A System for Retargeting of Facial Motion Captured Performance for Bone-deformable Models

by

Derek Foley, BA(Mod) Computer Science

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science

University of Dublin, Trinity College

August 2012

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Derek Foley

August 29, 2012

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Derek Foley

August 29, 2012

Acknowledgments

I would like to acknowledge my supervisor, Rachel McDonnell for her help and guidance as well as for introducing me to the dissertation topic. I would also like to acknowledge the GV2 lab of Trinity College, Dublin for providing me with data and assets for use in the dissertation.

Derek Foley

University of Dublin, Trinity College
August 2012

A System for Retargeting of Facial Motion Captured Performance for Bone-deformable Models

Derek Foley
University of Dublin, Trinity College, 2012

Supervisor: Rachel McDonnell

We present a solution for retargeting facial motion capture for bone-deformable 3D models. Currently bone-deformable models are used in the film and video game industries. The retargeting process however, is often performed manually and can be time consuming. We were interested in attempting to design an application that could add automation to the process as well as cater for arbitrary motion capture data and 3D models. There was not much available information on other systems for bone-deformable retargeting. Much of the work to date involves expression matching using blend-shapes, which are not applicable to bone-deformable models. Therefore we experimented with different approaches. We first experimented with "displacement from rest". This approach involves determining the displacement of a marker from its rest position and applying the same translation to a corresponding target bone. "Displacement from rest" however results in unwanted behaviour, presenting issues of mesh

interpenetration and incomplete facial feature movement.

We present range-of-motion matching as a viable approach to retargeting. Range-of-motion matching involves: determining the range-of-motion of a performance, specifying equivalent range-of-motion for a target model and matching percentage movement between the source and target ranges-of-motion.

The implemented system presents an approach to retargeting, respecting differences in proportion and scale of the target compared to the source and avoiding mesh interpenetration and incomplete movement. Unlike other retargeting approaches, it does not require a virtual representation of the source actor; retargeting can be preformed directly from the motion capture data. Additionally the system is easy to use and does not require much training. Finally it adds automation to a normally manual bone-deformable retargeting process.

Contents

List of Figures

Chapter 1

Introduction

This dissertation is concerned with facial motion retargeting from motion capture data. Motion capture is a method of recording an actor's performance for later use, potentially for animating a 3D model. Motion retargeting is taking an actor's performance data and adjusting it for use with a differently proportioned 3D model.

This chapter will introduce the goal of the dissertation (section 1.1) and the motivation for undertaking the dissertation topic (section 1.2). It will conclude with an overview of the chapters, summarising their contents (section 1.3).

1.1 Goal

The goal of this dissertation was to provide a solution for transferring facial performances (specifically from motion capture) to an arbitrary 3D model, using bone based deformations. This solution would be provided as a C++ application taking facial motion capture data and a 3D model and, successfully return an animation driven by the motion capture data, adapted for the 3D model.

1.2 Motivation

Currently, motion capture is used heavily in the video game and film industries. Many of the facial models used are bone-driven (bone-deformable). However, the retargeting

process is often done manually using the source performance as a reference rather than as a driver of the target model. This can allow for the animator to impart a certain style to the animation, but it is also time consuming and requires the attention of an experienced, highly skilled animation artist.

We were interested in investigating automation for this retargeting process and additionally, to design a solution that could be used by a less experienced user. It was felt an automated approach would be useful for handling a large set of diverse actors as well as freeing up the animator for other work.

1.3 Dissertation Guide

- **Chapter 1: Introduction**
  The current chapter. This chapter introduces the dissertation, discusses the motivations for undertaking the project and gives a reading guide for the remainder of this document.

- **Chapter 2: Background**
  Gives an overview of motion capture and retargeting. Discusses the work done to date on facial retargeting and provides some discussion on possible adaptations for this dissertation.

- **Chapter 3: Design**
  States the goal of the built application as well as detailing requirements and limitations of the design. It also discusses the methodology for the design decisions and provides a listing of the tools and assets employed.

- **Chapter 4: Implementation**
  This chapter details the stages of the retargeting application. It discusses the steps necessary to prepare the marker data and 3D models, the profiling of the marker data, the mapping of the markers to the model bones and retargeting of the marker movement to the target model.

- **Chapter 5: Results**
  Discusses the results of three approaches to facial retargeting: direct mapping,

displacement from rest and range-of-motion matching. This chapter provides images of each approach as they appear in the retargeting application.

- **Chapter 6: Conclusion**
  Restates the goal of this dissertation and discusses how the implementation addresses them. This chapter also discusses limitations in the final application and addresses issues that arose. It also provides a section on possible extensions to the application including catering for stylised characters.

- **Appendix A: TRC File**
  A sample .trc file, used in motion capture. This sample has been truncated for brevity as full .trc files are typically large.

- **Appendix B: Mapping File**
  An example of a marker-to-bone mapping file. It shows the format designed for this application. The file also contains example bone range-of-motion specifications.

- **Appendix C: Performance Range-of-Motion**
  This file details a range-of-motion performance for a motion capture actor.

Chapter 2

Background

The goal of this dissertation was to investigate a solution for adapting the facial animations of an actor to a non-representative (not a virtual replica of the actor) bone-driven 3D model. This chapter will discuss work done to date and to what extent this work can be adapted for use in the dissertation. Currently, the main body of research on facial retargeting uses blend-shape models, which can require a skilled animator to construct and could make them less suited to a bone-driven animation style. We will discuss aspects of the blend-shape approach, as well as other approaches to facial retargeting, while also looking at retargeting techniques of whole body skeletons which may fit better with bone-driven models. First, however, we will discuss some different methods of human motion capture.

2.1 Human Motion Capture

2.1.1 Active Sensors

Marker based human motion capture techniques apply markers or sensors to key features of the body. These techniques can use a variety of different marker types [27]. Markers can be both active and passive. Active markers, or sensors, send and receive signals from a base system relaying information such as position, orientation and speed of movement. Mechanical sensors are triggered when the actor bends a limb, sending out a signal which reflects the configuration of the body part it is attached to. Accelerometers can measure the acceleration of the attached body part. Electromagnetic

sensors can relay information about position and orientation with regards to some magnetic field generated by a transmitter. Acoustic sensors operate in a similar manner to sonar, relaying positional information on receipt of an audio signal. Finally, optic fibre sensors are similar to mechanical sensors but can be noted separately due to the fact that they are substantially less cumbersome.

While recent instances of these sensors can be quite fast and accurate, they require the actor to be wired up in many cases making any complicated motions difficult to capture.

2.1.2 Passive markers

The idea with passive markers is to capture human motion using information already present in the scene (visible light, electromagnetic distortion) without the need to produce external signals. The difficulty arises in using a camera to convert a 3D scene into a 2D image. The 2D image will present a lot of information and can lose information concerning depth. Therefore markers are attached to the body to reduce the amount of information to analyse. More recently, motion capture is done using reflective balls attached to the body at key areas and a number of infrared cameras capture an actor's performance at different angles. The IR cameras can pick up the reflective markers and the collection of different angles of the same scene allows 3D information to be reconstructed. These reflective markers are less cumbersome than equivalent active

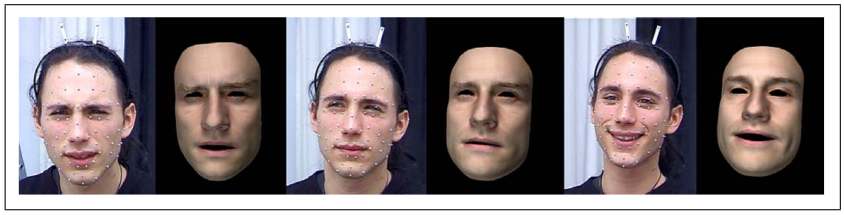

Figure 2.1: Example of passive marker motion capture [14]

sensors, in that the actor has full range of motion within the performance area. However, these markers can miss out on finer, subtle motions such as skin deformation, or muscle flexion.

2.1.3 Motion capture of faces

The set up for facial motion capture often has an actor wearing a number of reflective markers on their faces placed upon highly motile areas of the face such as eyebrows, eye lids, mouth etc. Face markers are often much smaller than those used for body motion capture [11] [14] [23]. The actor is often seated in a smaller performance area with cameras arranged in a tight half circle in front, although with higher resolution cameras it is possible to capture both body and facial motion simultaneously. Markers do not have to be reflective balls. Taking advantage of the fact that the actor does not move and is always facing the same direction, reflective paint can also be used for effective motion capture [6] [7]. Markers do not even have to be present at the time of capture. Motion capture can also be performed on video/image analysis using computer vision techniques to overlay markers on facial features in photographs/video frames [13] [30] Markers do not have to be dots either. Na et al. [28] use a contrasting stripe pattern to reveal fine facial details, such as wrinkles.

With improvements in camera hardware and facial analysis it is possible to perform facial motion capture without markers. One way, as seen in figure 2.1, or by annotation of video or images but also through improved body and face recognition techniques. Vlasic [37] performs decomposition and compositing of faces using video frames, Beeler at al. [5] rely on the repetitive nature of facial motion to produce *anchor frames* from marker-less motion capture, which are then used to produce vertex accurate 3D models. Weise et al. [39] [38] use Microsoft's Kinect camera and in-built depth map extraction to allow real-time puppeteering of a digital avatar.

2.2 Motion Retargeting

Retargeting is the process of adapting the animation of one character to one or more other characters. Difficulties can arise when source and target differ in proportion or underlying structure. Consider attempting to adapt a human walking animation to another animal, or vice versa. Skeletal structures differ greatly in both size and function, for example, many four-legged animal's knees bend differently to a human's. Body retargeting can be a difficult process.

Choi and Ko [12] performed body retargeting using inverse rate control which determines the changes in joint angle using the change in end effector position over time. The calculation was done in real time allowing for interactive performance retargeting. Dariush et al. [15] present a human to robot retargeting system (using the ASIMO robot) that can operate in real-time and does not require a marker based motion capture approach. The system uses only a depth map of the actor. The system can generate robot joint movements while considering joint constraints, self-collision constraints and balance constraints.

Gleicher [18] presents a retargeting system for adapting from one articulated character to another, with identical structure but differing segment lengths. The idea of the system is to allow retargeting but to preserve interesting or unique characteristics of the source motion by identifying these characteristics as constraints that must be maintained in the retargeting. Hecker et al. [20] built a system designed to allow retargeting of animations to unknown morphologies. The system was incorporated into the video game *Spore*, which allowed players to design their own lifeforms with a vast array of potential morphologies (Figure 2.2). Animations are designed and stored

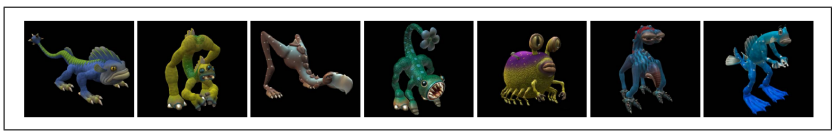

Figure 2.2: Sample of user-created morphologies in *Spore* [20]

independently of morphology, preserving the structural relationship and style. At runtime, the animation data is applied to the user-created morphology to produce a set of pose goals which are in turn supplied to a robust inverse kinematic solver. The system allows for the design of animations for creatures with vastly different skeletal systems and morphologies. Hsieh et al. [21] introduce a system for retargeting animations to different skeletal structures (specifically a dog and a human). Both source and target skeleton are loaded in, the system constructs a combined skeleton with users assigning bone correspondences. The target's initial pose is then lined up with the source's. With the unified skeleton it is possible to transfer motion data from source to target.

Monzani et al. [33] had previously experimented with an *intermediate skeleton* in order to allow easier retargeting of source motion capture data to a target avatar, however their focus was on structurally similar characters. Yamane et al. [40] focused on bringing human recognisable motion and expression to non-human characters (anthropomorphism). The system is presented as an alternative to key-framing using motion captured human performance of the target character. The method used employs a statistical mapping function learned from a set of character key poses and a physics based optimisation to improve realism.

2.2.1 Retargeting of faces

Blend-shape mapping

There are a number of approaches taken for the adapting of facial animations to different models [24]. *Blend-shape mapping* involves developing blend-shapes for the source and target models and using different weights to apply target to source [30]. The individual blend-shapes can be based on the Facial Action Coding System [17]. This method relies heavily on an artists ability to build these blend-shapes and requires there to be an equivalent number of associated blend-shapes between source and target. However it can be useful in retargeting for models that are not of the same general structure as the source.

Weise et al. [38][39] have been developing a system that allows a user to control the facial expressions of a digital avatar in real time using their own performance. The system uses the Kinect camera to capture user expression in both a 2D image and 3D depth map. Combined with a collection of generic blend-shapes, a user specific blend-shape collection is produced. Then a combination of blend-shape weights and previously learned animations are used to map the user's performance to the digital avatar. The rig uses only the Kinect camera, eschewing markers or 3D scanners to be unobtrusive.

Curio et al. [14] use a combination of 3D scans of the actor's face, acquired motion capture data and the Facial Action Coding System [17] to retarget acted expressions onto a morphable 3D model. The process works by matching semantically, an Action

Unit with both an associated 3D scan and motion capture data and then weighting the Action Units as necessary. Resulting basic expressions (angry, sad, etc.) become linear combinations of different Action Units. Constructing the 3D face model involved aligning each scan with that of a neutral scan using an *Iterative Closest Point* algorithm and applying a control point network to the surface mesh. Individual Action Units were combined and used within a single blend-shape to build blend weights. Motion capture blend weights extracted in a similar manner using a least-squares fitting of two 3D data sets.

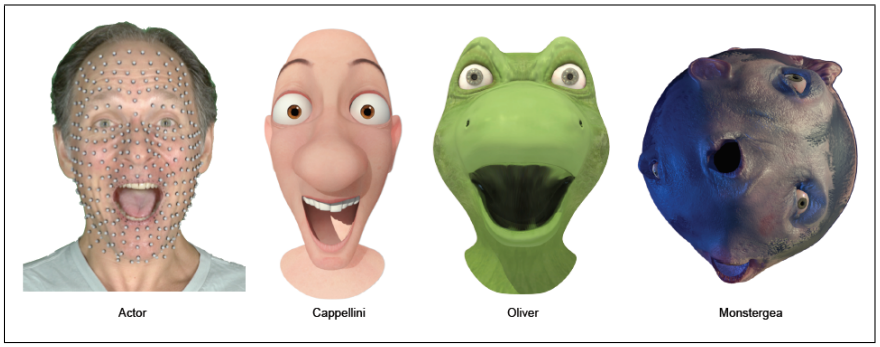

Figure 2.3: Retargeting actor's expressions to non-human characters [23]

Kholgade et al. [23] attempt to adapt human motions to non human characters using parameter parallel layers. They employ three layers: emotion, speech and blink, each focusing on a distinct facial action. Animation artists provide equivalent facial motions for the target character; e.g. the artist decides how a crocodile smiles. Layers are weighted and combined to produce the expected retargeted motion.

Orvalho et al. [8] begin with a generic face and allow an artist to deform it to create a desired character. The expression blend-shapes defined for the generic rig are then adapted to the deformed face. Chuang et al. [13] avoid constructing blend-shapes for a source model and instead apply motion data direct to a target model. The system uses video for its source acquiring potential blend-shapes and weights from the video performance and then having and artist construct associated blend-shapes for the target. The artist is free to design the look of the target with its animation being driven by an actor performance. Song et al. [34] focus on retaining stylistic facial

shapes and timing to emphasise the significant characteristics of the target model. They construct a prediction model to match semantically corresponding blend-shapes. Then, through combination Radial Basis Function, Kernel Canonical Correlation Analysis and 1D Laplacian motion warping, replaces stylistically important emotional sequences, preserving the characteristics of the target model.

Geometry mapping

Some work has also been done using radial basis functions and geometric mapping in place of blend-shapes, where the systems use a deformable mesh as both source and target.

Na and Jung [28] use a deformable mesh to perform retargeting. At each frame, feature points are extracted from an original mesh (constructed from image capture) and converted into a base mesh for the actor. This base mesh is then applied to the target mesh by consulting previously learned examples (Figure 2.4). The base mesh is then subdivided further to produce a denser mesh and again applied to the target mesh. The process continues to a specified density level. Vlasic et al. [37] allow the

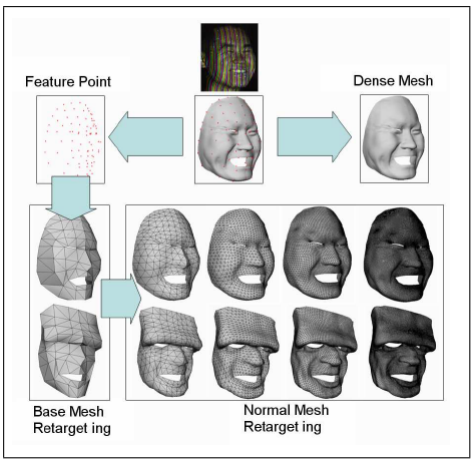

Figure 2.4: Overview of base mesh subdivision[28]

manipulation of faces in video using multilinear models comprising: actor pose and identity, viseme (mouth articulation) and expression. They show that the separation of these attributes allows for individual variance and interoperability. Combinations of

these attributes from different actors can be blended to create a composite result and then reapplied to the video.

Style learning

Ma, Le and Deng [25] present a style learning framework that can build a constraint-based Gaussian Process model from a small set of facial-editing pairs, which can then be used to automate the editing of any remaining animation frames. Miranda et al. [26] show a sketch based animation control interface, where an artist can use a canvas to specify the curvature and position of different key features of the facial model.

2.3 Conclusion

As can be seen there are a number of different approaches to both human motion capture and the retargeting of human motion. Due to the nature of bone-driven models and their ease of use, it would be useful to employ passive reflective markers for animation capture as the marker locations can be applied on the actor's face exactly where the model's bones would be, making for simple mapping of data.

For facial retargeting, it may be possible to adapt the concept of Action Units and the Facial Action Coding System [17] for use with bones to specify different expression types for a source and target model and use computable weights to determine expression combinations. It would be necessary to specify the different movement constraints of the different models (for characters with non-human proportions). This however would require constructing Action Units for every possible expression which may prove to be too time consuming.

Alternatively it may be possible to select a smaller set of expressions displaying maximum/minimum range of movement of each facial feature and interpolating target bone positions based on the proportion to which the source feature has moved between maximum/minimum extents.

Chapter 3

Design

As was discussed in chapter 2, there are a number of different approaches to facial motion retargeting. However, there is little literature available on facial retargeting for bone-deformable models. With this in mind, the application will be experimental and will investigate different possible approaches. Section 3.1 will state the aim the application hopes to achieve. Section 3.2 will discuss the different constraints the application will have to respect. Section 3.3 will define the design of the application, certain assumptions concerning its use and the different approaches considered. Finally, section 3.4 will describe the tools and assets employed to build the application.

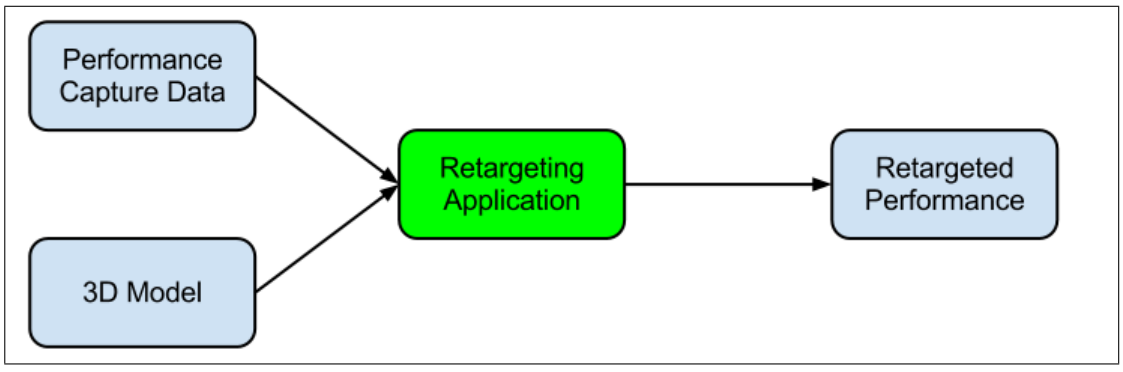

Figure 3.1: High level overview of proposed application

3.1 Application Goal

The goal of the application was to provide a black box executable that required only performance data and a target model, and returned a retargeted animation for the target model. In the course of building the application, different approaches to facial retargeting would be considered and tested.

3.2 Requirements

As the application could potentially be added to a pre-existing performance capture pipeline, it was necessary to adhere to certain requirements.

- **Work directly from marker data.**
  The supplied motion capture data is taken from a number of different performance sessions with a diverse group of actors of different age, gender, size, etc. Due to this and with respect to the time frame of the dissertation, it would not be possible to create a 3D model of each actor for use as a source model in facial retargeting. It was then necessary to design an application that could retarget directly from the performance data.

- **Work with arbitrary performance data.**
  In addition to working from marker data, it would also be necessary to handle different performance metrics, such as differences in performance length and number of markers used.

- **Map to arbitrary bone-deformable 3D model.**
  For the application to be useful as an automated process the user should not have to rewrite and recompile the application to handle different 3D models. Therefore it is necessary for the application to handle any human bone-deformable model in a generic manner.

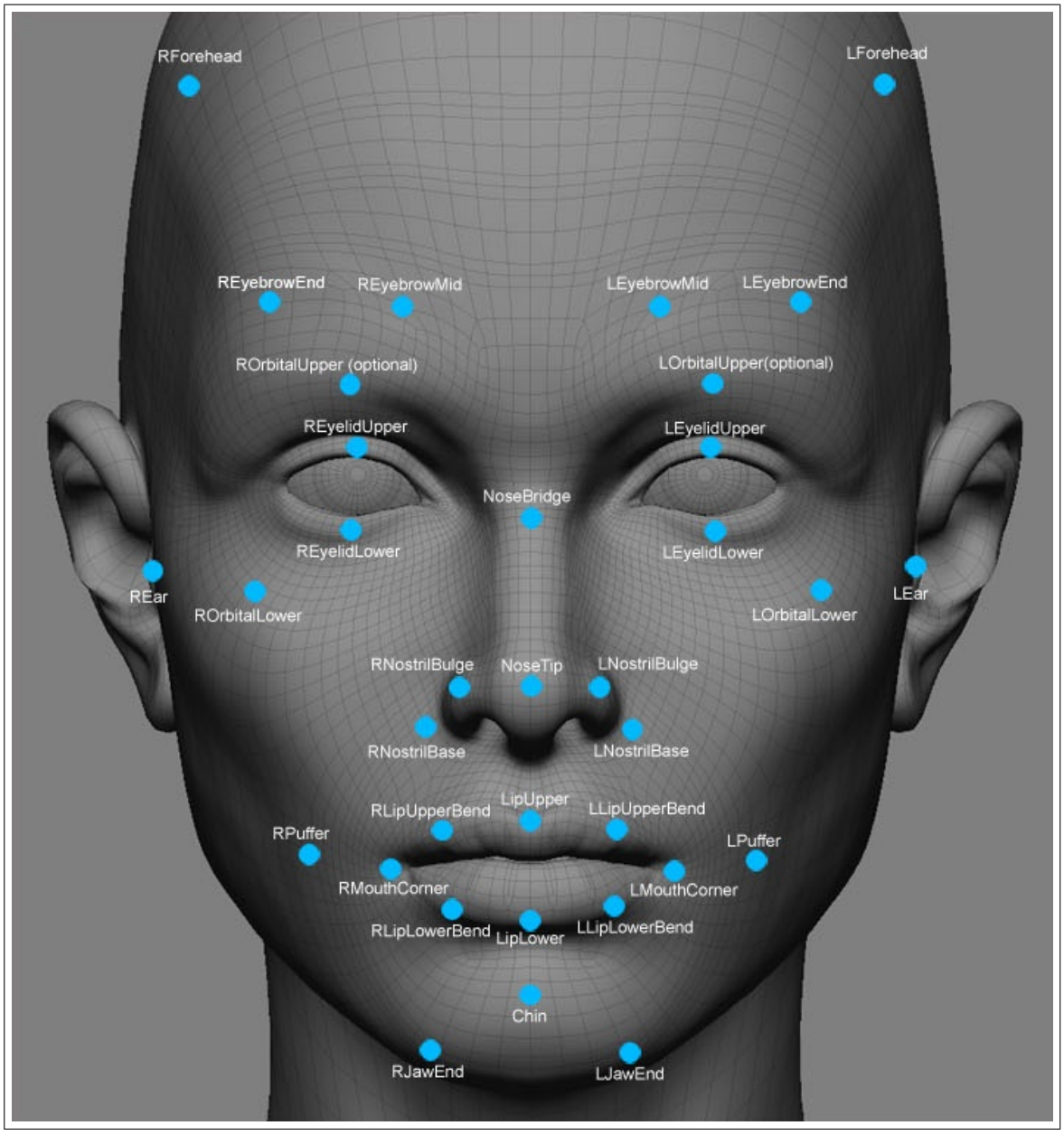

Figure 3.2: Autodesk Face Robot suggested marker set up[1]

3.3 Methodology

The application intends to take facial motion capture data and a 3D human model, fit the data to the model and return a retargeted animation. We decided to use the trace file (.trc, appendix A) format for the motion capture data. This format provides similar fidelity to other motion capture formats (such as .c3d or .csm) and contains the raw marker data and performance metrics. Additionally, the format is plain text so it would be simpler to parse than a binary file. The parser will make use of the standard C/C++ libraries for file I/O.

There are many variations on marker set up for facial motion capture. It was decided early in the dissertation that it would not be possible to cater for every possible marker variation. This application assumes marker set up and naming convention as shown in figure 3.2. This marker set up is minimal, covering the major facial landmarks (features). While the final application can handle any number of additional markers and marker to bone mappings, it assumes this minimum set up (figure 3.2) and that this naming convention is followed: FaceRobot_marker Name, e.g. FaceRobot_RForehead.

There are also a number of different bone set ups for 3D models. Instead of limiting to one set up type, it was decided to use an environment that would treat model bones similarly, regardless of model set up. To do this, we used the Ogre3D (section 3.4.3) rendering environment, in conjunction with the OgreMAX plug-in for 3ds Max (section 3.4.2). The plug-in can export many 3D model types into an Ogre friendly .mesh.

For motion retargeting, it was decided to attempt to match bone movement with the movement of a corresponding marker. As the range of facial positions could not be known with arbitrary performance data, it was decided that the method for retargeting would instead scale the range-of-motion of the source performance to the range-of-motion of the target model. The actor would perform a range-of-motion motion capture (appendix C) to prime the application with a source range-of-motion. Then an animator would specify the equivalent range-of-motion for the target 3D model. The application would then take performance data from the source actor and the target 3D model and adapt the performance by mapping between the source and target

ranges-of-motion.

3.4 Tools and Assets

This section will list the tools and assets employed in this dissertation. In addition to the following entries, the application itself was written in C++.

3.4.1 Vicon Blade

Vicon Blade [36] is an application that handles many elements of motion capture. It handles configuration of the performance space and actor set up (camera calibration, actor T pose) as well as capturing good quality data. It is primarily used in this dissertation for performance data pre-processing (sections 4.1.1, 4.1.2 and 4.1.3).

3.4.2 Autodesk 3D Studio Max

3D Studio Max [2] is a modelling environment used for creating and manipulating 3D models. It can be used to apply different rendering effects as well as design and apply different animations for 3D models. It is used in this dissertation to prepare and export the 3D model as well as define the model's limits (sections 4.1.4 and 4.1.5).

3.4.3 Ogre 3D

Ogre 3D [29] is an open source 3D graphics engine; a comprehensive collection of rendering libraries for use with C++, OpenGL and DirectX. It is used in this dissertation to animate 3D models with retargeted animation and render them to screen (sections 4.2, 4.3, 4.4, 5.1, 5.2 and 5.3).

3.4.4 Performance Capture Data

Performance data was provided by the GV2 [19] lab of Trinity College Dublin. This data was provided in trace format (.trc) and is a common performance capture format (appendix A). Supplied performance data closely follows the suggested placement shown in figure 3.2.

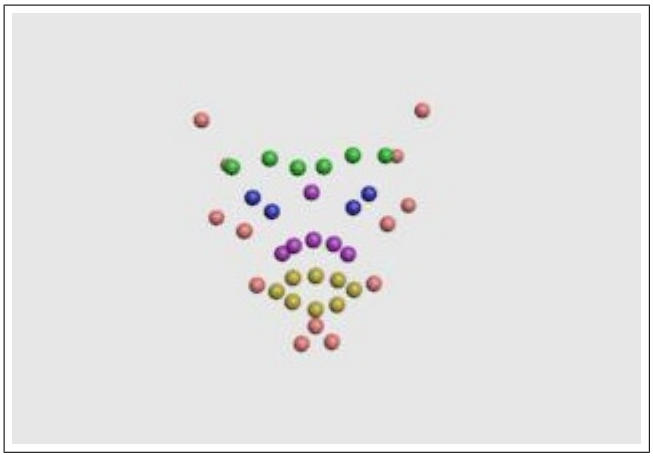

Figure 3.3: Marker data rendered with 3D studio max. Colours added for clarity.

3.4.5 3D Models

Two 3D human models (figure 3.4) were also supplied by the GV2 lab of Trinity College Dublin. Both models represent average male and female body type, with differing proportions and range-of-motion in the face.

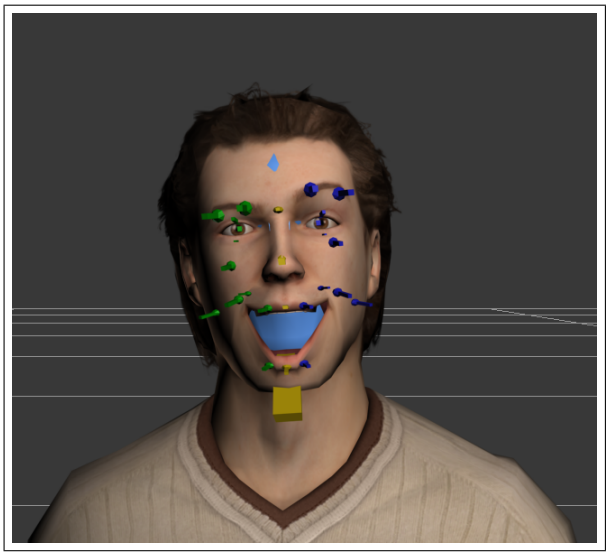

Figure 3.4: Example 3D model as it appears in 3D studio max. Coloured protrusions represent facial bones.

Chapter 4

Implementation

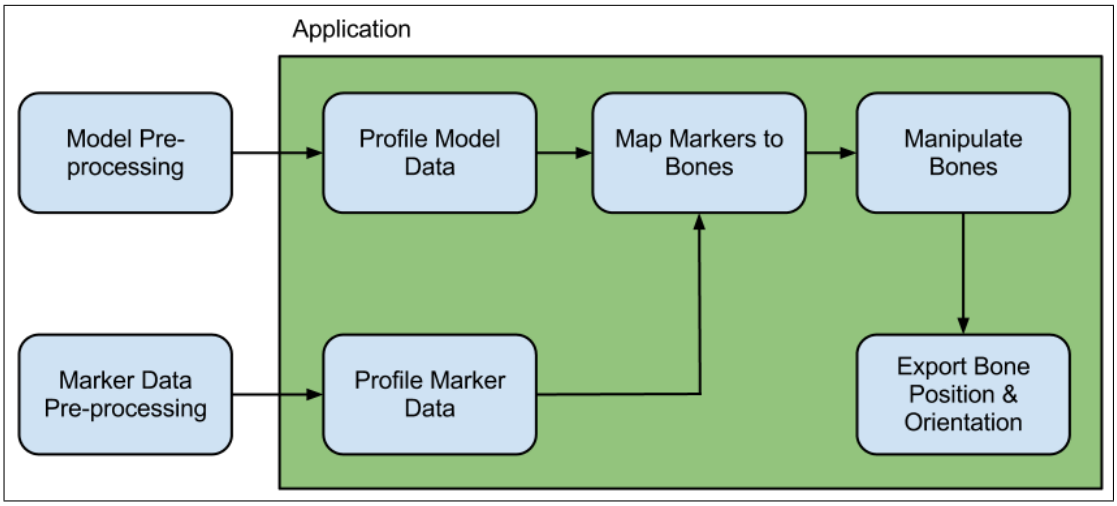

Figure 4.1: Overview of the Retargeting Application pipeline

This chapter is concerned with the implementation of the retargeting application. It will detail the steps taken in preparing the marker data and 3D models for use in the application (section 4.1). The acquisition of important information from the marker data and model mesh are detailed in section 4.2 as well as associating markers to corresponding bones in section 4.3. Finally, the process of calculating new bone positions and transforming the model bones to occupy those positions are detailed in section 4.4.

4.1 Pre-processing

Before entering data into the application it is necessary to perform some pre-processing to prepare it. Section 4.1.1 details the steps necessary to separate overall body and head motion from individual facial motion. In sections 4.1.2 and 4.1.3 we discuss common methods employed in performance capture post processing, specifically the removal of unwanted marker oscillation and filling in any missing data points, respectively. Finally, in section 4.1.5, we discuss the process of specifying the allowable movements of the model's facial bones.

4.1.1 Remove Head Motion

This dissertation is concerned with the retargeting of facial motion. It is not guaranteed that the performance data supplied will be free of extra motion, either from head movement or from full body movement. It is therefore necessary to isolate facial feature movement from all other motion, so that retargeting can be successfully applied.

To remove extra motion, we first locate a number of *stable* markers on the head. Candidates should have, ideally, no independent movement; their perceived movement can be attributed solely to head motion. Markers chosen for this implementation were those surrounding the head, the nose bridge and tip (*RForehead, RBHD, REar, LForehead, LBHD, LEar, NoseBridge, NoseTip*, in figure 3.2). *BHD* markers (not visible in figure 3.2) are placed at rear of head, in line with *Forehead* markers). These markers were considered to be the most *stable*.

With the *stable* markers chosen you can then use them to construct a rigid body. The data from the first frame of the performance is used as a reference point. For every subsequent frame the rigid body is reconstructed and compared with the rigid body from the first frame. Any translation (change in position) or rotation (change in orientation) the rigid body has undergone, for this frame, is reversed for **all** markers in that frame.

Changes in position can be reversed by calculating the displacement of the rigid body's centre point (an averaging of marker positions) and removing that translation

from the marker position. A change in rotation is reversed by determining the average angle between first frame and current frame position of the rigid body, for each axis in turn. A rotational matrix is constructed to correct for the changes in pitch, yaw and roll (rotations about X, Y and Z axes, respectively) and applied to the marker vector.

Since the *stable* markers do not represent facial features, any movement they undergo can be attributed to head motion. Reversal of these translations and rotations has the effect of cancelling out head motion, thereby leaving the data with only facial feature movement.

4.1.2 Filter Noise

With optical motion capture, a 3D scene is reconstructed from multiple 2D images taken from cameras placed at different points in the performance area. During reconstruction, the marker positions are calculated by referencing the 2D images, camera positions and orientations. By knowing the camera configuration and set up it is possible to determine the depth of a marker from multiple 2D scenes. In this way, the 3D position of the marker in the scene is calculated. However, this process is contained between frames, there is no continuity kept between frames. Therefore, it is possible for the marker to have subtly different 3D positions from frame to frame, independent of any performance movement. This data artefact is called marker oscillation.

For fine features, such as the face or hands, these oscillations can present unwanted motion, therefore it is necessary to lessen their impact or to remove them entirely. This can be done by passing the data through a signal, or noise filter. Signal filters are tasked with removing or lessening certain aspects of signal data, such as noise or specific frequency intensities. One signal filter, the Butterworth filter [9], can potentially smooth a signal and prevent deviation from a specified spline. This filter will help minimise marker oscillation. The Butterworth filter is supplied with a threshold and order value. The threshold value specifies the active range of the filter. Marker values outside of this range are ignored. The higher the order of the filter, the greater the smoothing applied. Care must be taken to choose an adequate threshold value and filter order. Too low a threshold value will result in no appreciable change to the data.

Too high, and you risk losing some viable performance. Similarly for the filter order.

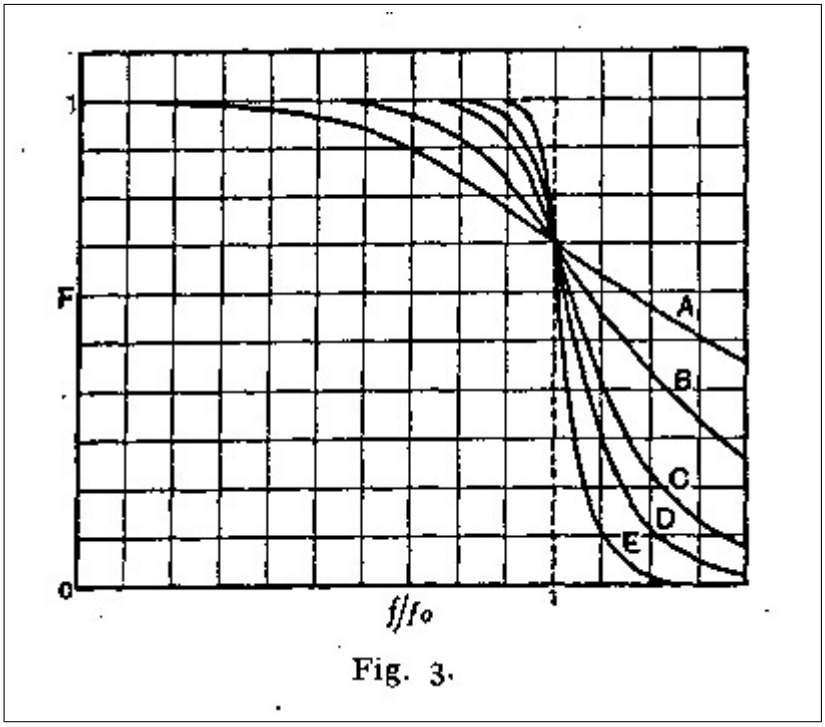

Figure 4.2: Frequency response plot of a Butterworth filter[9]

For this implementation, the Butterworth filter was applied using a filtering script within Vicon Blade (3.4.1).

4.1.3 Clean Marker Data

During performance capture it is possible for markers to become occluded from the IR cameras. This can happen if the performer places their hands in front of their face, or looks down at the ground, or has manoeuvred into a position that cannot be captured by the cameras (outside focal range). When this happens the marker is lost, or dropped, from the scene for the duration of the occlusion. The system does not know where the marker is. This results in missing data points when later processing the performance.

For markers used to depict rigid bodies, such as a limb, other markers in the rigid body can be used to determine the missing marker's position for the relevant frame. However, markers for the face do not necessarily contain an underlying rigid body. Therefore a different approach is needed.

For a large number of missing data points, it may be necessary to recapture the performance entirely. However, for small gaps (1 - 10 frames) it can be corrected simply. First, cycle through the data and locate the frame(s) at which the marker has dropped, it will appear as a blank spot in the data profile. The data for the missing frames can be calculated by constructing a spline from the point the missing marker was lost to the point it re-emerged, and using the spline to interpolate a position for the new marker. This effectively plugs the hole in the data, ensuring the trajectories are complete for the entirety of the performance.

4.1.4 Preparing Model

Before the 3D model can be used within the Ogre 3D environment it must be prepared and exported using the OgreMAX plugin. It is necessary to first ensure that the model mesh has an associated skeleton. Without a skeleton, it is not possible for this application to perform bone manipulation. Similarly, if the facial mesh is not associated with the facial skeleton, manipulation will not be possible.

Ideally, upon exporting the model with the OgreMAX plugin, there should be three files: .mesh, .skeleton and .material all with the same prefix, or filename. The .mesh file contains information primarily concerning the model's geometry as well as pointers to the associated skeleton and material files. The .skeleton file contains information concerning the different bones of the model. It specifies bone names, bone hierarchy, position, orientation and their effects on the mesh geometry. These are the bones that will be manipulated within the retargeting application. The .material file specifies different rendering effects to apply to the model. It is sometimes necessary to edit this file to ensure the model is actually viewable in the retargeting application incorrect values will not be recognised by the renderer. For this implementation, It was necessary to change a vector of all zeros to an easier to see value, best results all white (a vector

of all ones).

4.1.5 Specify Model Range-of-Motion

Specification of the model's allowed range-of-motion is a crucial part of the pre processing stage. Providing the model's range-of-motion to the retargeting application allows the application to perform successfully.

The specification process involves taking each bone in turn and manually adjusting its position and orientation along the X, Y and Z axes. The user should manipulate each bone to its minimum and maximum allowed positions in each axis. Once each position has been reached, the user should record these values into a text file (Appendix B).

The process can be time consuming as a high quality model can have a number of bones to manipulate. The models used in this implementation required specification of 21 different bones each. Additionally, the specification process is subjective for the user. It is up to the user to correctly define a good bone range-of-motion. However, once the process is completed for the model and specified range-of-motion is satisfactory, it need not be performed again. The range-of-motion for this model can now be used with any performance data supplied to the retargeting application.

The specification process can be performed either within the retargeting application (figure 4.3) or within modelling software, such as 3D Studio Max (3.4.2).

4.2 Profiling Data

With the performance data and model prepared, the next stage of the pipeline is to import the data into the retargeting application and to parse out th relevant data. The performance data is supplied to the retargeting application in TRC format (Appendix A). This is one of a number of different performance capture formats and was chosen for this implementation because the format can be read as plain text, thus making it simple to parse.

Figure 4.3: Specification process within retargeting application. Top pane provides notable information for bone limit specification

The TRC file contains information on the performance metrics such as: frame rate, scale, number of frames, number of markers. This information can be used to build data structures in C/C++, as well as tailor the application for playback speed, camera positioning, etc.

The TRC file also contains marker positions for each frame of the performance. This data is saved in the *Frame:xyz* and *Frame:frameNumber* attributes, figure 4.4(B). Frames for each marker are grouped together and stored in the *MarkerNode:frames* vector attribute, figure 4.4(C).

4.2.1 Remove Hierarchical Motion

Once the marker data is in memory the application begins to remove hierarchical motion. Hierarchical motion is any movement a marker undergoes that is a direct

result of another marker's movements. In the case of this implementation, it was determined that the movement of the chin and jaw markers influences the position of the bottom lip markers.

There is a biological hierarchy at work here where the position of a person's bottom lip can be influenced by the movement of their mandible (jaw bone). For this implementation, the chin and jaw markers are driving the model's mandible and are therefore imparting some additional movement to the bottom lip. For the bottom lip markers to effectively drive the model's bottom lip bones, it is necessary to isolate their movement from any hierarchical movement. Failure to isolate this movement will result it unwanted bottom lip movement, causing the lips to bend and part in unrealistic ways and potentially causing skin tearing in the model.

One method that we considered for removing hierarchical motion was to create a spring system between a marker and the parent marker causing extra motion. If the pull on the spring went beyond a certain threshold, the translation of the parent marker for that frame would be removed from the child marker. In this way, markers closer to the parent marker would be effected by smaller motions more so than further markers, as the spring would pull taught faster. Initial results were promising, however, it became difficult to choose a threshold value that would work for arbitrary mouth movements. The approach was abandoned in favour of simpler solution. It is believed that this approach could still yield good results, but requires further investigation.

The approach eventually implemented was, rather than completely remove the parent motion equally from all child markers, only a portion would be removed. The proportion, or weight, of removal is based on the ratio between the perpendicular distance of the child marker from the parent markers rotational axis versus, the absolute distance between the two markers. This weighting has the effect of removing large amounts of parent movement from child markers that are both close to the parent marker and inline with the parent's rotations. Conversely further away markers are less effected by the parent markers motion and thus receive a small weight. While this approach does lessen the impact of hierarchical motion, it can also cause some degradation in mouth visemes. This can detract from the quality of the final retargeted animation.

4.2.2 Determining Performance Range-of-Motion

With the hierarchical motion removed, each marker's positional data is now independent of any other marker. The next stage is to acquire the range-of-motion of the performance. This is done simply by looping through the frame data and determining the lowest and highest values reached for each axis (*ModelNode:minX, ModelNode:maxX*, etc., figure 4.4(C)).

Knowing these values, the position of each marker at each frame is then checked for what proportion of the full range of that marker it represents, for each axis (*Frame: proportionMovementX*, etc., figure 4.4(B)), resulting in some value between zero and one. Zero would indicate minimum position for that axis, while a value of one would indicate maximum position.

Finally, the rest pose for the performance is acquired. Earlier implementations took the average position of the marker for the entire performance to be the rest pose, however this resulted in some odd behaviour over time. It was eventually decided to simply use the marker's positions in the first frame as the rest position (*ModelNode:startX*, etc., figure 4.4(C)). The reasoning being that the performer would be relaxed and their face in a neutral expression at the beginning of the performance.

4.3 Mapping Markers to Bones

4.3.1 Importing Model

The model .mesh is imported into the Ogre environment as an *Ogre:Entity* and attached to an *Ogre:SceneNode*. It is necessary to attach the model to an *Ogre:SceneNode* so that it is actually rendered on screen. Creating an *Ogre:Entity* allows for instancing of the model, however the model is only used once in this implementation, so the instancing is not taken advantage of.

Upon importing the model .mesh, Ogre automatically loads the corresponding .skeleton and .material files. The .material file defines some rendering effects for the model but is ultimately of little interest beyond ensuring the model is visible.

The .skeleton file allows Ogre to acquire the skeleton associated with the model .mesh. This allows the application to manipulate this skeleton and in turn, deform the mesh.

4.3.2 Mapping

The retargeting application uses the mapping file (Appendix B) to acquire the bone names. With the bone names it is possible to index the model skeleton and acquire a pointer to the bone. The bone is set to manual control and stored as *ModelBone:bone*, figure 4.4(A). Additionally the mapping file is also used to fill in the range-of-motion for the bone, whether the bone movement is positional based or rotational based, and the name of the marker driving that bone. The marker name is then used to find the marker index.

Finally, the resting position of the bone is compared with the bone's range-of-motion to determine the proportion of the range the bones have traversed. This information will be used later when moving the bones (Chapter 5).

4.4 Transforming Bones

The following actions are performed in the order specified; rotations are performed before translations. When translating a bone to a new position, it is locked to that position for that frame. The bone position will not update if a rotation is later applied. This has the effect of stretching the model's skin. For example, if a bottom lip marker is repositioned and the jaw is then rotated to a point where the mouth should be open, the skin will instead stretch over the distance. For this implementation, a bone's movement cannot be both rotational and positional based. The majority of bones in the model are positional based. A change in their position results in an equivalent deformation in the skin, whereas a rotation of these bones results in unwanted deformation. Rotational bones, such as the jaw, operate in a different way; rotations correctly deform the skin.

4.4.1 Rotation

There is a simple calculation to determine the angle by which to rotate a bone by. The retargeting application determines the angle between the vector of the driving marker's rest position with the vector of the current frame's position. The associated bone is then rotated by that amount.

The current implementation performs these rotations in reference to world space. This was done for simplicity, however it may cause problems if head or body motion is later reintroduced to the scene. One way to adjust for this would be to perform rotations in the bone's parent's space. That should ensure any rotations undergone by the head or body will be accounted for.

4.4.2 Position

The following list details the different methods used to determine a proposed translation for each bone:

- **Direct Mapping**
  The proposed translation is simply the corresponding marker's position for that frame. Translation is scaled to the model's size.

- **Displacement from rest**
  The proposed translation is calculated as the associated marker's displacement from its rest position for that frame. Translation is scaled to the model's size.

  ```
  proposedNewBonePosition = boneRestPosition + (
      currentMarkerPosition - restingMarkerPosition)
  ```

- **Range-of-Motion Matching**
  The proposed translations is determined by measuring the percentage movement of the corresponding marker with respect to its minimum and maximum positions in each axis and applying the equivalent percentage movement for the bone. Scaling is not necessary as the bone translation is calculated in the model's scale.

  ```
  boneRange = boneMaxPosition - boneMinPosition,
  ```

```
translation = boneRange * markerPercentageMovement,
boneRestingTranslation = boneRange *
    boneRestPercentageMovement,
translation = translation - boneRestingTranslation
proposedNewBonePosition = boneRestPosition + translation
```

It is not enough to simply translate a bone by the proposed translation. Proceeding in that way will result in unwanted translations by the bone. This is a result of the proposed translation being multiplied into the different parent spaces and being subjected to their changes in position and orientation. Because of this, the proposed translation will be altered and, invariably, will not translate the bone to the desired position.

It is therefore necessary to convert the proposed translation from world space values to its parent's space and additionally account for the bone's orientation.

```
newBonePos = inverseParentOrientation * (parentPosition -
    proposedNewBonePosition)
```

This chapter outlined the steps taken to implement the methodology discussed in Chapter 3. The activities undertaken in the implementation result in the model undergoing bone deformation in its face. The following chapter will present the results of different retargeting approaches and will discuss the different advantages and disadvantages.

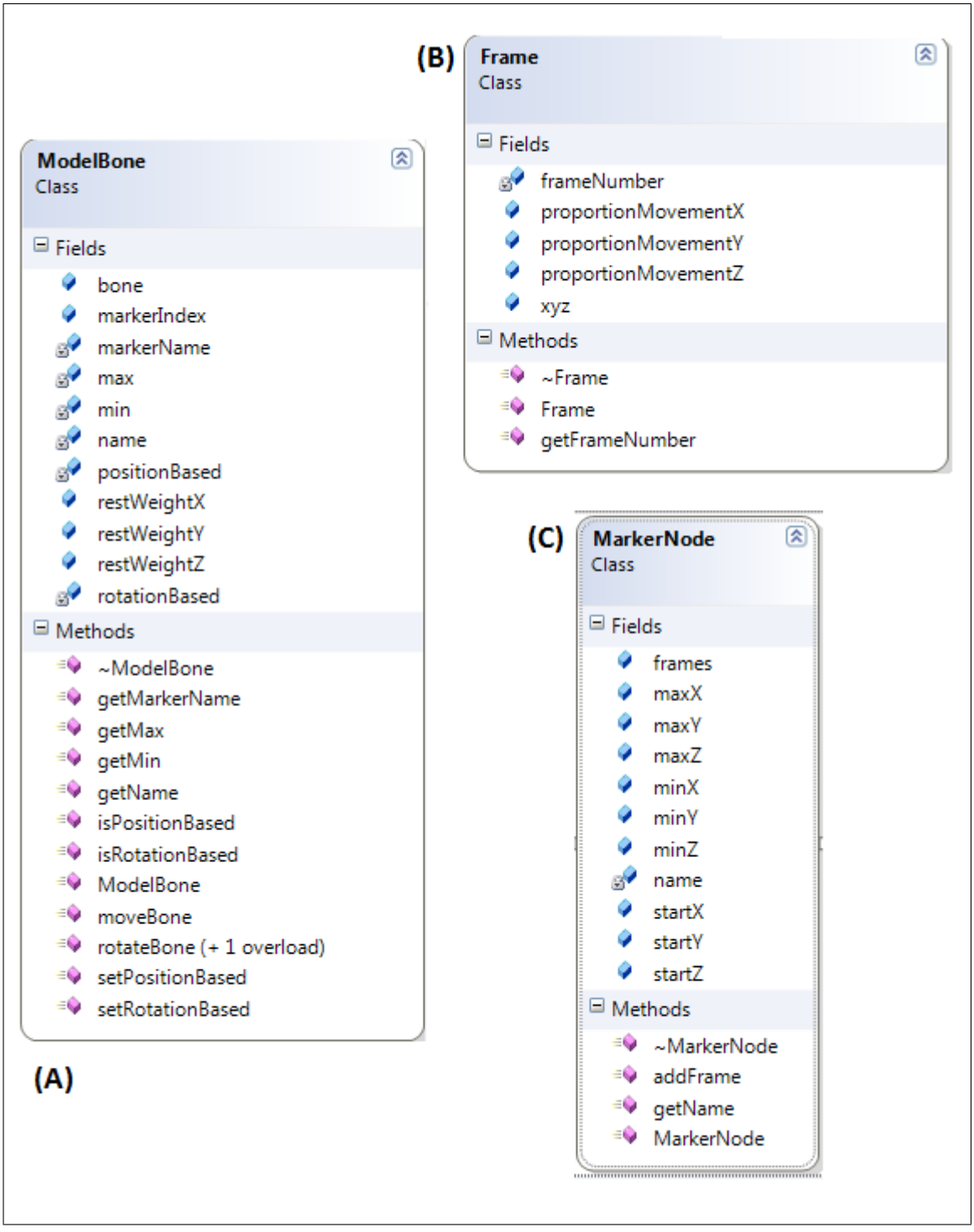

Figure 4.4: Class diagrams. (A) *ModelBone*, (B) *Frame* and (C) *MarkerNode*

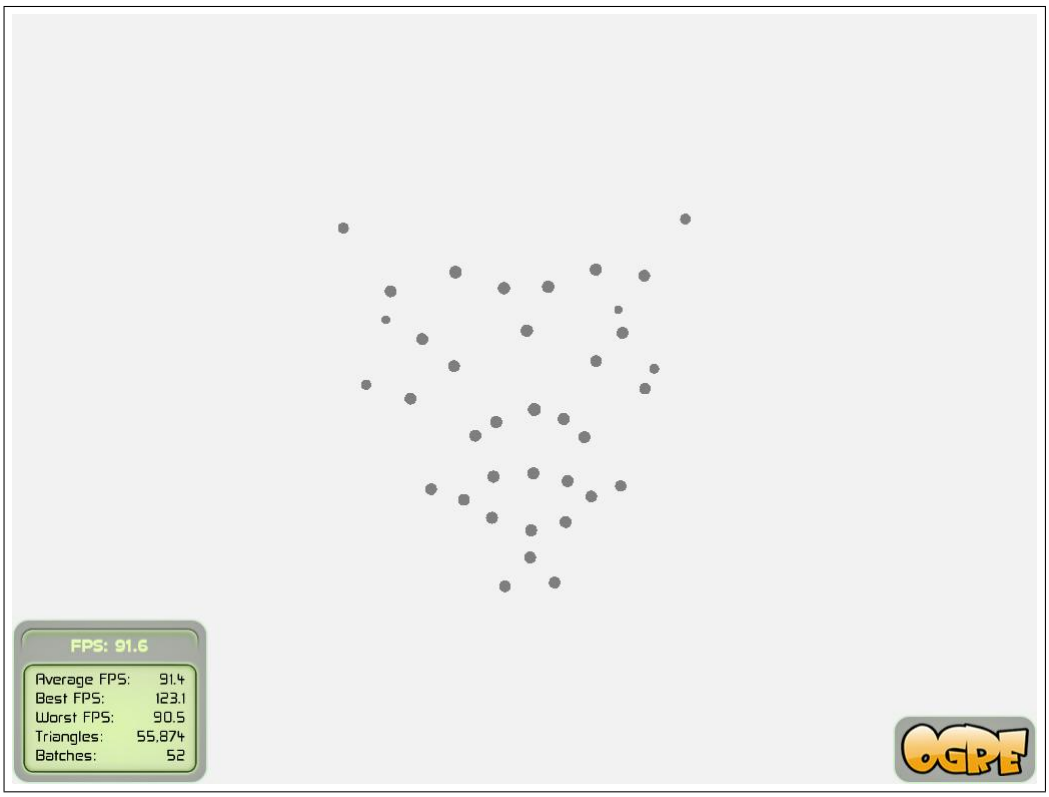

Figure 4.5: Marker data displayed in retargeting application

Chapter 5

Results

This previous chapter discussed the different steps the application performs in implementing retargeting for bone deformable models. This chapter will now present the results of different retargeting approaches and will discuss the different advantages and disadvantages for each approach. Section 5.1 will show the results of direct mapping (i.e. not performing retargeting) and is presented as a control case. Sections 5.2 and 5.3 present two different retargeting approaches.

5.1 Direct Mapping

Direct mapping simply takes the positions of the source performance markers and directly applies them to the target model bones, every frame. No attempt is made to adjust the motion for a differently proportioned model, or to scale the motion in reference to the target's range-of-motion. Figures 5.1 and 5.2 show the results of this approach. Direct mapping is presented as a control and example of the importance of retargeting.

Figure 5.1 shows a difference in proportion between the source actor and target model, in this case the target model has larger proportions. This results in animations tearing the facial mesh. Figure 5.2 shows both a difference in proportion and difference in height. In this case, the source actor is taller than the target model. Direct mapping will only result in good animations if both the source actor and target model are of the

same proportion and dimension. The target model would have to be a virtual replica of the source actor.

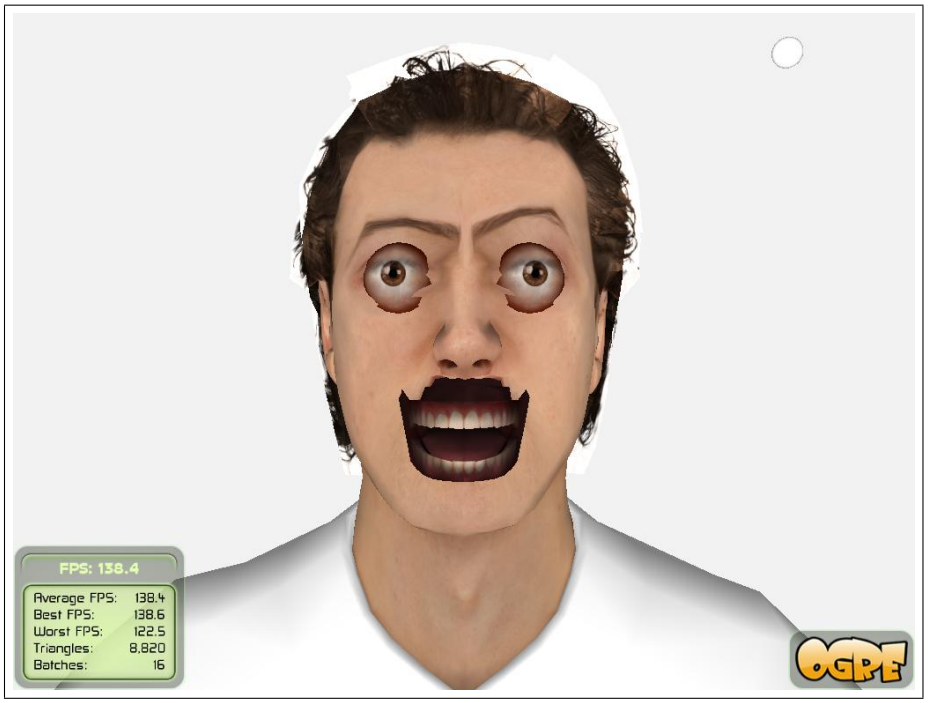

Figure 5.1: Example direct mapping, mismatch in proportion

5.2 Displacement from Rest

Displacement from rest was the first approach considered for retargeting. The idea was to determine a marker's displacement from some rest position and apply the same translation to that marker's driven bone. This approach would ensure that bone manipulation took the target model's proportion into account. Figures 5.3 and 5.4 show examples of displacement from rest.

Displacement from rest represents a substantial improvement over direct mapping and can be applied to any arbitrary bone-deformable model. However, this approach does not respect differences between source and target range-of-motion. This results in issues where source motion does not map equivalently to target motion. Consider figures 5.3 and 5.4, differences in source and target range-of-motion and resting position

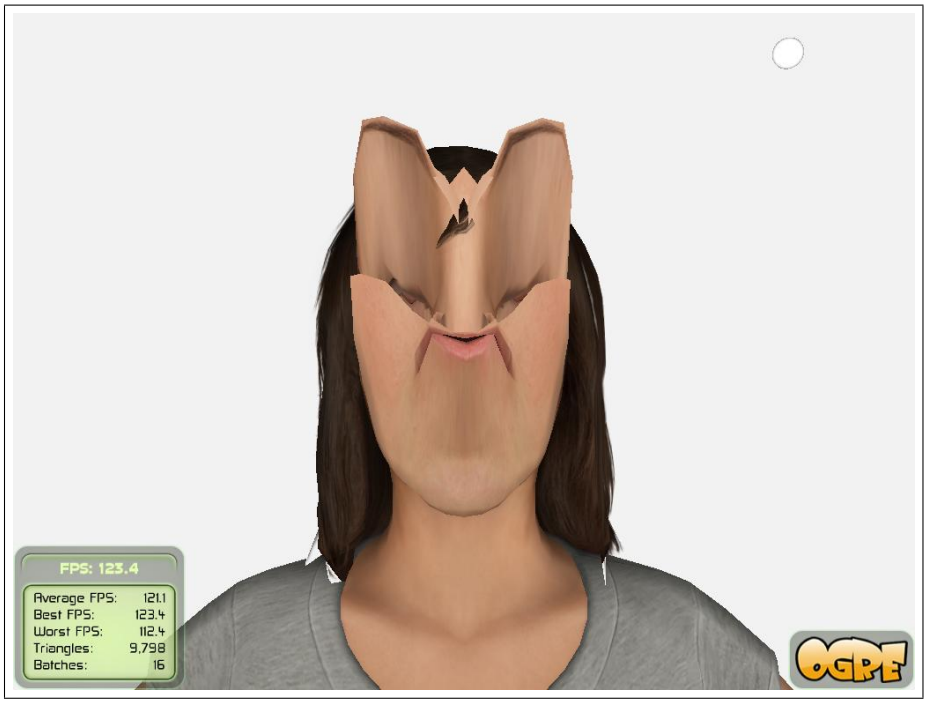

Figure 5.2: Example direct mapping, mismatch in proportion and height

lead to interpenetration of the top eyelid in both models as well as the mesh surrounding the mouth, collapsing. Other issues can include:

- Eyes opening too wide/ closing too far
- Incomplete blinks
- Mouth opening too wide/ not wide enough
- Mesh collapse at different facial features

5.3 Range-of-Motion Matching

Range-of-motion matching was designed to deal with issues manifest in displacement from rest (section 5.2). Range-of-motion matching involves determining the proportion of movement a source marker has undergone with respect to its minimum and maximum positions and scaling that motion for the target model. Scaled motion will ensure the

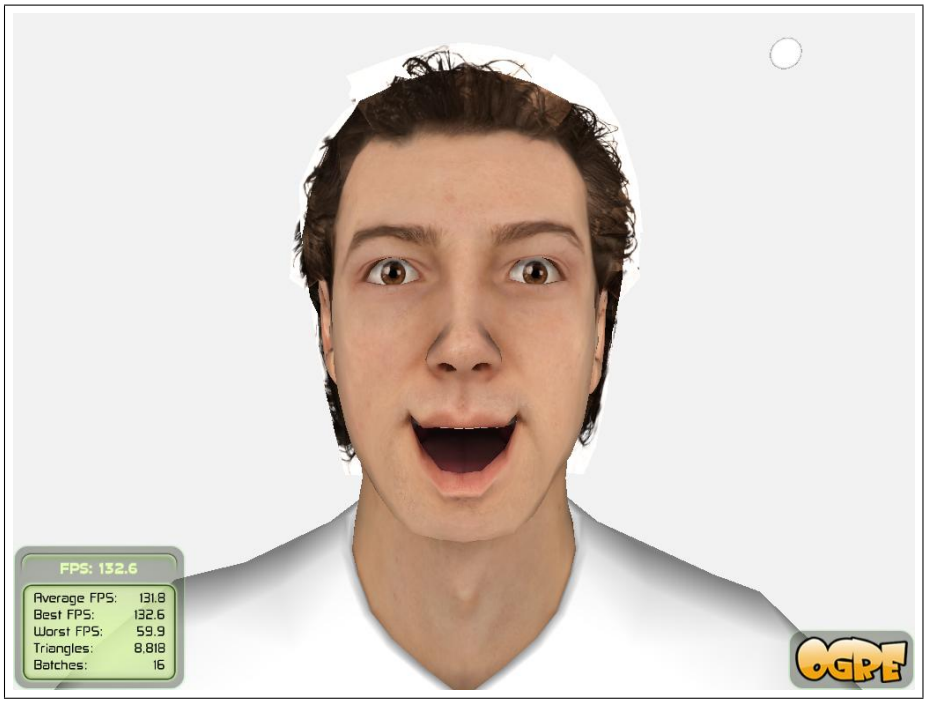

Figure 5.3: Example displacement mapping (1)

target model's driven bone will move the equivalent proportion with respect to its minimum and maximum position. For example, if it is determined that a marker has moved 30% of its full range, scaling will ensure the corresponding driven bone moves 30% of its full range, with the minimum position at 0% and maximum position at 100%.

Range-of-motion matching improves upon displacement from rest by ensuring retargeted motion respects the difference in range-of-motion between the source and target, figures 5.5 - 5.8. Range-of-motion matching uses the bone limits specified in section 4.1.5 and appendix B to effectively clamp the model's movements ensuring no mesh tearing or mesh interpenetration.

However, some issues are still present. If the source performance does not represent the source actor's full range-of-motion then it is possible that the retargeted animation will appear exaggerated and/or incorrect. This can be mitigated by ensuring the source performance contains a range-of-motion motion capture take. This dissertation does

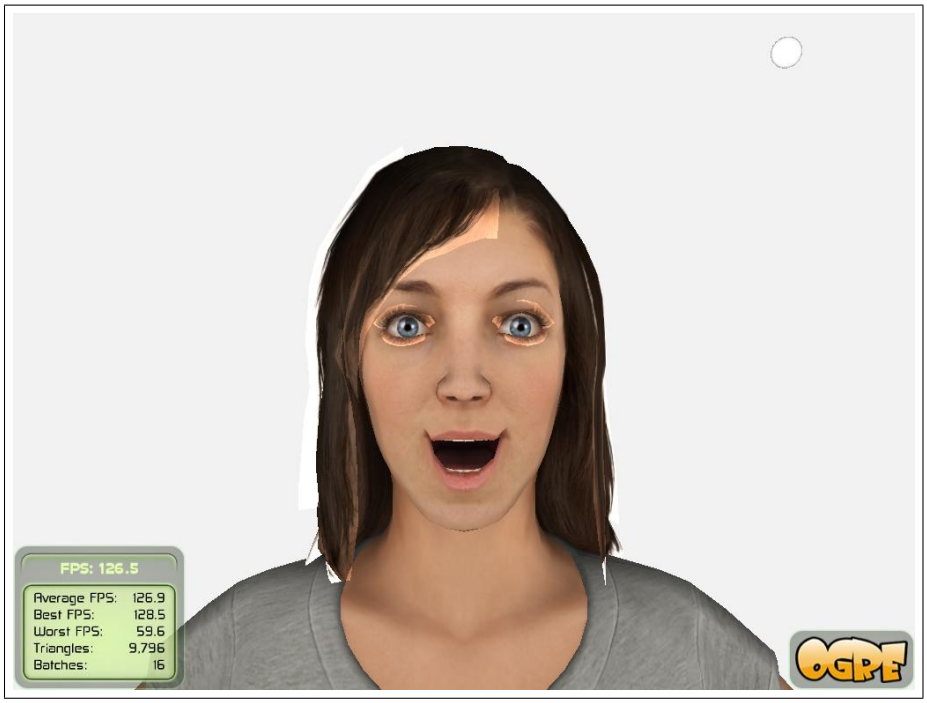

Figure 5.4: Example displacement mapping (2)

present an outline for such a session (appendix C), however, a test session was not recorded due to timing constraints.

An additional issue was noisy movement. Noisy movement presents as occasional facial ticks on the model's facial features. This can be due to noisy data or rounding in range-of-motion movement. Again, this could be reduced by capturing a full source range-of-motion, ensuring adequate signal filtering and care taken in specifying the target's range-of-motion.

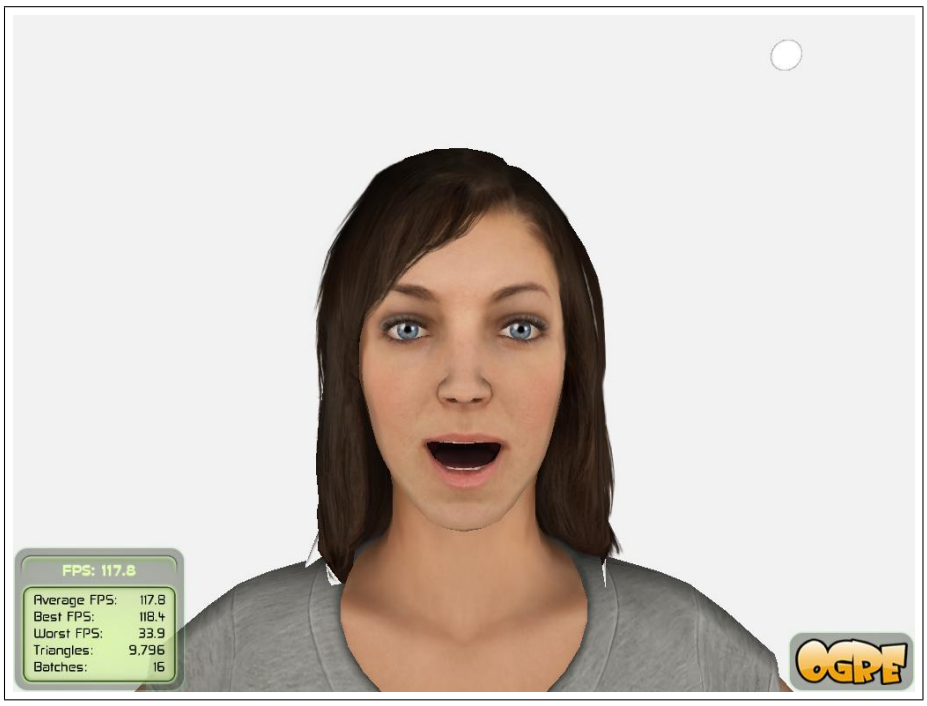

Figure 5.5: Example Range-of-Motion matching (1)

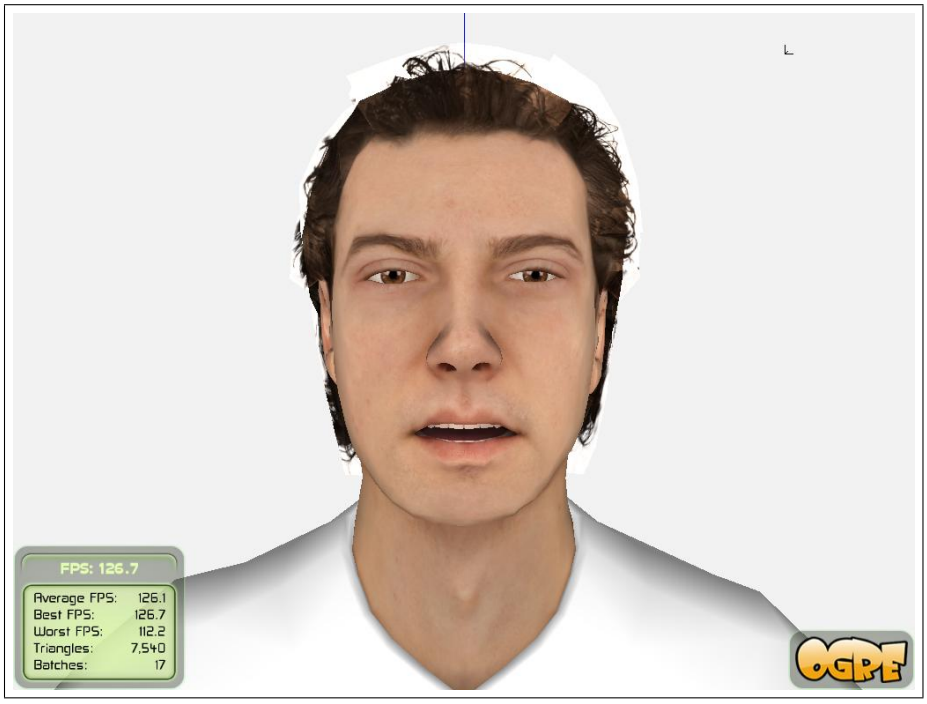

Figure 5.6: Example Range-of-Motion matching (2)

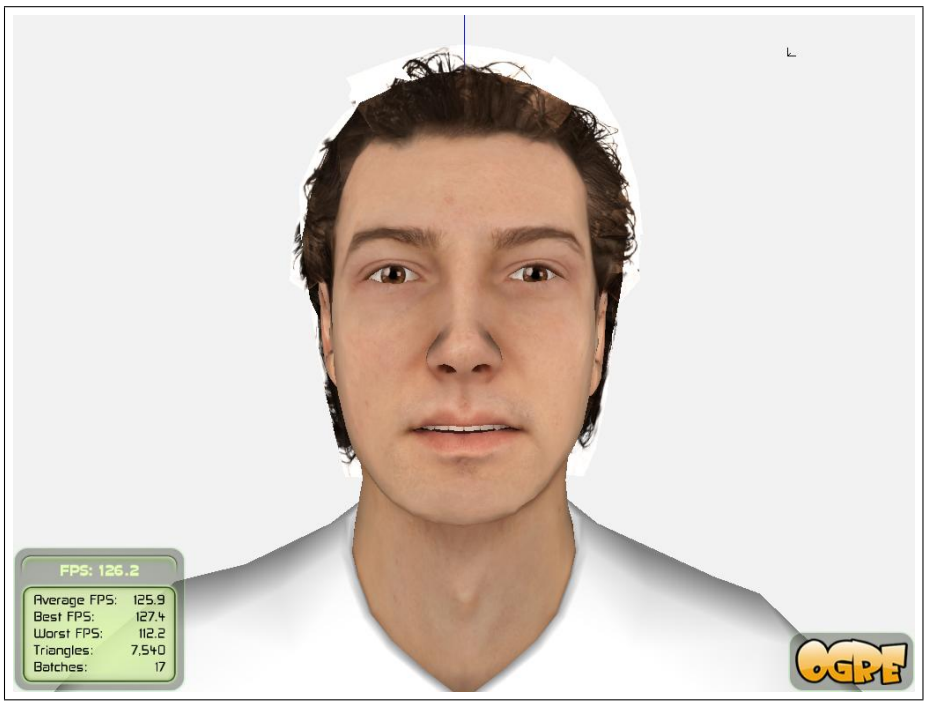

Figure 5.7: Example Range-of-Motion matching (3)

Figure 5.8: Example Range-of-Motion matching (4)

Chapter 6

Conclusion

The goal of this dissertation was to find a solution for retargeting facial performance capture using 3D bone-deformable models. This implementation of a retargeting application has achieved this goal. This implementation presents a solution for transferring an actor's facial performance to a 3D model.

Unlike current industry approaches to retargeting, this implementation retargets directly from performance data and does not require an intermediate model (virtual representation of source performer). This implementation has been shown to work with two different 3D models and perform similarly on both. This implementation presents a solution for retargeting and is a good first stage for further facial animation. For bone-deformable models, facial retargeting is often done manually, using the source model and performance for reference. This application introduces automation.

However, there are some limitations to this implementation. There is a large amount of pre-processing required. While many of these pre-processing stages are already present in a performance capture pipeline, the process of specifying the target model's range-of-motion (4.1.5) can be a potential bottleneck. The process may require some training and is time consuming, however if done with care, need only be performed once for that model.

6.1 Further Work

This section will discuss possible extensions to the retargeting application.

6.1.1 Many-to-One Mapping

The current implementation allows for only one marker to drive one bone. While this set-up is acceptable for markers that have a direct and obvious corresponding bone, it can present issues where there is no clear mapping. For example, the suggested marker placement (figure 3.2) presents three markers for the eyebrows, however the 3D models only possess two, therefore one marker was omitted.

There is a similar situation where the top eyelid marker was moved off to one side (to facilitate better capture) meaning this marker does not line up with the model's top eye lid bone.

With a many-to-one mapping it would be possible to allow multiple markers to drive the model's bones and additionally allow for weighted movement. This could present as a more accurate retargeting application.

6.1.2 Exaggeration

Exaggerated motion can currently be handled by the retargeting application, if the user specifies an exaggerated range-of-motion. However further steps would need to be taken to ensure the retargeted animation appears smooth. Exaggerated motion would allow for the user to emphasise certain areas of the source performance as well as impart personal style on to the target character.

6.1.3 Stylised Characters

Stylised characters are models of non-realistic proportion, such as a 'cartoon' character. It is believed that the current implementation can cater only for humanoid character proportions, however this theory has not been fully explored.

Additional extensions could be made to handle non-humanoid characters (animals, aliens, etc.) with largely different proportions and even differently placed and numbered facial features.

Appendix A

TRC File

Truncated for brevity

```
PathFileType	4	(X/Y/Z) C:/Users/Kenneth/Documents/
    Media/Blade Motions/Blade Database/Distinct/Actor_011/
    NT_C001_face_stab.trc
DataRate	CameraRate	NumFrames	NumMarkers
        Units	OrigDataRate	OrigDataStartFrame
    OrigNumFrames
120.000000	120.000000	5747	37	mm
    120.000000	1	5747
Frame#	Time	FaceRobot_LEar	FaceRobot_REar
                FaceRobot_RBHD
    FaceRobot_LBHD	FaceRobot_LForehead
                FaceRobot_RForehead
    FaceRobot_NoseBridge
    FaceRobot_REyebrowMid
    FaceRobot_REyebrowEnd
    FaceRobot_ROrbitalUpper
    FaceRobot_LEyebrowEnd
    FaceRobot_LOrbitalUpper
    FaceRobot_LEyebrowMid
    FaceRobot_LOrbitalLower
```

FaceRobot_LEyelidLower
FaceRobot_LEyelidUpper
FaceRobot_ROrbitalLower
FaceRobot_REyelidLower
FaceRobot_REyelidUpper
FaceRobot_LNostrilBulge
FaceRobot_RNostrilBulge
FaceRobot_LNostrilBase FaceRobot_NoseTip
FaceRobot_RNostrilBase
FaceRobot_LLipUpperBend
FaceRobot_LLipLowerBend FaceRobot_LPuffer
FaceRobot_LMouthCorner
FaceRobot_Chin FaceRobot_LJawEnd
FaceRobot_RLipLowerBend
FaceRobot_LipLower
FaceRobot_RPuffer
FaceRobot_RMouthCorner FaceRobot_LipUpper
FaceRobot_RLipUpperBend
FaceRobot_RJawEnd

X0	Y0	Z0	X1	Y1	Z1
X2	Y2	Z2	X3	Y3	
Z3	X4	Y4	Z4	X5	
Y5	Z5	X6	Y6	Z6	
X7	Y7	Z7	X8	Y8	
Z8	X9	Y9	Z9	X10	
Y10	Z10	X11	Y11	Z11	
X12	Y12	Z12	X13	Y13	
Z13	X14	Y14	Z14	X15	
Y15	Z15	X16	Y16	Z16	
X17	Y17	Z17	X18	Y18	
Z18	X19	Y19	Z19	X20	
Y20	Z20	X21	Y21	Z21	
X22	Y22	Z22	X23	Y23	

Z23 X24 Y24 Z24 X25
Y25 Z25 X26 Y26 Z26
X27 Y27 Z27 X28 Y28
Z28 X29 Y29 Z29 X30
Y30 Z30 X31 Y31 Z31
X32 Y32 Z32 X33 Y33
Z33 X34 Y34 Z34 X35
Y35 Z35 X36 Y36 Z36

69 0.575000 65.159576 66.171432
15.247816 −76.412193 47.632626 6.911851
−79.004478 128.804001 45.082470
54.275093 136.327423 51.519245 75.941559
38.061611 84.781288 −83.445587
27.548832 78.596481 0.970357 −24.715738
27.590555 −8.233782 −27.019285
44.478756 −55.896584 −12.419032 44.910355
−27.735384 −26.665453 50.921490
51.147614 −3.921833 52.508968 29.199774
−20.229677 52.860306 9.546706
−24.999146 45.266834 52.284695 2.775834
4.464105 30.247187 −7.736007
15.985188 41.206886 −9.500063 27.747540
−47.909496 −9.882837 0.160335
−29.360344 −13.564362 13.675882 −42.462852
−14.583048 24.803570 15.914208
−23.474262 −8.087914 −11.492588 −24.297987
−9.297639 25.104740 −11.080055
−15.964125 3.769651 −44.088474 −4.035847
−20.335209 −16.140812 −15.149405
17.849983 −15.277853 −34.007057 17.233952
−10.720400 −51.508080 42.149166
6.482348 −38.048141 28.661116 −1.952248
−41.722755 2.370543 −12.838081

−66.009285 12.886061 −3.606514 −78.276581
−13.453781 −16.271017 −49.038193
2.710675 −17.342730 −54.066006 −39.782635
−4.319119 −38.398796 −25.652632
−7.319777 −42.563065 3.551636 −24.251684
−30.088280 −12.676408 −20.432964
−31.695992 −8.327857 −5.181708 −79.523407
70 0.583333 65.175613 66.161156
15.264001 −76.405785 47.610046 6.909215
−79.001854 128.822540 45.094799
54.242695 136.347214 51.520615 75.955048
38.064037 84.776794 −83.419792
27.567476 78.584625 0.938395 −24.742249
27.579655 −8.221154 −27.034595
44.496651 −55.853851 −12.413730 44.910213
−27.739326 −26.649300 50.965534
51.129105 −3.919972 52.534393 29.227583
−20.247837 52.903664 9.549558
−24.981354 45.248436 52.268867 2.768425
4.480018 29.591827 −7.737854
16.208824 40.867828 −9.864295 27.456799
−47.882603 −9.889220 0.192612
−28.879553 −13.618629 13.656019 −42.156322
−14.825613 24.596436 15.956636
−23.498875 −8.059309 −11.792437 −24.479710
−9.216128 25.145437 −11.097833
−15.986203 3.795683 −44.075558 −4.013795
−20.354599 −16.161209 −15.144383
18.245113 −15.436312 −33.786003 17.272366
−10.724825 −51.448883 42.197533
6.487683 −38.026585 28.682522 −1.884929
−41.678383 2.397903 −12.866178
−66.007683 12.916308 −3.607146 −78.291641

−13.412254 −15.439973 −48.977810
2.726758 −17.171577 −54.023342 −39.764339
−4.323125 −38.359692 −25.644598
−7.295519 −42.524879 3.568994 −24.227646
−30.055849 −12.771162 −20.487753
−31.659710 −8.316338 −5.189254 −79.526871
71 0.591667 65.181602 66.158516
15.266777 −76.406456 47.600159 6.898308
−79.019524 128.805679 45.102482
54.278175 136.321564 51.529922 75.928535
38.092472 84.759613 −83.425285
27.581642 78.595734 0.947296 −24.729851
27.576881 −8.243583 −27.015415
44.526043 −55.896664 −12.500847 44.936764
−27.709093 −26.628563 50.971474
51.113811 −3.932818 52.525814 29.210939
−20.250267 52.916359 9.545291
−24.986374 45.255932 52.208870 2.731902
4.472628 28.828150 −7.637617
16.099911 40.234291 −10.324544 26.839798
−47.842197 −9.926949 0.186543
−28.372131 −13.494765 13.909547 −41.740929
−15.350941 24.028603 15.976239
−23.512466 −8.060114 −11.504373 −24.315849
−9.284396 25.122972 −11.093363
−15.959808 3.781919 −44.076572 −4.065672
−20.344015 −16.160822 −15.146696
18.246716 −15.385559 −33.733089 17.259098
−10.694113 −51.414139 42.189125
6.528830 −38.019127 28.665339 −1.821316
−41.620518 2.387192 −12.893726
−66.016762 12.905455 −3.609861 −78.286194
−13.466356 −15.598643 −48.936272

2.740660 −17.278048 −54.025784 −39.793350
−4.283241 −38.388016 −25.660337
−7.244686 −42.461117 3.557836 −24.211308
−30.051443 −12.849905 −20.470863
−31.686386 −8.310934 −5.213950 −79.524933
72 0.600000 65.147064 66.085281
15.238383 −76.150963 47.849556 6.965754
−79.094955 128.756775 45.071617
54.164555 136.343781 51.478710 75.943489
38.066120 84.757797 −83.447166
27.509970 78.616150 0.922314 −24.781300
27.601288 −8.251587 −27.045355
44.557922 −55.900978 −12.579547 44.934158
−27.707449 −26.651169 50.942852
51.065639 −3.956595 52.499340 29.215933
−20.305405 52.906292 9.517418
−25.024948 45.287971 52.130688 2.676298
4.499003 28.578207 −7.506136
16.407000 39.710125 −10.494969 26.098196
−47.790329 −10.023494 0.242722
−27.926001 −13.401171 13.965111 −41.120415
−15.431769 23.719435 16.019365
−23.565531 −8.024839 −11.569066 −24.411888
−9.238118 25.080254 −11.148741
−15.933424 3.774371 −44.113178 −4.019243
−20.383862 −16.258020 −15.127788
18.236740 −15.409441 −33.675014 17.254765
−10.760630 −51.361912 42.177345
6.482991 −37.991402 28.622181 −1.823055
−41.547390 2.373251 −13.016329
−66.010742 12.903170 −3.708434 −78.276527
−13.500454 −15.651444 −48.869656
2.745224 −17.508661 −54.054493 −39.767151

$$
\begin{array}{cccc}
-4.340452 & -38.368904 & -25.696848 & \\
-7.301236 & -42.401859 & 3.568286 & -24.279381 \\
-30.009624 & -12.629462 & -20.395895 & \\
-31.752630 & -8.323460 & -5.343803 & -79.525337
\end{array}
$$

Appendix B

Mapping File

```
BoneNames          MarkerNames          Positional          Rotational
          minX      maxX      minY      maxY      minZ      maxZ
ROuterEyebrow      FaceRobot_REyebrowEnd      true      false
    -0.050417          -0.0485669          1.69679  1.70139  0.10863
    0.10978
RInnerEyebrow      FaceRobot_REyebrowMid      true      false
    -0.0278134         -0.0251135          1.69831  1.70676  0.119968
              0.122568
REyeBlinkBottom FaceRobot_REyelidLower     true      false
    -0.0341108         -0.0341108          1.67368  1.68658  0.110327
              0.110327
REyeBlinkTop       FaceRobot_REyelidUpper     true      false
    -0.0333584         -0.0333584          1.68446  1.69301  0.109511
              0.109511
LOuterEyebrow      FaceRobot_LEyebrowEnd      true      false
    0.0470669          0.0507669           1.69509  1.70259  0.10863
    0.10978
LInnerEyebrow      FaceRobot_LEyebrowMid      true      false
    0.0236134          0.0289134           1.69846  1.70626  0.119968
              0.122568
```

```
LEyeBlinkBottom FaceRobot_LEyelidLower true false
   0.0341108       0.0341108       1.67368 1.68958 0.110327
         0.110327
LEyeBlinkTop    FaceRobot_LEyelidUpper true false
   0.0333583       0.0333583       1.68446 1.69301 0.109511
         0.109511
MMiddleEyebrow  FaceRobot_NoseBridge    true false
   -0.00050015     0.000699855     1.69392 1.70287 0.126331
         0.126331
MNose                   FaceRobot_NoseTip               true
      false   -0.000500143    0.000699855     1.65275 1.65855
    0.141839        0.141839
MUpperLip               FaceRobot_LipUpper              true
      false   -0.00210015     0.00219985      1.6178  1.6222
     0.129029        0.135479
MBottomLip              FaceRobot_LipLower              true
      false   -0.0019501      0.00269989      1.60062 1.60917
    0.129596        0.136496
MJaw                    FaceRobot_Chin                  false
     true    0.0     0.0     0.0     0.0     0.0     0.0
RCheek                  FaceRobot_ROrbitalLower true    false
      -0.051924       -0.036874       1.64949 1.66564 0.110384
         0.124737
RMouthCorner    FaceRobot_RMouthCorner  true    false
   -0.0296512      -0.0248512      1.60667 1.61632 0.115975
         0.124325
RUpperlip               FaceRobot_RLipUpperBend true    false
      -0.0181059      -0.0112059      1.61651 1.62171 0.128039
         0.135189
RMouthBottom    FaceRobot_RLipLowerBend true    false
   -0.0169976      -0.0107476      1.60454 1.61139 0.126647
         0.135524
```

```
LCheek                  FaceRobot_LOrbitalLower true    false
      0.035824        0.047774        1.65104 1.66279 0.110834
           0.121384
LMouthCorner    FaceRobot_LMouthCorner  true    false
   0.0241012       0.0313012       1.60687 1.61797 0.115775
           0.124475
LUpperlip               FaceRobot_LLipUpperBend true    false
      0.00990596      0.019006        1.61556 1.62336 0.126789
           0.136089
LMouthBottom    FaceRobot_LLipLowerBend true    false
   0.00930111      0.0183511       1.60469 1.61159 0.127024
           0.134874
```

Appendix C

Performance Range-of-Motion

Range of Motion - Face

Approximation

- Neutral
- Happy
- Sad
- Angry
- Scared
- Pain Grimace
- Surprised
- Confused
- Attentive
- Dismissive

Specific

- Mouth
  - Wide as possible
  - Purse lips
  - Big smile
  - Big frown
  - Bear teeth (top, bottom, both)
  - Roll Jaw

- Nose
  - Scrunch up

- Eye lids
  - Open eyes wide
  - Slow blink
  - Slow wink (both eyes)
  - Squint
  - Scrunch up (think pain grimace)

- Eyebrows
  - High as possible
  - Low as possible
  - Alternate (as best you can)
  - Roll eyebrows

Figure C.1: Performance Range-of-motion

Bibliography

[1] Autodesk, F. R. http://softimage.wiki.softimage.com/. Internet, August 2012.

[2] Autodesk3DStudioMax. http://usa.autodesk.com/3ds-max/, August 2012.

[3] Baek, S., Lee, S., and Kim, G. J. Motion retargeting and evaluation for vr-based training of free motions. *The Visual Computer 19* (2003), 222–242. 10.1007/s00371-003-0194-2.

[4] Baran, I., Vlasic, D., Grinspun, E., and Popović, J. Semantic deformation transfer. *ACM Trans. Graph. 28*, 3 (July 2009), 36:1–36:6.

[5] Beeler, T., Hahn, F., Bradley, D., Bickel, B., Beardsley, P., Gotsman, C., Sumner, R. W., and Gross, M. High-quality passive facial performance capture using anchor frames. *ACM Trans. Graph. 30*, 4 (July 2011), 75:1–75:10.

[6] Bickel, B., Botsch, M., Angst, R., Matusik, W., Otaduy, M., Pfister, H., and Gross, M. Multi-scale capture of facial geometry and motion. *ACM Trans. Graph. 26*, 3 (July 2007).

[7] Bickel, B., Lang, M., Botsch, M., Otaduy, M. A., and Gross, M. Pose-space animation and transfer of facial details. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2008), SCA '08, Eurographics Association, pp. 57–66.

[8] Brunet, P., Correia, N., (editors, G. B., Costa, V., Orvalho, T., Zacur, E., and Susin, A. Transferring facial expressions to different face models.

[9] Butterworth, S. On the theory of filter amplifiers. *Experimental Wireless and the Wireless Engineer 7* (1930), 536–541.

[10] Captavatar. http://gv2.cs.tcd.ie/captavatar/. Internet, August 2012.

[11] Choe, B., Lee, H., and seok Ko, H. Performance-driven muscle-based facial animation. *The Journal of Visualization and Computer Animation 12* (2001), 67–79.

[12] Choi, K.-J., and Ko, H.-S. On-line motion retargetting. In *Proceedings of the 7th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 1999), PG '99, IEEE Computer Society, pp. 32–.

[13] Chuang, E., and Bregler, C. Performance driven facial animation using blendshape interpolation. Tech. rep., 2002.

[14] Curio, C., Breidt, M., Kleiner, M., Vuong, Q. C., Giese, M. A., and Bülthoff, H. H. Semantic 3d motion retargeting for facial animation. In *Proceedings of the 3rd symposium on Applied perception in graphics and visualization* (New York, NY, USA, 2006), APGV '06, ACM, pp. 77–84.

[15] Dariush, B., Gienger, M., Arumbakkam, A., Goerick, C., Zhu, Y., and Fujimura, K. Online and markerless motion retargeting with kinematic constraints. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on* (sept. 2008), pp. 191 –198.

[16] Deng, Z., and Noh, J. Chapter 1 computer facial animation: A survey.

[17] Ekman, P., and Friesen, W. V. Facial action coding system: A technique for the measurements of facial movement, 1978.

[18] Gleicher, M. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 33–42.

[19] GV2. http://gv2.cs.tcd.ie/. Internet, August 2012.

[20] Hecker, C., Raabe, B., Enslow, R. W., DeWeese, J., Maynard, J., and van Prooijen, K. Real-time motion retargeting to highly varied user-created morphologies. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 27:1–27:11.

[21] Hsieh, M.-K., Chen, B.-Y., and Ouhyoung, M. Motion retargeting and transition in different articulated figures. In *Computer Aided Design and Computer Graphics, 2005. Ninth International Conference on* (dec. 2005), p. 6 pp.

[22] Johansson, G. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics 14*, 2 (1973), 201–211.

[23] Kholgade, N., Matthews, I., and Sheikh, Y. Content retargeting using parameter-parallel facial layers. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, ACM, pp. 195–204.

[24] Lewis, J. P., and Pighin, F. Performance-driven facial animation. SIGGRAPH Course Notes, July 2006.

[25] Ma, X., Le, B. H., and Deng, Z. Style learning and transferring for facial animation editing. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 123–132.

[26] Miranda, J. C., Alvarez, X., Orvalho, J. a., Gutierrez, D., Sousa, A. A., and Orvalho, V. Sketch express: facial expressions made easy. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling* (New York, NY, USA, 2011), SBIM '11, ACM, pp. 87–94.

[27] Moeslund, T. B., and Granum, E. A survey of computer vision-based human motion capture, 1999.

[28] Na, K., and Jung, M. Hierarchical retargetting of fine facial motions. *Computer Graphics Forum 23*, 3 (2004), 687–695.

[29] Ogre3D. http://www.ogre3d.org/. Internet, August 2012.

[30] Pighin, F., Hecker, J., Lischinski, D., Szeliski, R., and Salesin, D. H. Synthesizing realistic facial expressions from photographs. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 75–84.

[31] Poirier, M., and Paquette, E. Rig retargeting for 3d animation. In *Proceedings of Graphics Interface 2009* (Toronto, Ont., Canada, Canada, 2009), GI '09, Canadian Information Processing Society, pp. 103–110.

[32] Savenko, A., and Clapworthy, G. Using motion analysis techniques for motion retargeting. In *Information Visualisation, 2002. Proceedings. Sixth International Conference on* (2002), pp. 110 – 115.

[33] sbastien Monzani, J., Baerlocher, P., Boulic, R., and Thalmann, D. Using an intermediate skeleton and inverse kinematics for motion retargeting. In *Proc. Eurographics* (2000), Blackwell Publishers.

[34] Song, J., Choi, B., Seol, Y., and Noh, J. Characteristic facial retargeting. *Comput. Animat. Virtual Worlds 22*, 2-3 (Apr. 2011), 187–194.

[35] Tak, S., and Ko, H.-S. A physically-based motion retargeting filter. *ACM Trans. Graph. 24*, 1 (Jan. 2005), 98–117.

[36] ViconBlade. http://www.vicon.com/products/blade.html. Internet, August 2012.

[37] Vlasic, D., Brand, M., Pfister, H., and Popović, J. Face transfer with multilinear models. *ACM Trans. Graph. 24*, 3 (July 2005), 426–433.

[38] Weise, T., Bouaziz, S., Li, H., and Pauly, M. Realtime performance-based facial animation. *ACM Trans. Graph. 30*, 4 (July 2011), 77:1–77:10.

[39] Weise, T., Li, H., Van Gool, L., and Pauly, M. Face/off: live facial puppetry. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 7–16.

[40] Yamane, K., Ariki, Y., and Hodgins, J. Animating non-humanoid characters with human motion data. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2010), SCA '10, Eurographics Association, pp. 169–178.