in a few frames of pre-roll animation. This often works, however, different CFX tasks need different types and lengths of pre-roll, and the back and forth between the animation and CFX departments in defining and producing this pre-roll animation can be time-consuming.

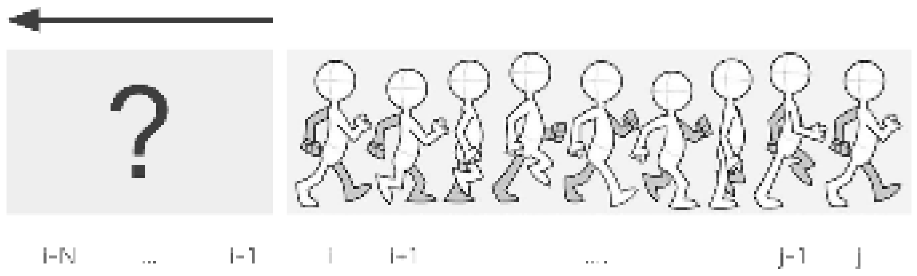


Figure 5.1 Given animation frames i to j, the N-frame pre-roll is the animation frame i-N to frame i-1.

The focus of this work is to allow the CFX artists to generate the pre-roll animation rather than relying on the animation department to provide it. There are two methodologies described in this chapter, the first, detailed in section 5.2 focuses on skeleton generation from the animated geometry. The idea being that if a skeleton can reliably be generated from the geometry, that skeleton can simply be posed and blended, driving the geometry to move in an expected way in a similar approach to that of the footstep cleanup mechanisms described in the previous chapter. The second approach uses shape interpolation techniques to move the vertex positions of the mesh directly, whilst attempting to ameliorate artifacts introduced with simple vertex blending.

5.2 Skeleton generation for pre-roll

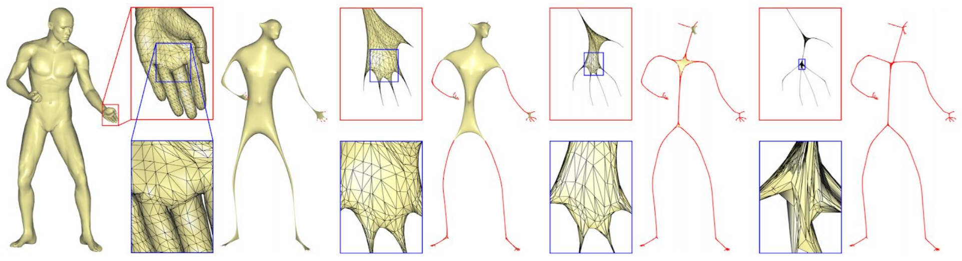


Figure 5.2 A skeleton from mesh contraction by Au et al. [2008].

Since manual rig creation can be laborious, there have been many different techniques attempting to automate the skeletal construction, such as thinning the geometry along the thickness until it converges to a line structure [Au et al. 2008] (Figure 5.2). Each method has pros and cons depending on the constraints of the problem. For our use case, we made the assumption that the tool would be predominantly deployed in scenes comprising realistic human characters, so we could utilise the conventional proportion in anatomy drawings and character designs to approximate the bone dimensions and locations (Figure 5.3, left). To maximize the usability of our tool, we based our skeleton structure on the Carnegie Mellon University's (CMU) standard widely used in motion-capture production and research (Figure 5.3, right). This skeleton is cross-compatible with the one used in crowd characters of Deliverable 6.8. The tool was created and used in the Houdini 18.0 software package. Figure 5.4 shows our test results of our skeleton generator which is fully automated.

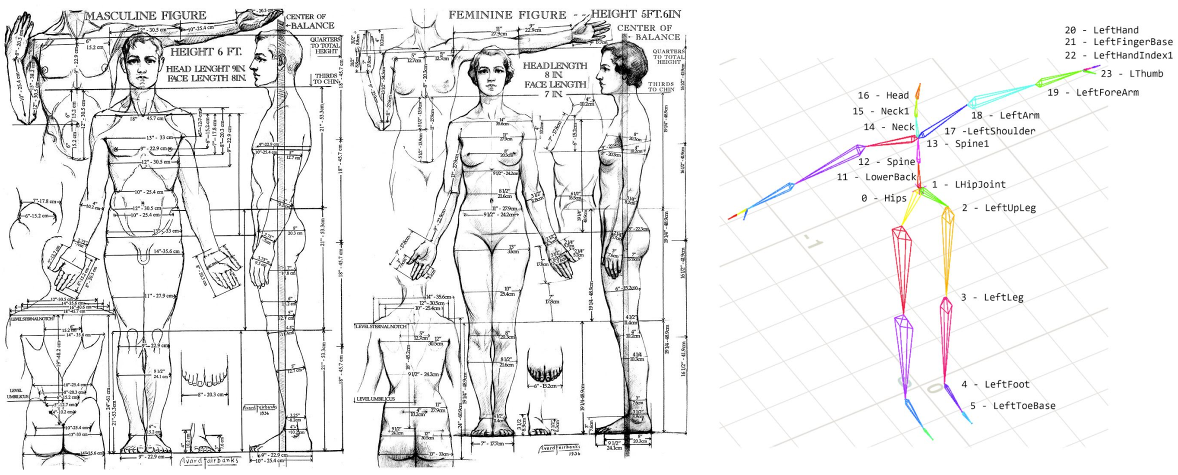


Figure 5.3 Left: conventional body proportions used in anatomy drawings & character designs [Fairbanks 2011]. Right: CMU's skeleton standard implemented in a Houdini crowd character.

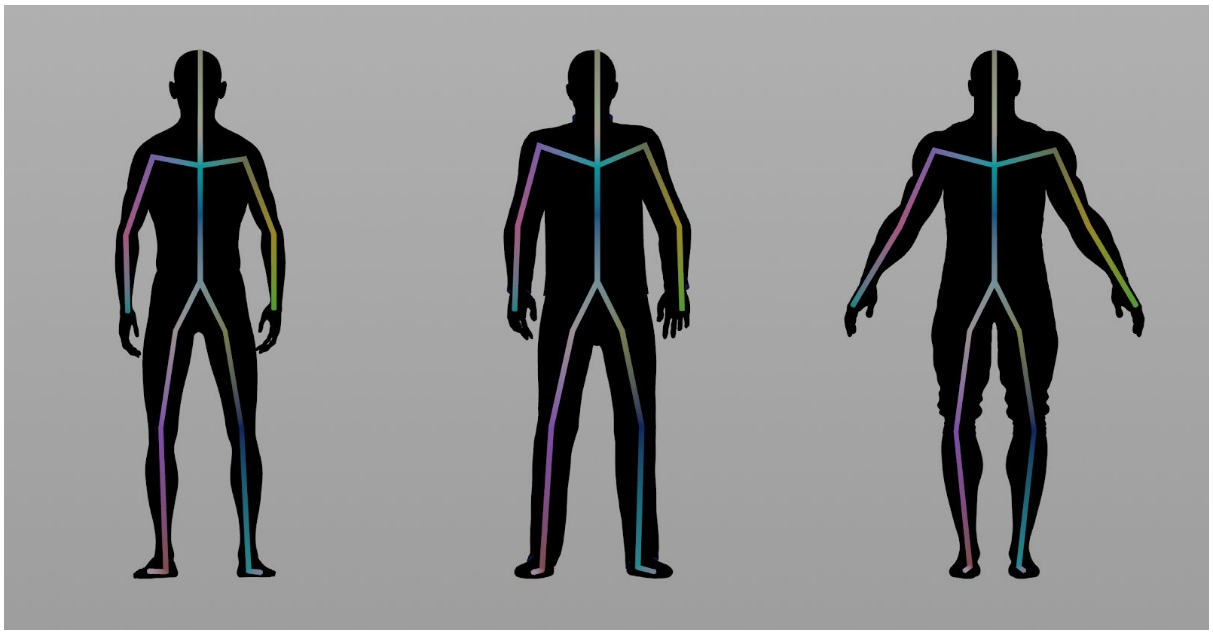


Figure 5.4 The results of our skeleton generators from left to right, DNEG's generic man, Houdini's crowd character, and Houdini's test male character (Tommy).

5.2.1 The skeletal construction process for a character in a rest pose

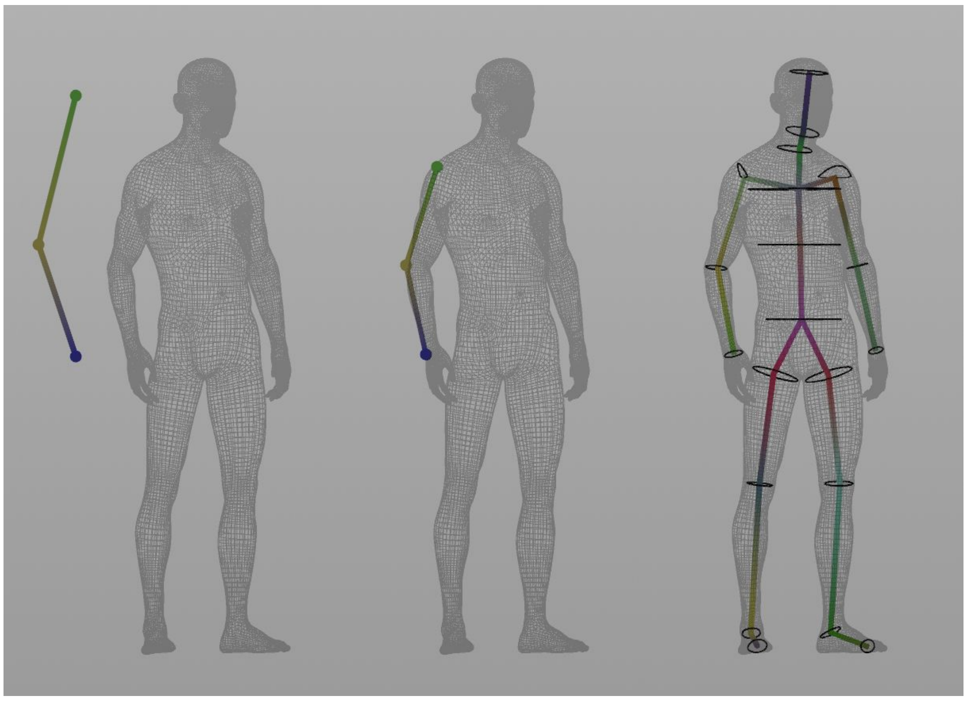


Figure 5.5 - Skeletal construction process of the right arm.

1. The process starts with estimating the positions of the bone points based on the human anatomy dimension. Note that the CMU's skeleton was simplified by omitting the finger joints.
2. Some bone points such as those along the head, neck and spine were placed inside the volume of the character, while others, such as shoulders, elbows and knees were offsetted by a small distance as shown in Figure 5.5. The bones with offsets were then projected to the character mesh either along a predefined direction or the closest distance. For bone locations that vary greatly in a rest pose such as the arms and lower legs, offsetting and projecting yield more accurate results.
3. Using the direction of the surface normal of the nearby polygons, the bones were moved inside the volume of the character. After that, at each bone point, a ring is projected outward in the direction orthogonal to the bone orientation to form a cross section of the character volume. The bone points were then moved to the centroids of the corresponding cross sections ensuring the skeleton was center-aligned throughout the character mesh.

As stated before, this approach only works with human anatomy in a rest pose and can be performed in a fully automated manner. We have tested the skeleton generator on three different human meshes and the results were robust as shown in Figure 5.4.

5.2.2 The skeletal construction process for a character in other poses

At the end of the skeletal construction of a rest pose, the cross section information is embedded into the rest mesh, and can be transferred to corresponding polygons on a new mesh. To generate the bone of the new mesh, we simply re-calculate the centroids of the transferred cross sections and connect them (Figure 5.6).

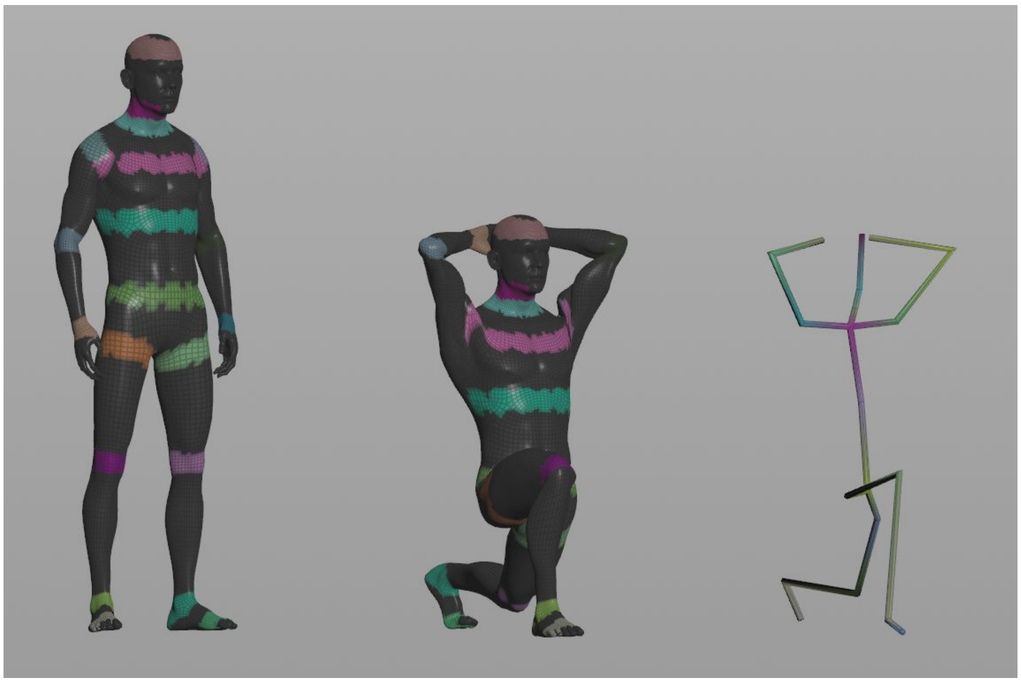


Figure 5.6 - The embedded skeletal construction information is transferred from the rest pose to a new pose and use to create the new skeleton.

5.2.3 The Pre-roll Creation with Auto-rig Workflow

The skeleton generator can be used to create a CMU's bone hierarchy of any given realistic human character. The resulting skeleton can then be used to deform the mesh by using the *biharmonic bone deformation* tool (the bones capture and deform the nearby polygons based on the mesh structure and proximity). This is a native tool available in the Houdini software. The combination of our skeleton generator and Houdini's existing mesh deformers establishes a simple but capable auto-rig system.

Without a rig, a simple way to create a pre-roll animation for a character would be to directly interpolate the mesh from a starting pose to the pose at the first frame. However, a naive linear mesh interpolation can create artifacts and undesirable deformations of the inbetweens. With our auto-rig system, artists are equipped with simple high level controls that can be used to resolve the issues quickly.

We have devised a toolset specifically for the pre-roll animation creation consisting of:

1. **Skeleton Rest** - generates a skeleton from a character mesh in a rest pose.

2. **Skeleton Pose** - generates a skeleton from a character mesh in any pose.

3. **Pose Interpolate** - generates key poses by interpolating between two given skeletons. Artists can specify the number of poses, duration (the number of frames), as well as visualize the result. These results are shown in Figure 5.7, top.

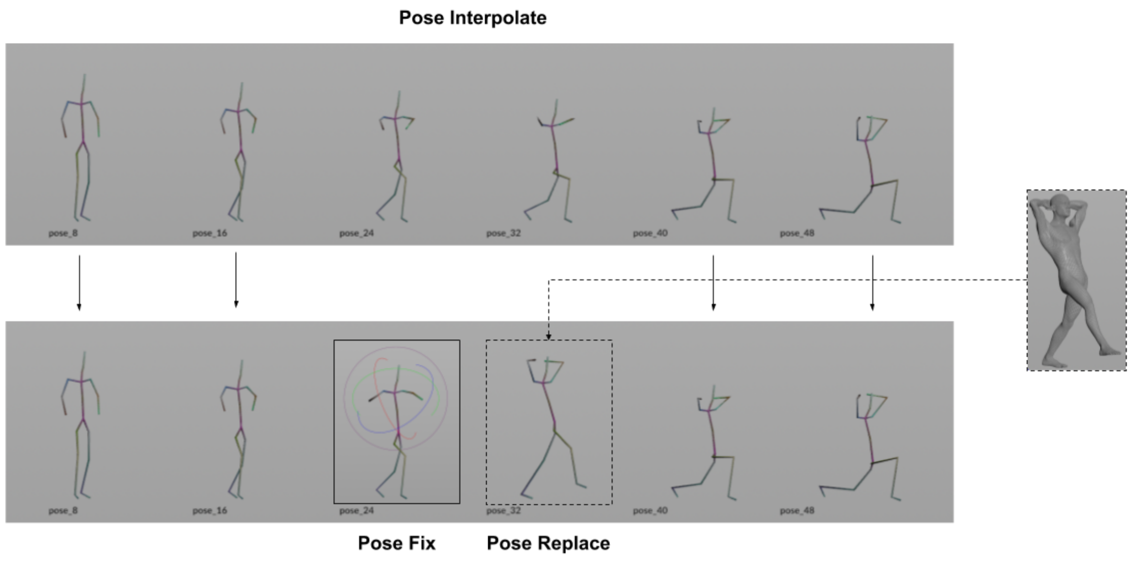


Figure 5.7 - Top: the visualization of "Pose Interpolate" tool interpolating the two poses from Figure 5.6 with 6 poses and the duration of 48 frames. Bottom: The changes made by the "Pose Replace" and "Pose Fix" tools.

4. **Pose Replace** - replaces a specific pose with another one that is generated from a different character mesh, shown in Figure 5.7, bottom.

5. **Pose Fix** and **Pose Fix Merge** - manually isolate, edit and merge back a specific pose. (Figure 5.7, bottom).

6. **Mesh Deform** - interpolates the key poses to create per-frame skeletons and use them to deform the character mesh for the pre-roll.

Figures 5.8 and 5.9 show the overview of the pre-roll workflow with the skeletal construction and the rest of the toolset, and its implementation inside Houdini software.

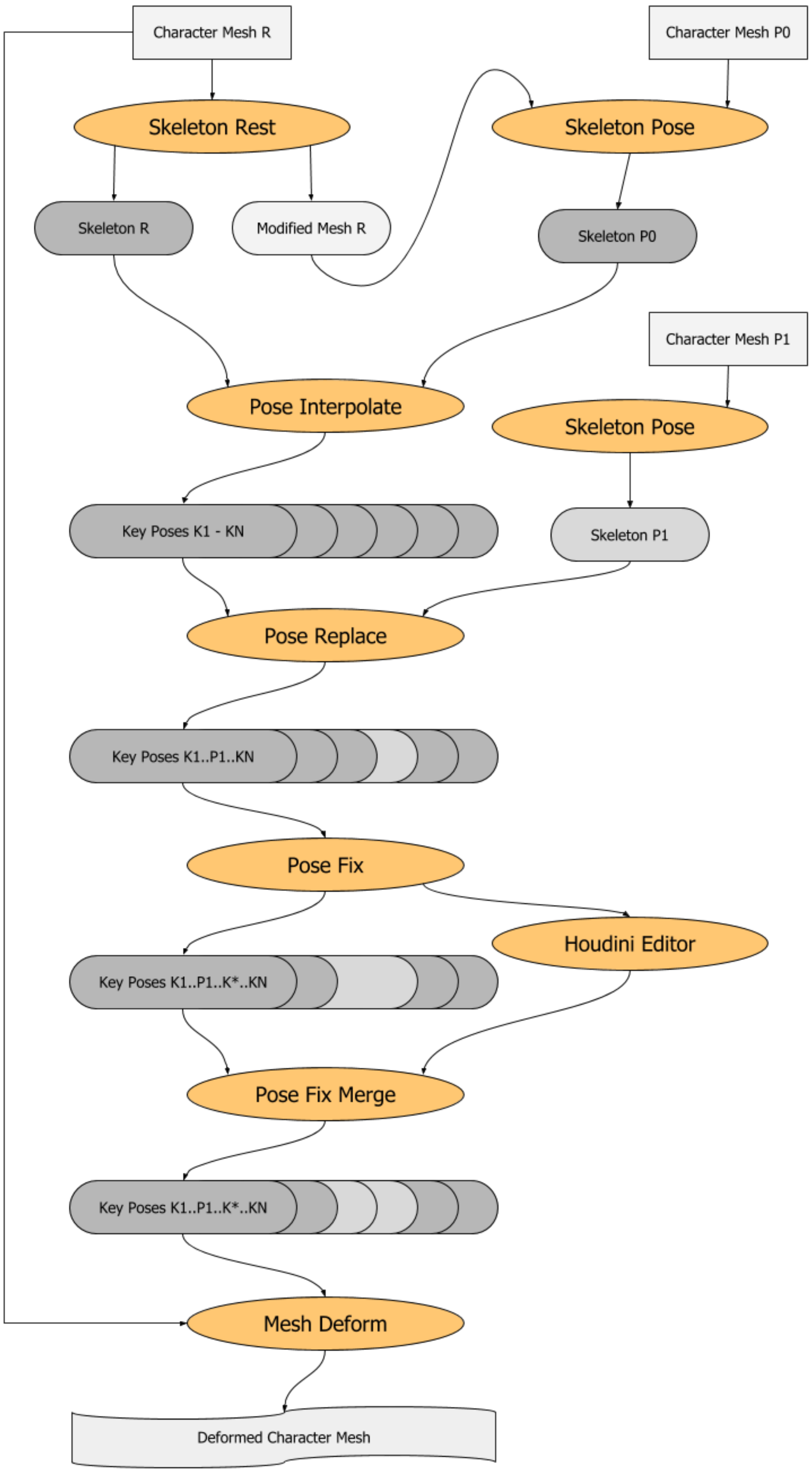


Figure 5.8 - The skeletal construction and the pre-roll workflow.

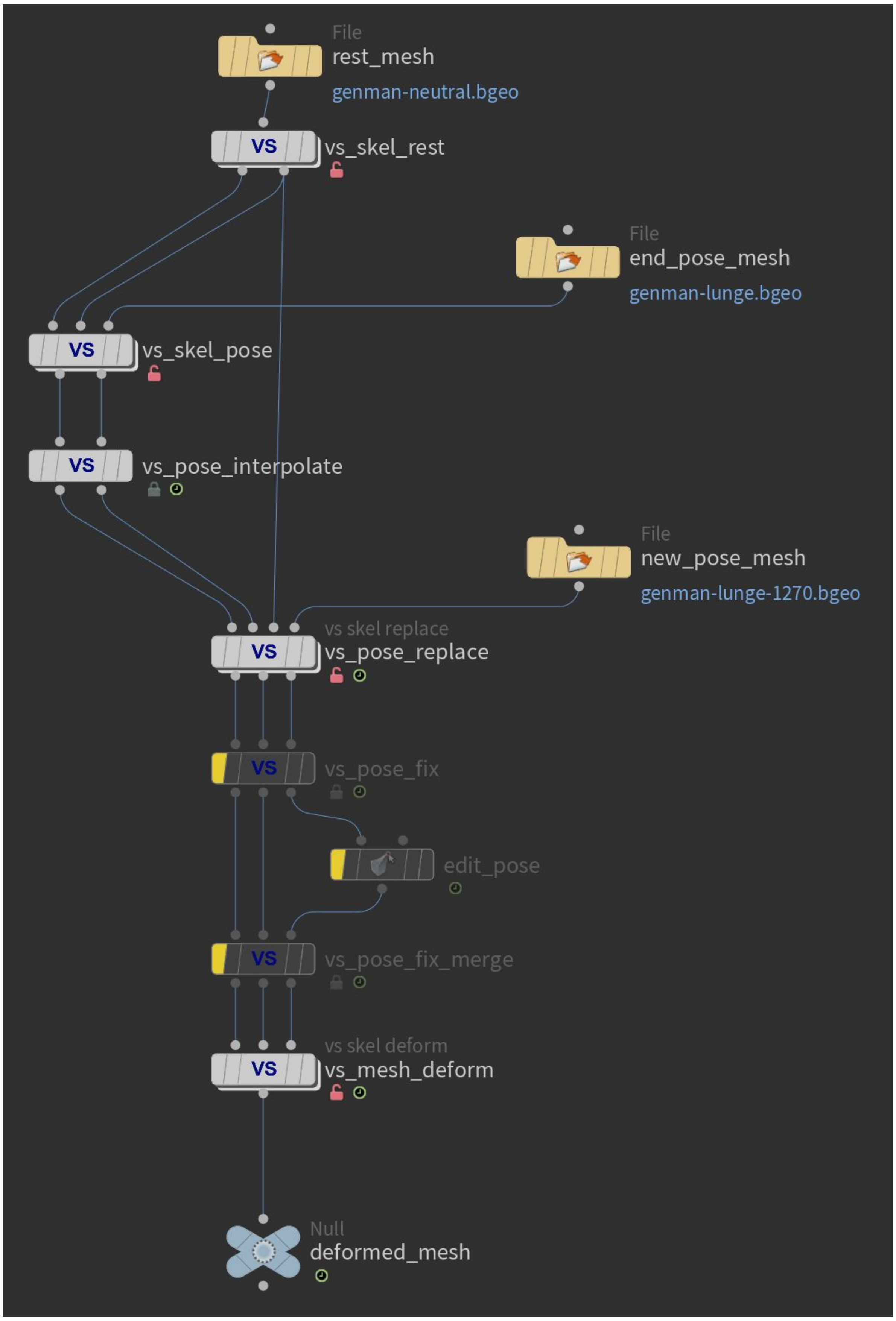


Figure 5.9 - The network tree of our toolset and pre-roll workflow implemented inside Houdini DCC software.

5.3 Shape interpolation based pre-roll tool

In this section, we consider the use of example-based deformation methods to address the pre-roll problem. For our purposes, "example-based" methods are those where a deformed shape is created by interpolating between given example shapes i.e. the deformed shape is a function of the example shapes. For clarity, we distinguish these from "handle-based" methods where a deformed shape is created by manipulating additional control objects (e.g. bones in a rig). In this case, the deformed shape is a function of the handles and a single neutral shape.

5.3.1 Deformation quality

With example-based deformation methods, the quality of deformation is largely dependent on the chosen interpolation scheme. Blendshape deformation, for example, is a particularly ubiquitous example-based deformation method whose popularity is largely due the simplicity of its interpolation scheme i.e. the deformed shape is a linear combination of example shapes.

While its simplicity ensures low computational overhead, blendshape deformation results in noticeable loss of surface area and volume when used to approximate rotational deformation (figure 5.10). In practice, these artefacts are often alleviated by introducing additional "corrective" example shapes which effectively provide a finer discretization of the desired interpolation path. This comes at a considerable cost, however, as modelling large quantities of corrective shapes is a time-consuming task that is typically carried out by skilled artists.

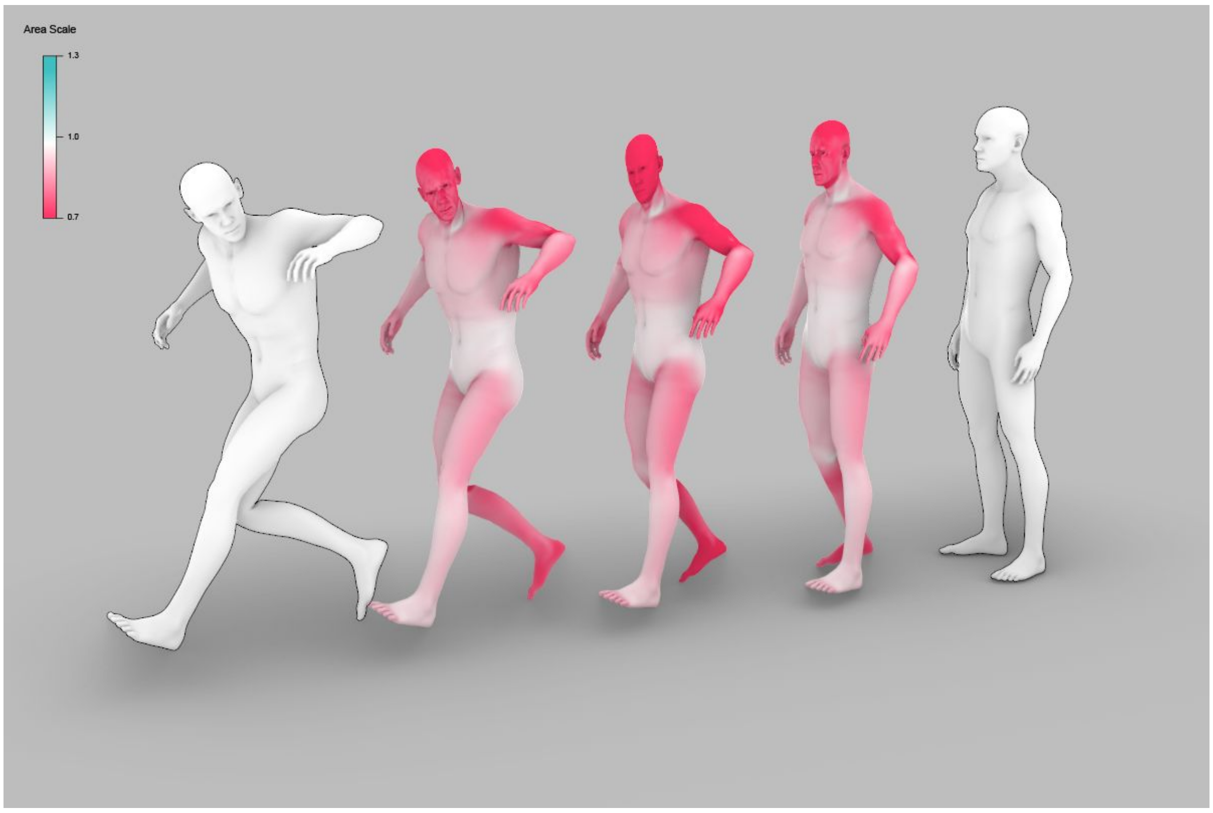


Figure 5.10 - Blendshape deformation results in loss of surface area at intermediate frames. Colour indicates relative deviation from linearly interpolated surface area. Example shapes are outlined.

Blendshape deformation offers a tradeoff between speed of evaluation and deformation quality that is widely considered to be acceptable in contexts where interactivity is critical and the performance bottleneck is deformation (e.g. animation).

This tradeoff is less acceptable in the context of the pre-roll problem however. Here, deformation quality is critical as simulated character effects (hair, cloth, etc.) are expected to physically interact with the deforming shape such that their motion appears natural on the first frame of animation. Any non-negligible loss of surface area and/or volume during deformation is, therefore, likely to result in unstable dynamics. Furthermore, correcting these deformation artefacts by creating more example shapes is no longer feasible here as doing so introduces an expensive feedback loop between CFX and animation departments.

Simulation of CFX elements is the clear performance bottleneck in the pre-roll context - the cost of which precludes interactivity in most real-world examples. Compared with animation contexts, this relaxes constraints on evaluation time for deformation, allowing us to consider more expensive interpolation schemes that offer better deformation quality.

5.3.2 Existing work

Existing example-based deformation methods can be broadly categorized into two types based on the nature of their interpolation schemes [Von-Tycowicz et al. 2015]. This section aims to highlight the advantages and disadvantages of each with respect to the pre-roll problem.

Both types follow the same general approach to the problem i.e. rather than interpolating vertex coordinates of example shapes directly (as per blendshape deformation), they interpolate other geometric quantities and then solve for the embedding that satisfies them which yields the interpolated shape. Where the two differ is in the types of quantities they interpolate which has a profound impact on the complexity of the solve step.

5.3.2.1 Intrinsic methods

Intrinsic methods are those which interpolate *intrinsic* quantities on the example shapes e.g. edge lengths and dihedral angles. While these methods are often quite robust in their ability to capture complex non-linear deformation, recovering the embedding of the interpolated shape from its intrinsic representation is typically a large non-convex optimization problem [Von-Tycowicz et al. 2015] which presents several problems in the pre-roll context.

Firstly, non-convexity implies the existence of multiple local minima which may cause discontinuities in the animation as the solution (i.e. the interpolated shape) jumps from one local minimum to another between frames. Secondly, numerical methods for solving non-convex optimization problems are prohibitively expensive for our purposes.

5.3.2.2 Variational methods

Variational methods are those which interpolate differential quantities on the example shapes. In particular, these methods focus on differential quantities that can be evaluated via linear operators (e.g. the gradient or the Laplacian) as they allow the embedding to be recovered by solving a convex optimization problem with a quadratic positive semi-definite objective [Botsch et al. 2007]. In practice, minimizing such an objective amounts to solving a sparse linear system - a procedure for which mature efficient solvers are readily available. This convexity is particularly important in the context of the pre-roll problem as it implies the existence of a single global minimum, guaranteeing continuity of the animation between frames.

While the use of linear differential operators greatly simplifies the solve step, it does so at the cost of deformation quality. In general, these methods struggle to capture complex non-linear deformation, making them unsuitable for certain applications [Botsch et al. 2007]. Despite this, variational methods offer a significantly less dramatic speed-quality tradeoff than blendshape deformation, making them worth further consideration in the context of pre-roll.

5.3.3 Poisson shape interpolation

Poisson shape interpolation [Xu et al. 2006] is a variational method used to interpolate between a source and target shape in a way that captures rotational deformation. As a byproduct, it also approximately preserves surface area and volume during interpolation making it particularly well suited to the pre-roll problem (figure 5.11).

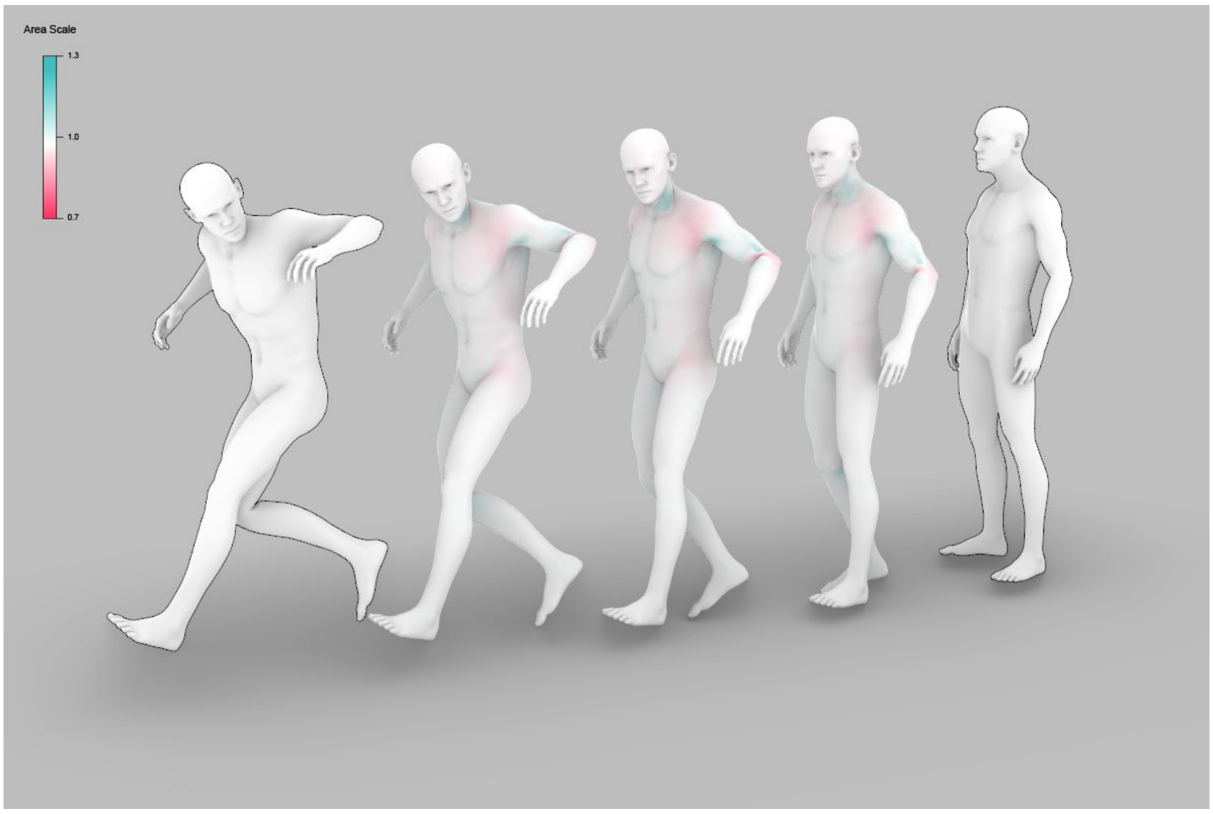


Figure 5.11 - Poisson shape interpolation roughly preserves surface area at intermediate frames. Color indicates relative deviation from linearly interpolated surface area. Example shapes are outlined.

The method is able to capture rotational deformation by performing a polar decomposition of the deformation for each face, yielding its rigid (rotational) and non-rigid (scale/shear) components. These are interpolated separately for each pair of corresponding faces using different interpolation schemes (i.e. spherical linear for rigid and linear for non-rigid) and then recombined to get the interpolated deformation (figure 5.12). This avoids the familiar "candy-wrapper" effect (i.e. volume loss) caused by linearly interpolating deformations directly.

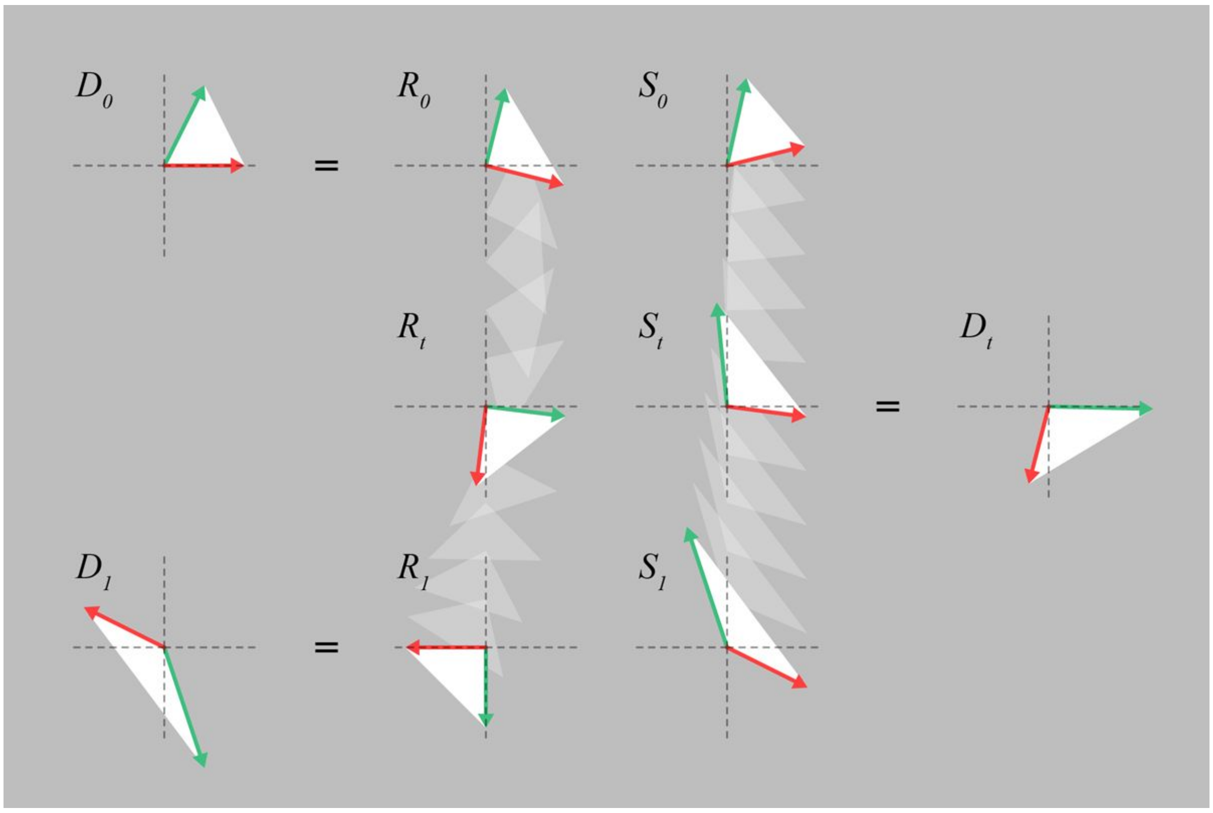


Figure 5.12 - Rigid (R_t) and non-rigid components (S_t) of deformation are interpolated separately for each pair of corresponding triangles.

Given the interpolated deformation (D_t) for each pair of corresponding faces, the rest of the method proceeds as follows:

- Evaluate the gradient of the vertex coordinates within each deformed face

$$\nabla x' = \nabla(D_t x)$$

- Solve a Poisson equation to get the vertex coordinates of the interpolated embedding (x')

$$\begin{aligned} \Delta x' &= \nabla \cdot \nabla x' \\ \Delta x' &= \nabla \cdot \nabla(D_t x) \\ x' &= \Delta^{-1}(\nabla \cdot \nabla(D_t x)) \end{aligned}$$

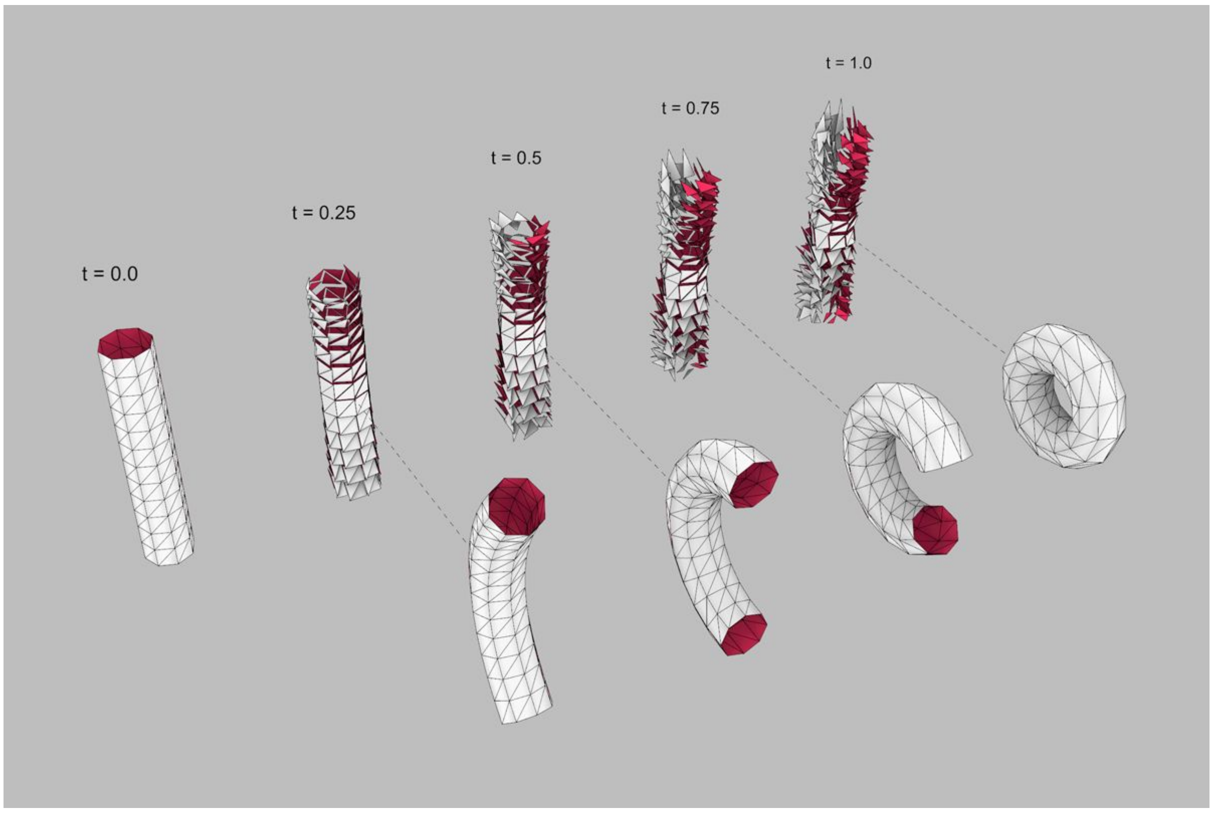


Figure 5.13 - Deformation of corresponding faces in the source and target shapes are interpolated independently. A sparse Poisson problem is then solved to recover the vertex coordinates of the interpolated shape.

5.3.3.1 Issues

Poisson shape interpolation tends to fail when the given source and target shapes have large rotational distances between corresponding faces, resulting in inconsistent interpolation paths (figure 5.14). This is a natural consequence of interpolating the rotation of each face independently since there is nothing preventing adjacent faces from taking diverging paths through SO^3. This is more likely to occur as the rotational distance between source and target faces approaches π.

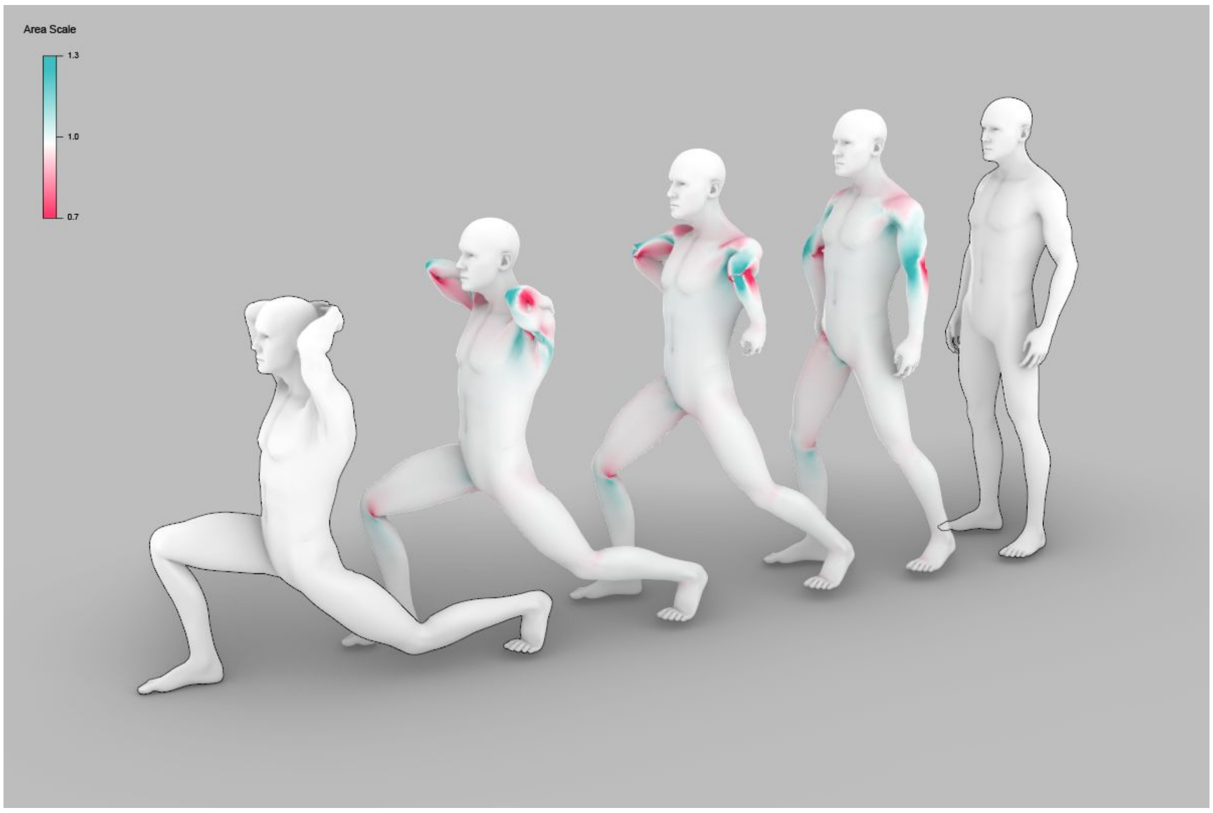


Figure 5.14 - Large rotations between corresponding faces in example shapes causes inconsistent interpolation paths. Color indicates relative deviation from linearly interpolated surface area. Example shapes are outlined.

5.3.4 Implementation

The limitations of Poisson shape interpolation discussed above preclude direct application in the context of the pre-roll problem as large rotations between the neutral shape and the first frame of animation are entirely likely. To address this fundamental issue, we implement a modified version of Poisson shape interpolation that provides users with a means of resolving interpolation artefacts if and when they arise. These modifications also provide some additional utility for addressing other challenges related to pre-roll generation.

5.3.4.1 Pre-transformation

The most direct way of resolving inconsistent interpolation paths caused by large rotations is by simply pre-transforming the source shape such that it better aligns with the target. Finding the optimal transformation (i.e. the one that minimizes the rotations between all pairs of corresponding faces) is non-trivial however it can be reasonably well approximated by solving a standard orthogonal Procrustes problem using covariance matrices of the source and target shapes.

Our implementation also allows for this transformation to be manually specified which was found to be more intuitive in cases where interpolation artefacts were quite local and therefore easy to correct interactively. While we found that pre-transformation resolved the majority of problematic test cases from production, it is not a general solution as the best alignment may still result in large rotations between corresponding faces.

5.3.4.2 Intermediate shapes

A more robust approach to resolving inconsistent interpolation paths is through the use of intermediate example shapes. These allow the user to better define the interpolation path taken from source to target, providing an intuitive and directable means of fixing deformation artefacts as they become apparent (figure 5.15).

Unlike with blendshape deformation, the number of intermediate shapes required here is guaranteed to be small, eliminating much of the modelling burden typically associated with corrective approaches. This guarantee is simply due to the fact that the distance between any two rotations in SO^3 is at most π. By introducing an intermediate example shape, the rotational distance between corresponding faces is potentially halved, at which point paths through SO^3 can no longer diverge. This means that in all but the most pathological cases, a single intermediate shape is enough to correct any inconsistencies.

Beyond providing a means of resolving deformation artefacts, intermediate shapes also allow users to avoid self-intersection during interpolation. This is particularly important in the context of pre-roll as self-intersection will typically lead to numerical instability in simulated CFX elements.

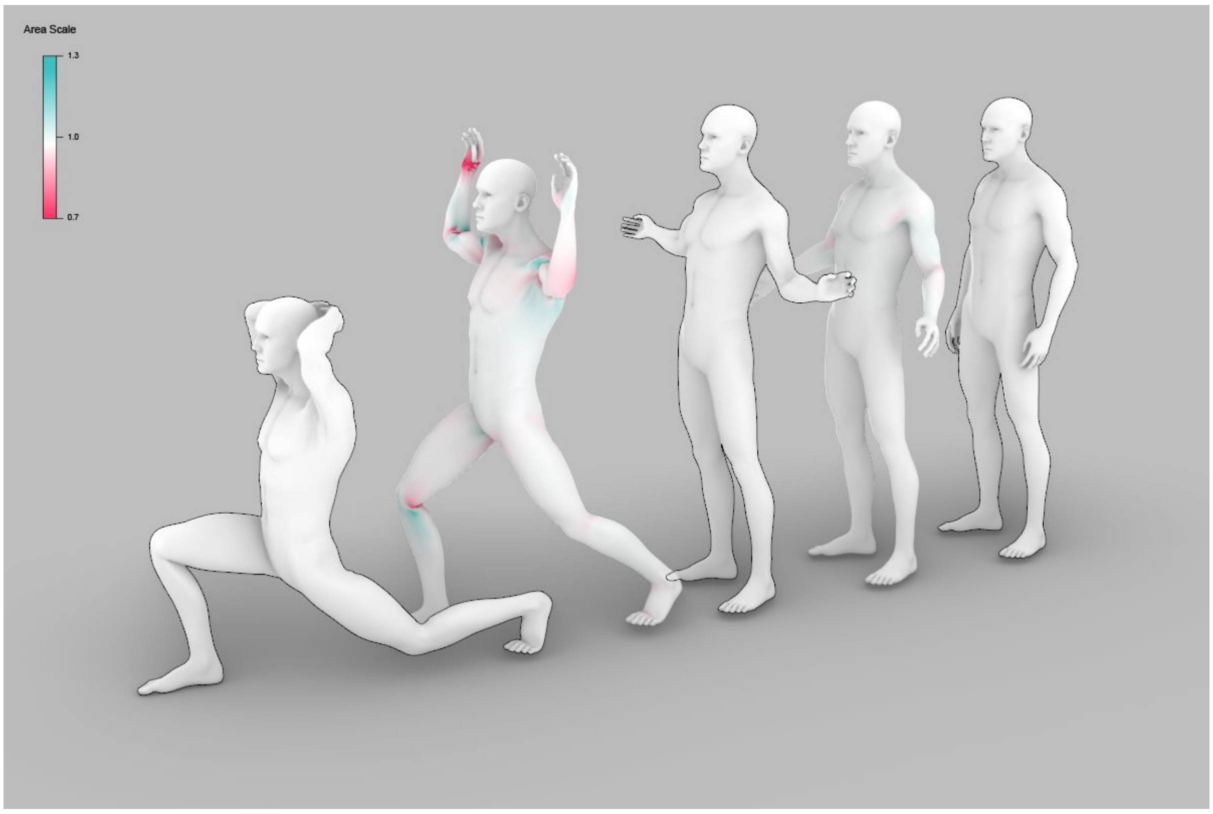


Figure 5.15 - Adding a single intermediate example shape corrects the interpolation path. Color indicates relative deviation from linearly interpolated surface area. Example shapes are outlined.

6 Conclusion

Whilst the use cases and technological implementations described in this deliverable are varied, the common theme is one of animation re-use. A large amount of time is spent in the creation of bespoke animations for multiple situations, and yet more time is spent moving backwards and forwards between departments when requirements are unclear. The work done in chapters 4 and 5 helps to move the job of custom asset creation and adaptation into the department that will end up using it which adds clarity to the requirements.

For footstep cleanup and terrain adaptation, the IK system used (FABRIK) produced realistic poses at runtime speeds, suitable for use on many crowd agents. The constraint rig used with the IK system is not as complex as the rigs used to author animations and therefore not suitable for providing high-quality hero animations, but for crowds the results are promising. An advantage of using the FABRIK algorithm is its implementation of full-body IK, which may be useful in the future in areas beyond terrain adaptation. Retargeting animation for different skeletons, interaction with other agents, environments or props and applying physical reactions to agents are all achievable with this IK system.

The results of the terrain adaptation are convincing while leaving areas to improve upon. Artists will be able to re-use many locomotion animations to traverse quite extreme terrain including stairs and maintain control over the output animation. Closer scrutiny of the adapted animation does show an unrealistic shifting of weight, especially over large changes in terrain height. A future improvement would be to add counter-balancing, where the IK system may be used to keep the centre of mass of the character over its support points (feet in contact with the ground). This would give us poses where the hips are shifted forward over a leading foot as the character steps up onto higher terrain, and when traversing down sloped terrain the character will lean back.

The two methods for generating pre-roll animation presented in chapter 5 provide complimentary means of circumventing a time-consuming feedback loop that currently exists between animation and CFX departments. Both of the proposed workflows shift the authoring of pre-roll sequences from animation to CFX, allowing CFX artists to create sequences that are better tailored to specific simulation requirements which animators may not typically be aware of.

The auto skeleton generator has been tested with three human character meshes, DNEG's generic man, Houdini's crowd character and Houdini's test male character (Tommy), with varying topology complexity and rest poses, which yielded robust results (Figure 5.4). Although the current tool cannot handle a T-pose and other skeleton standards, such as the one used at DNEG, generalisation should be achievable with a few minor tweaks. One avenue of future research would be to evaluate whether the tool can be improved to reliably construct skeletons of stylized bipeds and quadrupeds with machine learning and enough training data. In terms of mesh deformation, our method does not take into account the bone rotations and undesirable twists may occur in the interpolated meshes. To address these issues, aim directions for the elbows and knees, as well as angle limiters can be added to the workflow in the future.

Generating pre-roll animation via shape interpolation worked reasonably well in most production test cases. For those where it failed due to large rotations between corresponding faces in source and target shapes, the interpolation path could usually be corrected by simply pre-transforming the source shape to better align with the target. For the few cases that could not be resolved in this way, the interpolation path was corrected by adding a single intermediate example shape. The creation of intermediate shapes remains a challenge that we intend to address in future work. Given that the proposed workflow is intended to be carried out by CFX artists rather than animators, providing accessible tooling for doing so will be essential for use in production.

7 References

Alexa, Marc, Daniel Cohen-Or, and David Levin. "As-rigid-as-possible shape interpolation." *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000.

Aristidou, Andreas, and Joan Lasenby. "FABRIK: A fast, iterative solver for the Inverse Kinematics problem." *Graphical Models* 73.5 (2011): 243-260.

Aristidou, A, Chrysanthou, Y, and Lasenby, J. 2016. *Extending FABRIK with model constraints.* Comp. Anim. Virtual Worlds, 27: 35– 57. doi: 10.1002/cav.1630.

Au, Oscar Kin-Chung, et al. "Skeleton extraction by mesh contraction." *ACM transactions on graphics (TOG)* 27.3 (2008): 1-10.

Botsch, Mario, and Olga Sorkine. "On linear variational surface deformation methods." *IEEE transactions on visualization and computer graphics* 14.1 (2007): 213-230.

Fairbanks, Eugene F. *Human Proportions for Artists*. Bellingham, WA: Fairbanks Art and Books, 2011.

Fröhlich, Stefan, and Mario Botsch. "Example-driven deformations based on discrete shells." *Computer graphics forum*. Vol. 30. No. 8. Oxford, UK: Blackwell Publishing Ltd, 2011.

Kovar, Lucas, John Schreiner, and Michael Gleicher. "Footskate cleanup for motion capture editing." *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 2002.

Pražák, Martin. *Locomotion for Crowd Animation*. Diss. Trinity College, 2012.

Sumner, Robert W., et al. "Mesh-based inverse kinematics." *ACM transactions on graphics (TOG)* 24.3 (2005): 488-495.

van den Bergen, Gino. "Rotational Joint Limits in Quaternion Space." *Game Engine Gems* 3 (2016).

Von-Tycowicz, Christoph, et al. "Real-time nonlinear shape interpolation." *ACM Transactions on Graphics (TOG)* 34.3 (2015): 1-10.

Xu, Dong, et al. "Poisson shape interpolation." *Graphical models* 68.3 (2006): 268-281.

8 Web references

Clavet, Simon. *Motion Matching And The Road To Next-Gen Animation*. 2016, https://www.gdcvault.com/play/1023280/Motion-Matching-and-The-Road. Accessed 24 June 2020.

9 Acronyms and abbreviations

- CFX - Creature Effects
- CMU - Carnegie Mellon University
- DCC - Digital Content Creation
- FABRIK - Forward and Backward Reaching Inverse Kinematics
- IK - Inverse Kinematics