Constraint-orientierte und personalisierbare generische Gesichtsmodelle

Diplomarbeit

vorgelegt von
Martin Stöcker

Institut für Computervisualistik
Arbeitsgruppe Computergraphik

Competence Center
NetMedia

Betreuer: Dipl.-Inform. Johannes Strassner, ehem. Fraunhofer Institut IMK.NM
Prüfer: Prof. Dr. Stefan Müller, Universität Koblenz-Landau

August 2005

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit selbstständig und nur mit angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Diese Arbeit wurde in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Koblenz, den 01.08.2005

Martin Stöcker

Danksagung

Hiermit bedanke ich mich bei allen, die mir bei der Anfertigung dieser Arbeit geholfen haben.
Besonderer Dank geht an Johannes Strassner für seine kompetente Betreuung. Weiterhin danke ich Roland Kuck und Frank Hülsken für viele informative Gespräche sowie Jörg Unterberg und Marion Langer für die Erstellung der 3D Scans. Bedanken möchte ich mich auch bei Professor Dr. Stefan Müller der Universität Koblenz für zahlreiche Ratschläge und Anregungen.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Die Animation virtueller Charaktere ist aus heutigen Kinofilmen, Computerspielen und interaktiven Lernumgebungen nicht mehr wegzudenken. Virtuelle Schauspieler übernehmen Rollen in realen Szenen und interagieren mit realen Persönlichkeiten. Die Animationsindustrie entwickelt computeranimierte Filme, in denen Erscheinung, Bewegungen und Gesichtsausdrücke der Charaktere realistisch imitiert werden können.
Die Animation des menschlichen Gesichts ist interessant und eine Herausforderung aufgrund seiner Vertrautheit. Wir können ein bestimmtes Gesicht aus einer Vielzahl ähnlicher Gesichter identifizieren und die kleinsten Veränderungen im Gesichtsausdruck erkennen. Diese Eigenschaften unterstützen die Kommunikation, in der Emotionen aus dem Gesichtsausdruck und aus der Gestik unseres Gegenübers abgelesen werden können. Aus diesem Grund sind personalisierbare Gesichtsmodelle in der Gesichtsanimation notwendig, die menschliche Gesichtszüge und ein glaubwürdiges emotionales Verhalten imitieren können. Die Animation der Gesichtsmodelle basiert auf fundamentalen Animationstechniken, deren Ziel die zeitliche Manipulation der Gesichtsoberfläche zur Darstellung eines gewünschten Gesichtsausdrucks ist. Der Nachteil von Animationstechniken wie Keyframing ist ein additives Verhalten, durch das ähnliche Gesichtsausdrücke gemischt werden können. Abbildung 1.1 zeigt ein Beispiel der Additivität anhand der Vermischung eines überraschten Gesichtsausdrucks und eines Gesichtsausdrucks mit geöffnetem Mund. In diesem Fall addieren sich ähnliche Mundkonfigurationen. Die entstehenden gemischten Gesichtsausdrücke verlangen eine Definition von Grenzen, um unrealistische Gesichtsausdrücke zu vermeiden. Ein Anwendungsgebiet für diese Art von Constraints ist die Integration von Animationsdaten aus verschiedenen Quellen. Dadurch können z.B. durch aufgenommene Audiodaten erzeugte Sprachanimationen mit Animationen aus Bewegungsdaten (Tracking) gemischt werden, ohne dass eine Anpassung der Animationsdaten nötig ist.

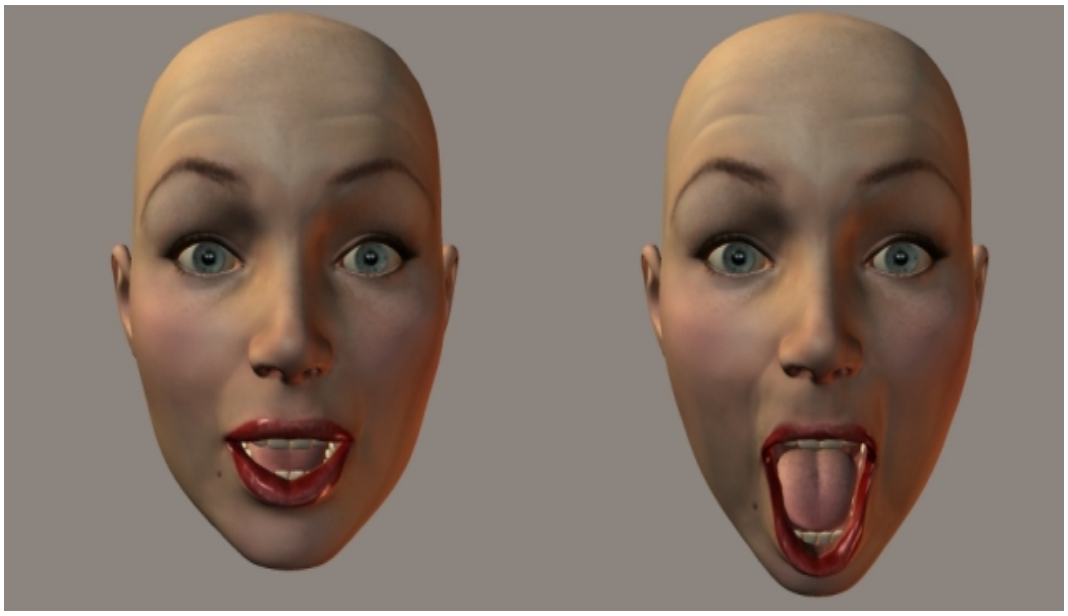

Abbildung 1.1: Vermischung zweier Gesichtsausdrücke: Überraschter Gesichtsausdruck (links), überraschter Gesichtsausdruck und addiertes A-Visem (rechts)

1.2 Ziele der Arbeit

Diese Arbeit soll sich mit der Personalisierbarkeit von Gesichtsmodellen zur Erzeugung realistischer Animationen und der Entwicklung dynamischer Grenzen während der Animation auseinandersetzen. Die grundlegende Animationstechnik des Systems ist Keyframing (Morphing).
Das Ziel dieser Arbeit ist die Entwicklung eines zweischichtigen Systems, das aus Informationen realer Personen Animationen generiert (1. Ebene), die durch ein Constraintsystem in realistischen Grenzen gehalten werden (2. Ebene).
Die 1. Ebene ist für die Erzeugung und Analyse personalisierbarer 3D Scans und generischer Modelle zuständig. Die Verarbeitung der Modelle liegt in der Registrierung, der Bestimmung von Korrespondenzen untereinander sowie deren Übertragung auf generische Modelle, um daraus Animationen durch realistische Morphtargets erzeugen zu können.
Die 2. Ebene behandelt die Integration von Constraints in den Animationsprozess. Die Entwicklung des Constraintsystems basiert in dieser Arbeit auf folgender Definition:
Constraints sind dynamische Grenzen, die den additiven Animationsprozess in der Gesichtsanimation auf die ausschließliche Darstellung realistischer Gesichtsausdrücke beschränkt.

1.3 Aufbau der Arbeit

Die Arbeit ist in sechs Kapitel gegliedert. Nach dem ersten einleitenden Kapitel werden im zweiten Kapitel die Grundlagen bezüglich Bildverarbeitung und Computergrafik vermittelt. Weiterhin werden wichtige Informationen aus der Gesichtsanimation sowie Möglichkeiten zur Realisierung von Constraints in der Animation beschrieben. Aus einer Beschreibung und Analyse verschiedener, relevanter Lösungsmöglichkeiten der Teilprobleme wird im dritten Kapitel das Konzept zur Erstellung personalisierbarer Animationen und eines Constraintsystems entwickelt. Auf Basis der gefällten konzeptionellen Entscheidungen wird in der Realisierung im vierten Kapitel die Umsetzung der Verfahren beschrieben. Das fünfte Kapitel zeigt und beschreibt die Ergebnisse der realisierten Module. Das letzte Kapitel fasst die Ergebnisse dieser Arbeit zusammen, bildet daraus ein Fazit und gibt abschließend einen Ausblick auf mögliche Optimierungen und Weiterentwicklungen.

Kapitel 2

Theoretische Grundlagen

Dieses Kapitel gliedert sich in vier Abschnitte. Zunächst werden die wichtigsten mathematischen Grundlagen und Verfahren der Bildverarbeitung und der Computergrafik vermittelt, die für das Verständnis folgender Kapitel nötig sind. Im dritten Abschnitt befindet sich ein Überblick über die Animationstechniken in der Gesichtsanimation. Der letzte Abschnitt vermittelt generelle Methoden zur Realisierung von Constraints in der Animation.

2.1 Grundlagen der Bildverarbeitung

2.1.1 Bildpyramiden

Eine Bildpyramide ist eine Darstellung von Bildern in verschiedenen Auflösungen. Zur Konstruktion einer Pyramide können unterschiedliche Filtertechniken verwendet werden. Die Gauß Pyramide zerlegt das originale Bild in eine bestimmte Menge Tiefpass gefilterter Bilder. Eine Laplace Pyramide erstellt Frequenzbilder, während die orientierte Laplace Pyramide die Laplace Pyramide in eine Menge von Orientierungen zerlegt. Eine kontinuierliche Reduktion der Bildauflösung eines originalen Bildes wird durch Filtern mit einer Gauß Maske und gleichzeitiges Downsampling erreicht. Durch dieses aufeinander folgende Glätten des originalen Bildes entstehen mehrere Tiefpass gefilterte Bilder.
Wie die Anwendung von Glättungsfiltern gehört auch die Verwendung von Ableitungen zu den fundamentalen Operationen auf Bildern. Da ein Bild keine kontinuierliche Funktion, sondern eine diskrete Funktion der Bildkoordinaten ist, werden mathematische Ableitungen durch Filtertechniken approximiert. Die Laplace Pyramide wird durch eine Gauß Pyramide konstruiert, indem Differenzen zwischen aufeinander folgenden Ebenen berechnet werden. Die Differenz zweier Level der Gauß Pyramide entspricht einem Hochpass gefilterten Bild.

2.1.1.1 Gauß Pyramiden

In [1] und [3] werden die Pyramidenoperationen für Gauß Pyramiden mathematisch bestimmt. Eine Operation faltet ein Eingabebild mit einem Gauß Filter, dezimiert gleichzeitig das Bild und erstellt ein Ausgabebild G_{l+1} mit einem Viertel der Größe des Eingabebildes G_l. Aus G_l mit $w \times h$ wird die nächste Ebene der Pyramide G_{l+1} durch

$$G_{l+1}(x,y) = \sum_m \sum_n w(m,n) G_l(2x-m, 2y-n)$$

mit

$$x \in \{0, ..., \lfloor \frac{(w-1)}{2} \rfloor\}, \quad y \in \{0, ..., \lfloor \frac{(h-1)}{2} \rfloor\}$$

berechnet, wobei die Summe unter der Gauß Maske $w(m,n)$ berechnet wird. Abb. 2.1 zeigt das Prinzip der Gauß Pyramide.

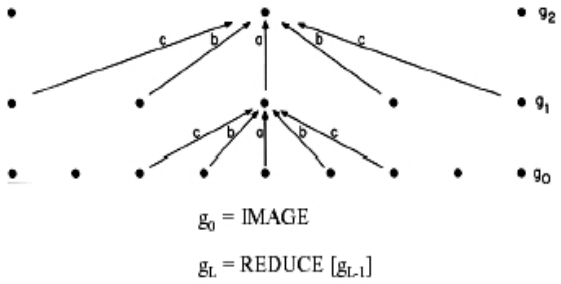

Abbildung 2.1: Berechnung einer Gauß Pyramide. Jeder Punkt repräsentiert einen Grauwert eines Pixels. Jede höhere Ebene erhält einen gewichteten Mittelwert des Grauwerts der niederigeren Ebene [1]

Aufgrund der Separierbarkeit des Gauß Filters kann die Faltung durch eine horizontale Filtermaske berechnet werden: $w(m,n) = w_x(m) w_y(n)$. Die entsprechende Rückrichtung berechnet ein interpoliertes Bild G_{l-1}, das viermal so groß ist wie das Eingabebild G_l mit den Maßen $w \times h$:

$$G_{l-1}(x,y) = \sum_m \sum_n w(m,n) G_l(\frac{x-m}{2}, \frac{y-n}{2})$$

mit

$$x \in \{0, ..., 2w\}, \quad y \in \{0, ..., 2h\}$$

wobei die Summe nur dann berechnet wird, wenn $\frac{x-m}{2}$ und $\frac{y-n}{2}$ ganzzahlige Werte sind. Abb. 2.2 zeigt eine Gauß Pyramide mit vier Ebenen.

Abbildung 2.2: Gauß Pyramide

2.1.1.2 Laplace Pyramiden

Die Laplace Pyramide ist eine Zerlegung des originalen Bildes in eine Hierarchie von Bildern, wobei jede Ebene einem unterschiedlichen Frequenzbereich des Bildes entspricht. Die Laplace Pyramide wurde von Burt und Adelson [1, 2] zur Bildkompression entwickelt. Das mittels Gauß Filter und Subsampling erstellte Bild G_1 dient als Vorhersage der Pixelwerte des Originalbildes G_0. Für eine komprimierte Repräsentierung des Bildes wird ein Fehlerbild konstruiert, das sich aus der Subtraktion von G_1 und G_0 ergibt. Dieses stellt den untersten Level der Laplace Pyramide dar. Die Differenz zwischen diesen beiden Funktionen ist dem Laplace Operator gleich, der in der Bildverarbeitung zum Image Enhancement verwendet wird. Eine lineare Faltung mit dem Laplace Operator detektiert Kanten im Bild durch Hervorhebung hoher Ortsfrequenzen (Grauwertdifferenzen). Die Differenzen zur Berechnung der Laplace Pyramide sind Absolutbeträge der einzelnen Farbwerte, da negative Farbwerte enstehen können. Durch die Absolutwerte werden horizontale und vertikale Kanten im Laplace Bild abgespeichert.
Der Laplace Operator verwendet die zweite Ableitung und ist definiert als:

$$\nabla^2 g = \frac{\delta^2 g}{\delta^2 x} + \frac{\delta^2 g}{\delta^2 y} = (h_{2x} \otimes g) + (h_{2y} \otimes g),$$

mit h_{2x} und h_{2y} als Filter der zweiten Ableitung, g als das Eingabebild und $\otimes$ als der Faltungsoperator. Generell lässt sich folgende Filtermaske für die Berechnungen verwenden:

$$[h_{2x}] = [h_{2y}]^T = [1 \quad -2 \quad 1].$$

Um eine Laplace Pyramide eines Bildes G_l zu konstruieren, wird zunächst eine Gauß Pyramide erstellt. Die Sequenz der Fehlerbilder $L_0, L_1, ..., L_n$ ist

$$L_l = G_l - expand(G_{l+1}).$$

Da kein Bild G_{n+1} als Vorhersagebild für G_n vorhanden ist, gilt $L_N = G_N$. Abbildung 2.3 zeigt eine Laplace Pyramide mit vier Ebenen.

Abbildung 2.3: Laplace Pyramide mit vier Ebenen. Die ersten drei Laplace Bilder wurden zusätzlich aufgehellt.

2.1.2 Gradienten

Der Bildgradient an einem Punkt (x, y) entlang einer Richtung (u, v) ist definiert als das Verhältnis, mit dem sich die Bildintensität in Richtung (u, v) verändert. Gradienten werden in der Bildverarbeitung zur Detektion von Strukturen im Bild, wie Kanten und Ecken, eingesetzt. Ein Gradient ist immer orthogonal zur Kante, daher wird z.B. eine horizontale Kante durch den vertikalen Gradienten detektiert. h_x sei ein horizontaler Ableitungsfilter und h_y ein vertikaler Ableitungsfilter. Der Gradient $\nabla g(x, y)$ ist

$$\nabla g = \frac{\delta g}{\delta x}\vec{i}_x \frac{\delta g}{\delta y}\vec{i}_y = (h_x \otimes g)\vec{i}_x + (h_y \otimes g)\vec{i}_y,$$

mit $\vec{i_x}$ und $\vec{i_y}$ als Einheitsvektoren in horizontale und vertikale Richtung, g als das Eingabebild und $\otimes$ als der Faltungsoperator.
Die Länge eines Gradienten

$$|\nabla g| = \sqrt{(h_x \otimes g)^2 + (h_y \otimes g)^2}$$

gibt an, wie stark die Kante ist. Die Resultate dieser Berechnungen hängen von der Wahl der Ableitungsfilter (h_x, h_y) ab. Grundlegende Ableitungsfilter sind:

$$[h_x] = [h_y]^T = [1 \quad -1],$$

$$[h_x] = [h_y]^T = [1 \quad 0 \quad -1].$$

2.1.3 Optical Flow

Der optische Fluss ist die Verteilung von Beschleunigungen der Bewegung von Helligkeitsmustern zwischen zwei Bildern einer Bildsequenz. Der Algorithmus berechnet ein Beschleunigungsfeld in x- und y-Richtung, das die Verschiebung der Pixel zwischen den beiden Bildern beschreibt. Wichtige Informationen über den räumlichen Aufbau von Objekten sowie Bewegungen bzw. Veränderungen in der Bildfolge können daraus extrahiert werden. Es existieren verschiedene Algorithmen zur Schätzung des optischen Flusses. Allgemein können diese in drei Kategorien eingeteilt werden [45]:

1. gradientenbasierte,
2. korrelationsbasierte,
3. filterbasierte.

Der konventionelle Algorithmus ist gradientenbasiert und wurde von Horn und Schunck entwickelt [4]. Korrelationsbasierte Verfahren sind sehr berechnungsintensiv und benötigen hohe Auflösungen für die Extraktion des optischen Flusses. Filterbasierte sind zwar sehr stabil, dennoch sind viele Frames nötig, um den optischen Fluss zu berechnen. In den folgenden Abschnitten wird das konventionelle, gradientenbasierte Verfahren erklärt.

2.1.3.1 Gradientenbasierte Verfahren

Die Beziehung zwischen dem optischen Fluss in der Bildebene und der Bewegung des dargestellten Objekts in der dreidimensionalen Welt ist nicht offensichtlich: Eine geshadete, rotierende Kugel verändert in ihrer Bewegung das Shading nicht. Daher wäre der optische Fluss gleich Null für alle Punkte in der Bildebene. Horn und Schunck nehmen eine eingeschränkte Problemdomäne an: Die im Bild dargestellte Oberfäche ist flach und die einfallende Beleuchtung ist einheitlich über die gesamte Fläche verteilt. Die Helligkeit in einem Pixel ist dann proportional zur Reflexion des korrespondierenden Punktes auf der Oberfläche des Objektes. Weiterhin wird angenommen, dass sich die Lichtreflexion auf dem Objekt gleichmäßig verändert und keine räumlichen Unstetigkeiten auftreten. Situationen, in denen Objekte andere teilweise überdecken, werden in [4] nicht betrachtet, da dieser Fall Unstetigkeiten in der Reflexion bei den Objektgrenzen hervorrufen würde.

Auf Basis dieser Domäne wird eine Gleichung abgeleitet, die die Beziehung zwischen Bildhelligkeit an einem Punkt und Objektbewegung des Helligkeitsmusters darstellt. Die Bildhelligkeit an einem Punkt (x, y) in der Bildebene zur Zeit t sei $E(x, y, t)$. Wenn sich das Helligkeitsmuster bewegt, bleibt die Helligkeit eines bestimmten Punktes im Muster konstant:

$$\frac{dE}{dt} = 0.$$

Teile des Helligkeitsmusters verschieben sich mit einer Distanz δx in x-Richtung und δy in y-Richtung zur Zeit δt:

$$E(x, y, t) = E(x + \delta x, y + \delta y, t + \delta t).$$

Daraus ergibt sich

$$\frac{\delta E}{\delta x}\frac{dx}{dt} + \frac{\delta E}{\delta y}\frac{dy}{dt} + \frac{\delta E}{\delta t} = 0$$

und die Gleichung des optischen Flusses mit zwei Unbekannten $u = \frac{dx}{dt}$ sowie $v = \frac{dy}{dt}$. Diese Gleichung wird auch als Optical Flow Constraint Equation bezeichnet:

$$E_x u + E_y v + E_t = 0$$

mit E_x, E_y und E_t als partielle Ableitungen der Bildhelligkeit in Bezug auf x, y und t.
Die lokale Berechnung der Beschleunigungen des optischen Flusses u und v kann nur durch Definition einiger weiterer Einschränkungen stattfinden:
Die Beschleunigung u und v muss entlang einer Linie liegen, die senkrecht auf dem Gradientenvektor der Helligkeit (E_x, E_y) steht (aperture problem). Diese Einschränkung folgt aus der oben genannten Gleichung des optischen Flusses:

$$(E_x, E_y)(u, v) = -E_t$$

$$\Rightarrow (u, v) = -\frac{E_t}{\sqrt{(E_x)^2 + (E_y)^2}}.$$

Die Gleichung der Bildhelligkeit schränkt die Beschleunigungen des optischen Flusses ein. Abb. 2.4 verdeutlicht diesen Constraint. Daher kann die Bewegung in Richtung der Helligkeitskonturen im rechten Winkel zum Helligkeitsgradienten nicht berechnet werden.
Um eine aussagekräftige Lösung für die Gleichung des optischen Flusses zu erhalten, sind weitere Constraints nötig. Der Smoothness Constraint geht davon aus, dass auf die im Bild dargestellten Objekte eine rigide Bewegung oder Deformation wirkt. Benachbarte Punkte auf den Objekten und in der Bildebene beschleunigen daher ähnlich, sodass sich Helligkeitsmuster im gesamten Bild nur

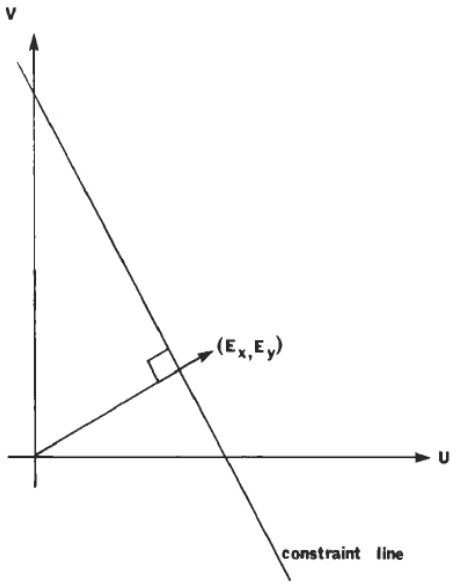

Abbildung 2.4: Das Aperture Problem aus [4]

langsam verändern. Dies kann durch eine Minimierung der quadratischen Länge der Beschleunigung des optischen Flusses ausgedrückt werden:

$$(\frac{\delta u}{\delta x})^2 + (\frac{\delta u}{\delta y})^2, \quad (\frac{\delta v}{\delta x})^2 + (\frac{\delta v}{\delta y})^2.$$

Die Gleichung des optischen Flusses kann nicht für jeden Pixel im Bild gelöst werden, da sie zwei Unbekannte enthält. Gilt jedoch die zusätzliche Annahme des Smoothness Constraints, entsteht ein überbestimmtes Gleichungssystem, das mit einer Optimierungsmethode gelöst werden kann.

2.1.3.2 Hierarchische, gradientenbasierte Verfahren

In [3, 5] wird der optische Fluss zwischen zwei Bildern mit Hilfe von Bildpyramiden berechnet. Aufgrund der verschiedenen Auflösungen kann die Gleichung des optischen Flusses auch dann berechnet werden, wenn Verschiebungen in der Szene groß sind. Dieser Ansatz beginnt damit, dass eine Gauß Pyramide der Eingabebilder berechnet wird. Durch die kontinuierliche Reduzierung der Auflösung durch Bildfilter und Subsampling werden die Verschiebungen der Pixel reduziert. Der höchste Level der Pyramide enthält aufgrund der niedrigen Auflösung zwar auch ein sehr niedrig aufgelöstes Beschleunigungsfeld, dennoch kann es als Anfangsschätzung verwendet werden. Der Analyselevel wird dekrementiert, das vom vorherigen Level berechnete Beschleunigungsfeld (v_x, v_y) wird mit Faktor

zwei upgesampelt und wird als Startschätzung für den aktuellen Level der Pyramide verwendet. Ein durch Upsamling vergrößertes Beschleunigungsfeld wird auf Level $(G_l)^0$ angewendet, indem ein Warpbild W erzeugt wird. Ein Punkt (x, y) des Ausgabebildes W wird durch das Eingabebild und das Beschleunigungsfeld berechnet:

$$(x', y') = (x + v_x(x, y), y + v_y(x, y)).$$

Falls x' und y' nicht ganzzahlig sind, wird der Grauwert an dieser Stelle bilinear interpoliert. Die Bilder W und G_l^1 werden verwendet, um ein residuales Beschleunigungsfeld zu berechnen. Um die Bewegung $v = (v_x, v_y)^T$ an einem Bildpunkt (x, y) zu berechnen wird folgender Fehler minimiert:

$$E = \sum (v_x F_x + v_y F_y - F_t)^2.$$

F_t ist die zeitliche Veränderung zwischen den zwei Frames F_1 und F_2, F_x und F_y sind Bildableitungen und v_x und v_y sind die Bewegungskomponenten in x- und y-Richtung. Die Summe des oben genannten Fehlers wird unter einer Region R mit dem Zentrum (x, y) gebildet. Daraus lässt sich ein Gleichungssystem ableiten:

$$Wv = \gamma,$$

wobei

$$W = \begin{bmatrix} \sum F_x^2 & \sum F_x F_y \\ \sum F_x F_y & \sum F_y^2 \end{bmatrix},$$

$$v = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

und

$$\gamma = \begin{bmatrix} \sum F_x F_t \\ \sum F_y F_t \end{bmatrix}.$$

Die Matrix W ist symmetrisch und positiv-semidefinit. Das bedeutet, dass sie reale, nicht-negative Eigenwerte haben muss. Die maximalen und minimalen Eigenwerte von W sind λ_{max} und λ_{min} und die entsprechenden Eigenvektoren ϕ_{max} und ϕ_{min}. Wenn $\lambda_{min} = 0$ gilt, dann ist die Matrix W singulär und reduziert sich:

$$W = \lambda_{max} \phi_{max} \phi_{max}^T.$$

Wenn jedoch $\lambda_{max} = 0$ gilt, sind alle Einträge in W Null und die Beschleunigung ist uneingeschränkt. Dies geschieht, wenn der Gradient in der Region R auch Null ist.
Falls die Eigenwerte von W nicht Null sind, kann der Beschleunigungsvektor mit

$$v = \begin{bmatrix} \sum F_x F_t \\ \sum F_y F_t \end{bmatrix}$$

berschnet werden. Die Gradienten im Bild und die temporären Gradienten an einem Punkt (x, y) werden durch einfache Subtraktionen zwischen Pixelwerten berechnet:

$$\begin{aligned} F_x(x,y) &= \frac{(F_2(x+1,y)-F_2(x-1,y))}{2.0} \\ F_y(x,y) &= \frac{(F_2(x,y+1)-F_2(x,y-1))}{2.0} \\ F_t(x,y) &= F_2(x,y) - F_1(x,y) \end{aligned} \quad .$$

Das berechnete residuale Beschleunigungsfeld wird auf das vorherige Feld addiert und aktualisiert (v_x, v_y). Bevor dieselben Schritte auf den tiefer liegenden Ebenen der Gauß Pyramide ausgeführt werden, wird das Beschleunigungsfeld mit einem Gauß Filter geglättet, um enstandene Ausreißer zu eliminieren. Wenn die höchste Auflösung verarbeitet wurde, wird zur Visualisierung des resultierenden Vektorfeldes ein Nadeldiagramm erstellt (Abb. 2.5).

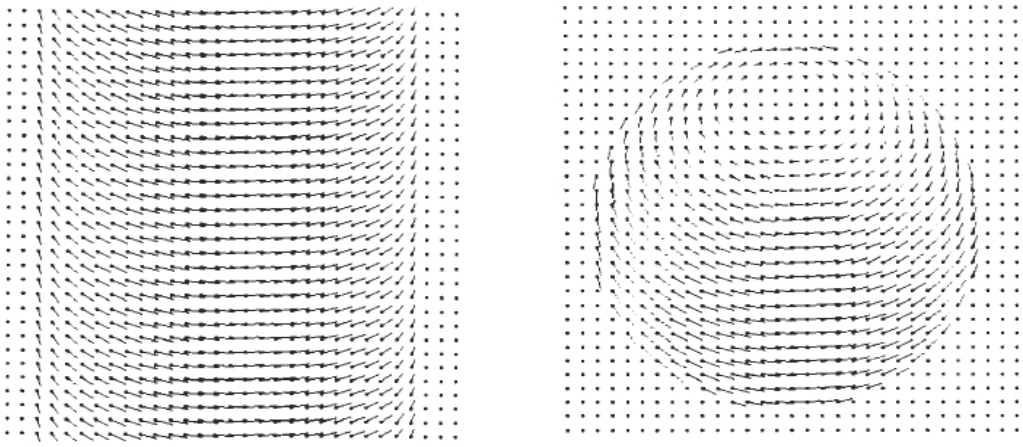

Abbildung 2.5: Nadeldiagramme eines rotierenden Zylinders und einer Kugel aus [4]

2.2 Grundlagen der Computergrafik

2.2.1 T-Meshes und zylindrische Texture Maps

T-Meshes sind triangulierte Objekte und bestehen aus grafischen Primitiven. In OpenGl Performer werden geometrische Beschreibungen von Objekten in Geosets gespeichert. Geosets sind Sammlungen grafischer Primitive wie Punkte, Linien, oder Triangle-Strips. Ein Primitivtyp beschreibt die Konnektivität von Vertex zu Vertex und definiert die Geometrie. Abb 2.6 zeigt die Beziehung zwischen Konnektivität und Primitivtyp aus . Die Kombination von Geosets erlaubt die Konstruktion komplexer 3D Objekte, die durch Manipulation der Primitive innerhalb der Geosets animiert werden können. Die Informationen zur Beschreibung eines grafischen Primitivs werden in Attributlisten gespeichert. Dazu gehören die

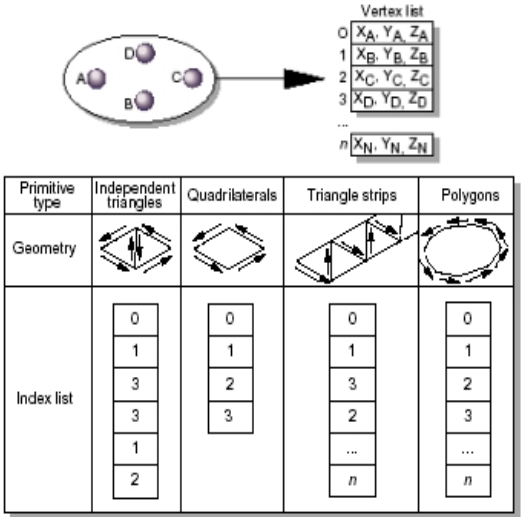

Abbildung 2.6: Konnektivität und Primitivtypen aus [6]

Vertizes, Farben, Normalen und Texturkoordinaten. Der Zugriff auf diese Daten erfolgt über Indizes der grafischen Primitive.

Kopf- und Gesichtsmodelle verwenden zylindrische Texturen. Während einer zylindrischen Projektion werden die Farbwerte eines Punktes auf dem Kopfmodell entlang eines radialen Strahls von der vertikalen Achse des Kopfes auf eine umgebene Mantelfläche eines Zylinders übertragen [7].
Eine weitere Möglichkeit zur manuellen Erzeugung zylindrischer Texturen besteht darin, Frontal- und Profilfotos einer Person zu überblenden. Abb. 2.7 zeigt eine zylindrische Texture Map eines generischen Kopfmodells.

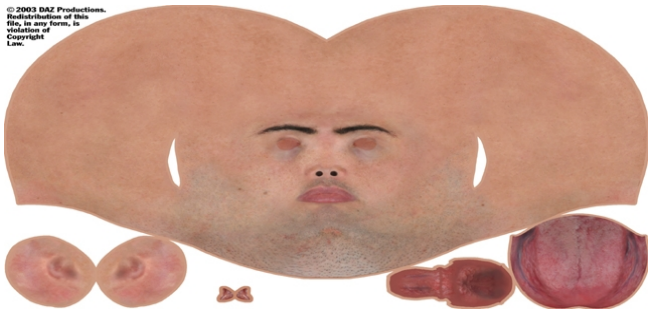

Abbildung 2.7: Zylindrische Texture Map

2.2.2 Transformationen mit Quaternionen

Die Computergrafik verwendet Quaternionen für Rotationen und Orientierungen aufgrund einer kompakten Repräsentierung im Gegensatz zu orthogonalen Ma-

trizen. Der Euler Winkel ist der klassische Weg, um Rotationen im 3D Raum zu realisieren. Euler Winkel sind anfällig gegenüber dem sogenannten *Gimbal Lock*. Rotationsmatrizen um die x, y und z Achse werden durch Matrixmultiplikationen kombiniert. Diese Operation ist nicht kommutativ, daher ist die Reihenfolge der Multiplikationen wichtig. Da die Rotationsmatrix von der Reihenfolge der Matrixmultiplikationen abhängt, kann es der Fall sein, dass die Rotation um eine Achse auf eine andere Rotationsachse gemappt wird. Durch dieses Problem kann es unmöglich werden, das Objekt um eine Achse zu rotieren. Durch Quaternionen kann das beschriebene Problem gelöst werden. Statt der Rotation eines Objekts durch eine Reihe von Rotationen, erlauben Quaternionen eine Rotation um eine beliebige Rotationsachse und einen Winkel.
Quaternionen sind erweiterte komplexe Zahlen, $w + ix + jy + kz$, mit $i^2 = j^2 = k^2 = -1$, $ij = k = -ji$. Sie sind durch Quadrupel reeler Zahlen (x, y, z, w) definiert, die als eine Kombination der drei Koordinaten der Rotationsachse durch einen Vektor v und des Rotationswinkels durch ein Skalar w dargestellt werden und für Operationen wie Addition und Multiplikation definiert sind [8]. Bei Quaternionen werden die Rotationsachsen durch Multiplikationen kombiniert und das resultierende Quaternion wird in eine Rotationsmatrix konvertiert.
Ein Einheitsquaternion $q = (x, y, z, w)$ mit $x^2 + y^2 + z^2 + w^2 = 1$ rotiert einen Vektor v durch

$$v' = qvq^{-1}.$$

Die Beziehung zwischen Quaternionen und 3D Rotationen verdeutlicht Abb. 2.8.

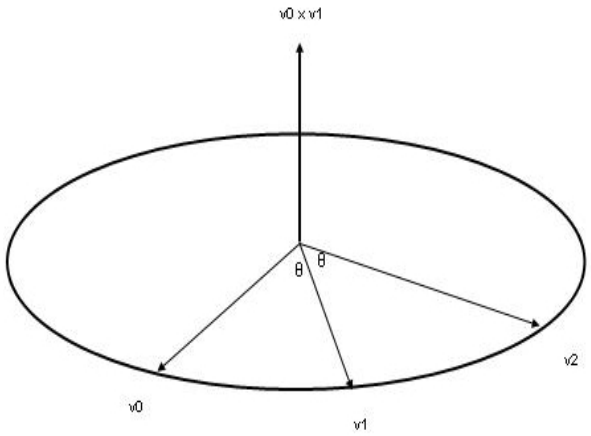

Abbildung 2.8: Rotation durch qv_0q^{-1}

v_0, v_1 und v_2 sind koplanare Einheitsvektoren, v_2 repäsentiert eine Rotation von v_0 um die Achse $v_0 \times v_1$ und Winkel 2θ. Die Rotation kann durch das Quaternion $q = (v_0 \times v_1,\ v_0 \circ v_1)$ ausgedrückt und mittels $v_2 = qv_0q^{-1}$ berechnet werden.

Eine 3×3 Rotationsmatrix R kann durch die obige Gleichung generiert werden, sodass $v_2 = Rv_0$ gilt:

$$R = \begin{bmatrix} x^2+y^2-z^2-w^2 & 2(yz-xw) & 2(yw+xz) \\ 2(yz+xw) & x^2+z^2-y^2-w^2 & 2(zw-xy) \\ 2(yw-xz) & 2(zw+xy) & x^2+w^2-y^2-z^2 \end{bmatrix}.$$

.

2.3 Grundlagen der Gesichtsanimation

Im folgenden Abschnitt werden wichtige Grundlagen der Gesichtsanimation vermittelt. Der erste Abschnitt beschäftigt sich mit der Kategorisierung und Beschreibung von Gesichtsausdrücken, die als eine Basis für die Kontrolle von Gesichtsmodellen während der Animation dienen. Der zweite Abschnitt beschreibt kurz zwei verschiedenene Arten von Gesichtsmodellen. Der dritte Abschnitt behandelt die wichtigsten Animationstechniken, die in heutigen Animationsumgebungen verwendet werden.

2.3.1 Facial Action Coding System

Das Facial Action Coding System (FACS) wurde von Ekman und Friesen zur Beschreibung und Kategorisierung von Gesichtsausdrücken entwickelt [9]. Obwohl es eigentlich nicht für den Gebrauch in der Gesichtsanimation vorgesehen war, wurde dieses deskriptive Schema weitgehend als Basis für die Kontrolle der Gesichtsausdrücke in vielen Modellen der Gesichtsanimation verwendet. Es basiert auf einer Gliederung des menschlichen Gesichts in minimale Bewegungseinheiten, den Action Units (AUs), die nicht mehr in kleinere Einheiten gegliedert werden können. Die Bewegungen des Gesichts werden von darunterliegenden Muskelkontraktionen und Muskelentspannungen einzelner Muskelstränge und Muskelgruppen generiert. FACS definiert alle *visuell unterscheidbaren* Bewegungen als eine Zuordnung $AU\text{-}ID \rightarrow Action$. Insgesamt enstanden in Ekmans Untersuchungen 46 Action Units für Gesichtsbewegungen und weitere 12 Action Units für Kopf- und Augenrotation. Beispielhafte AUs sind Inner Brow Raiser, Outer Brow Raiser oder Lid Tightener.
Der erste Schritt in der Entwicklung von FACS war die Bestimmung der Muskeln, die unabhängig voneinander bewegt werden können, und wie diese Muskeln den Gesichtsausdruck verändern. Der zweite Schritt war es herauszustellen, ob alle vereinzelten Muskeln durch Beobachtung des Gesichtsausdrucks unterschieden werden konnten. In einigen Fällen gestaltet sich dies als schwierig, daher wurden auch Action Units definiert, die das Resulat mehrerer Muskeln sind. Eine eins-zu-eins Korrespondenz zwischen Action Units und einzelnen Muskeln konnte nicht aufgebaut werden.

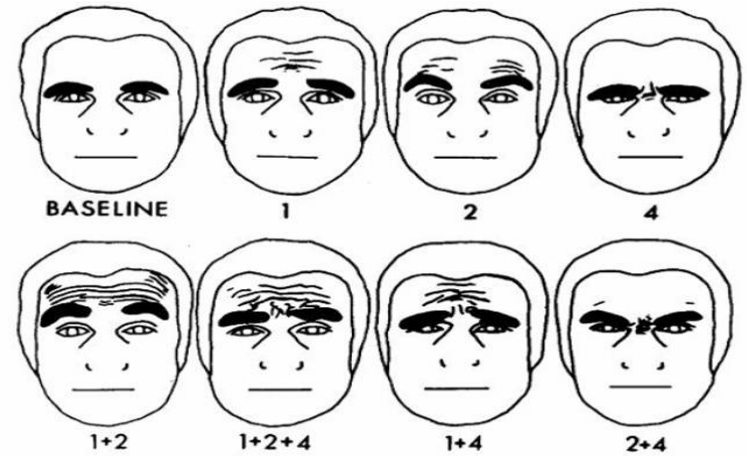

Abbildung 2.9: AUs der Augenbrauen [9]

Abb. 2.9 zeigt die verschiedenen AUs im Bereich der Augenbrauen und ihre Kombinationen. AU 1 (Inner Brow Raise) basiert auf der Kontraktion des inneren M. Frontalis und hebt die inneren Teile der Augenbrauen. AU 2(Outer Brow Raise) hebt die äußeren Teile der Augenbrauen durch Kontraktion des äußeren M. Frontalis und AU 4 (Brow Lower) zieht die Augenbrauen zusammen und nach unten durch Beanspruchung des M. Corrugator an der Stirn und des M. Depressor Supercilii nahe der Nasenwurzel (Anhang B).

2.3.2 Generische Modelle und 3D Scans

Die Entwicklung von Gesichtsmodellen beinhaltet die Bestimmung geometrischer Beschreibungen und Animationsfähigkeiten. Die Gesichtsgeometrie muss unter Beachtung der Animierbarkeit konstruiert werden [9]. Die Mechanik des Gesichts spielt dabei eine übergeordnete Rolle und definiert die Struktur des Modells. In der Gesichtsanimation werden größtenteils generische Modelle verwendet. Generische Modelle sind synthetische Kopfmodelle, die aus einem Modellierungsprozess entstehen. Die Modellierung wird unter Einbezug der oben genannten Kriterien durchgeführt.
Die Erstellung von 3D Scans basiert meist auf der Verwendung von 3D Scansystemen (z.B. Laserscansysteme von Cyberware [11]). Die Scans befinden sich in einem zylindrischen Koordinatensystem mit einer zentrierten, vertikalen Hauptachse. Der Laserscanner zeichnet 512 Schritte in einem bestimmten Winkel ϕ bis zu 360 Grad und 512 vertikale Schritte h auf. Jeder Schritt misst den Radius r zusammen mit den roten, grünen und blauen Komponenten der entsprechenden Textur. Das Ergebnis sind triangulierte Meshes sowie Texturen des Modells.
Weitere Scansysteme fundieren auf bildbasierten Verfahren und rekonstruieren Kopfmodelle aus Fotos mit projizierten Gitterlinien (Anhang B).
Im Gegensatz zu generischen Modellen sind 3D Scans sehr hochaufgelöst und ent-

halten wichtige Details der spezifischen Person. Die erzeugten Meshes bestehen aus dichten Polygonnetzen, was sich nachteilig auf die Animierbarkeit auswirkt.

Abbildung 2.10: Generisches Modell und Scanmodell

2.3.3 Animationsmodelle

Die wichtigsten Animationsmethoden können in drei Kategorien eingeteilt werden:

1. Morphing bzw. Keyframing,
2. Parametrisierte Modelle,
3. Physikalische Modelle.

Morphing Modelle verlangen die Spezifikation des Gesichtsmodells in allen Extremkonfigurationen, zwischen denen während der Animation linear interpoliert wird (Abschnitt 2.3.3.1). Parametrische Modelle kontrollieren die Gesichtsform durch direkte Manipulation von Vertexgruppen der Geometrie. Dieses Verfahren wird in Abschnitt 2.3.3.2 und 2.3.3.3 verdeutlicht. Physikalische Modelle verwenden Feder-Masse Systeme oder Finite Element Netzwerke [10], um die elastischen Eigenschaften der Haut und der darunter liegenden Muskel- und Schädelstruktur (Anhang B) zu simulieren. Dieser Vorgang wird in Abschnitt 2.3.3.4 erklärt.

2.3.3.1 Morphing

In [11, 12, 13, 14] werden verschiedene Erstellungs- und Einsatzmöglichkeiten für Morphing Modelle beschrieben. Sie finden Verwendung in der Gesichtserkennung als statistische Morphing Modelle zur Abschätzung von 3D-Form und Textur aus Bildern einzelner Aufnahmen des Gesichts [11]. In [13] werden Techniken

beschrieben, um fotorealistische Animationen aus Bild- oder Videomaterial zu erstellen. Diese Techniken basieren auf einer gebräuchlichen Repräsentierung von Gesichtern und Gesichtsausdrücken als Vektorraum von 3D-Formen und Texturen. Dieser Vektorraum wird aus 3D-Laserscans von neutralen Gesichtern und Gesichtsausdrücken berechnet. Die Geometrie des *Morphing Modells* kann durch Shape- und Texturvektoren repräsentiert werden, der die x,y,z Koordinaten (bzw. RGB Werte) seiner n Vertizes enthält:

$$S = (X_1, Y_1, Z_1, \ldots, X_n, Y_n, Z_n)^T.$$

Ähnlich werden Texturvektoren erstellt, die die Farbinformation jedes Vertizes beinhalten:

$$T = (R_1, G_1, B_1, \ldots, R_n, G_n, B_n)^T.$$

Ein Morphing Modell wird konstruiert, indem eine Datenmenge von m weiteren Gesichtsgeometrien integriert wird, jede definiert über einen Shape- und einen Texturvektor S_i bzw. T_i. Alle Geometrien müssen sich in voller Korrespondenz zueinander befinden. Dadurch können neue Shape- und Texturvektoren durch Linearkombinationen ausgedrückt werden:

$$S_{mod} = \sum_{i=1}^{m} a_i S_i, \quad \sum_{i=1}^{m} a_i = 1,$$

$$T_{mod} = \sum_{i=1}^{m} b_i T_i, \quad \sum_{i=1}^{m} b_i = 1.$$

Das Morphing Modell kann durch eine Menge von Gesichtern beschrieben werden $(S_{mod}(\vec{a}))$ und kann durch den Koeffizienten $a_i = (a_1, a_2, \ldots, a_m)^T$ parametrisiert werden. Neue Gesichtsausdrücke können durch Variation des Parameters a erzeugt werden.
Dieses vorgestellte Prinzip der Shapevektoren wird in der Literatur als das Prinzip der *Morphtargets* oder auch *Blendshapes* bezeichnet.

2.3.3.2 Parametrische Modelle

Parametrisierungstechniken in der Gesichtsanimation können jedes mögliche Gesicht und jeden Gesichtsausdruck durch Kombination verschiedener Parameterwerte spezifizieren [15]. Parametrisierbarkeit erlaubt eine explizite Kontrolle der Konfigurationen des Gesichts. Durch die Kombination der Parameter wird eine große Auswahl an Gesichtsausdrücken mit relativ niedrigen Berechnungskosten möglich. Parametrische Modelle sind z.B. Pseudo-Muskelmodelle wie MPEG-4 [16] oder Muskelvektoren [17].
Zu den ersten parametrischen Modellen gehört eine frühe Entwicklung von Parke [18], um die nachteiligen Eigenschaften des Keyframings zu verbessern. Zwei

grundlegende Ideen sind im Konstruktionsprozess eines parametrischen, grafischen Modells involviert: Das erste Konzept bezieht sich auf die Parametrisierung und die Entwicklung eines geeigneten Parameterraumes, das zweite beschäftigt sich mit der Synthese zwischen den definierten Parametern und der grafischen Darstellung.
Parametrisierbarkeit bezeichnet die Identifizierung eines individuellen Members einer Klasse von Objekten durch Kriterien. Jedes individuelle Objekt kann durch spezifizierte Parameter beschrieben werden. Eine *vollständige* Parametermenge erlaubt die Spezifikation jedes Members durch Selektion geeigneter Parameterwerte. Mögliche Parameterwerte werden aus finiten, diskreten Mengen oder aus Intervallen ausgewählt und ergeben ein n-Tupel für jedes Member. Ein gegebenes Parametrisierungsschema wird in der Synthese auf das Gesichtsmodell angewendet. Aus den definierten Parameterwerten werden vor dem Renderprozess grafische Primitive berechnet.
Die Entwicklung eines Parametersets beruht auf Beobachtungen der Oberflächeneigenschaften des menschlichen Gesichts, dadurch können allerdings keine vollständigen Parametermengen extrahiert werden. Ein zweiter Versuch liegt in der Analyse der Anatomie des Gesichts. Platt, Badler [19] und Waters [17] verwenden subkutane Strukturen in einem parametrischen Muskelmodell, das Gesichtsausdrücke generieren kann. Der dritte Ansatz ist eine Kombination der ersten beiden Konzepte. Zwei Parameterkategorien beschreiben das Gesicht nach Parke:

1. *Anpassungsparameter* bestimmen das Aussehen des individuellen Gesichts,

2. *Ausdrucksparameter* kontrollieren seinen emotionalen Inhalt.

Um ein vollständiges parametrisches Ausdrucksmodell zu erstellen, basieren die Parametersätze auf dem Facial Action Coding System von Ekman (vgl. 2.3.1). Die meisten Ausdrucksparameter beziehen sich auf Regionen der Augen und des Mundes. Parke verwendet Parameter für Kieferrotation, Ausdrücke des Mundes, Position und Form der Augenbrauen, Öffnung des Augenlids etc. Veränderungen der Anpassungsparameter wirken sich global auf das Kopfmodell aus. Hautfarbe, Höhe und Breite des Modells etc. können über diese Parameter kontrolliert werden.
Parke nennt fünf Operationen, die neue Positionen der Vertizes aus den Parameterwerten bestimmen:

1. *Prozedurale Konstruktion* modelliert die Augen. Die Prozedur akzeptiert Parameter für den Augapfel, die Farbe der Iris, die Position der Augen und die Orientierung. Daraus werden Polygone für die Augengeometrie generiert.

2. *Interpolation* wird auf die Regionen des Gesichts angewendet, die ihre Form verändern. Diese werden unabhängig voneinander zwischen zwei extremen Positionen interpoliert und mit einem Parameterwert assoziiert.

3. *Rotation* wird für die Kieferrotation verwendet und kontrolliert die Öffnung des Mundes über eine Kieferachse.

4. *Scaling* kontrolliert die relative Größe von Features des Gesichts und wird nur auf Vertizes in den spezifizierten Regionen des Gesichts angewendet.

5. *Position offsets* bewegen bestimmte Gruppen von Vertizes zur Kontrolle der Nasenlänge, der Mundwinkel etc.

2.3.3.3 Parametrische Muskelmodelle

Eine frühe Arbeit von Waters [17] war die Entwicklung eines parametrischen Muskelmodells, das durch eine geringe Anzahl von Parametern animiert werden konnte. Dieser Ansatz resultierte wie alle parametrischen Ansätze aus den Nachteilen des Keyframings. Seine Arbeit basiert auf der Entwicklung eines Parametersets für das Gesicht. Die Isolierung solcher Parameter ist schwierig, aber fundamental. Gesichtsmuskeln entspringen aus dem Schädelknochen und fügen sich am anderen Ende in die Hautstruktur ein. Muskeln können durch die Orientierung der Fasciculi, den individuellen Muskelfasern, relativ zur Zugrichtung definiert werden. Waters unterteilt die Gesichtsmuskeln in obere und untere Gesichtsmuskeln. In der unteren Gesichtshälfte gibt es fünf Hauptgruppen von Muskeln: Uppers und Downers bewegen die Haut in Richtung Augenbrauen und umgekehrt in Richtung Kinn. Horizontale Muskeln kontrahieren in Richtung der Ohren bzw. Richtung der vertikalen Hauptachse des Gesichts. Schräge Muskeln kontrahieren von den Lippen aufwärts in Richtung der Wangenknochen. Kreisförmige Muskeln befinden sich um die Augen und um den Mund. Die oberen Gesichtsmuskeln sind verantwortlich für die Bewegungen der Augenbrauen, Stirn und der oberen und unteren Augenlider. Die Muskeln des Mundes zeichnen sich durch eine sehr komplexe Muskelinteraktion aus. Der Hauptmuskel Orbicularis Oris ist ein Sphinkter ohne Knochenverbindung. Zusätzlich kreuzen die Muskelfasern des M. Buccinator, M. Depressor, M. Anguli Oris, M. Depressor Labii Inferioris und M. Mentalis den Orbicularis Oris (Anhang B).
Waters Modell unterstützt lineare Muskeln und Sphinkter. Jedem Muskel wird eine Einflussregion zugeordnet, die eine konvexe Zone auf dem Mesh bildet. Die Einflussregion in der einfachsten Form ist ein Kreissektor für lineare Muskeln und eine ellipsoide Region für Sphinkter. Lineare Muskeln werden durch Muskelvektoren mit einer Länge und einer Richtung definiert. Die Richtung verläuft vom Insertionspunkt zum statischen Verbindungspunkt im Schädelknochen. Die Länge ist am Knochenpunkt Null und nimmt in Richtung Insertionspunkt zu.
Durch die Kontraktion eines Muskelvektors wird benachbartes Hautgewebe in der Einflussregion bewegt. Abb. 2.11 verdeutlicht den Effekt einer Kontraktion auf den Punkt p.

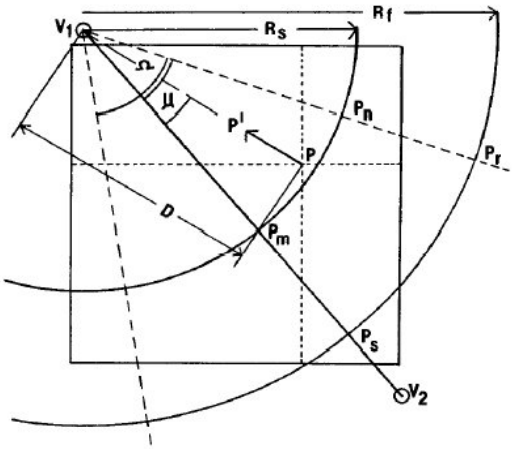

Abbildung 2.11: Muskelvektor und Einflussregion

Am Insertionspunkt wird eine maximale Verschiebung der Punkte berechnet, die in Richtung Ursprungspunkt abnimmt. Die Verschiebung wird in benachbartem Gewebe innerhalb des Sektors P_m, P_n und V_1, P_s verringert und wird anhand des Radius bestimmt.
Die Kontraktion von zirkularen Muskeln kann durch einen Kontraktionspunkt beschrieben werden, in dessen Richtung sich das benachbarte Gewebe uniform verschiebt. Abb. 2.12 zeigt den Effekt von Kontraktionen auf eine Ebene.

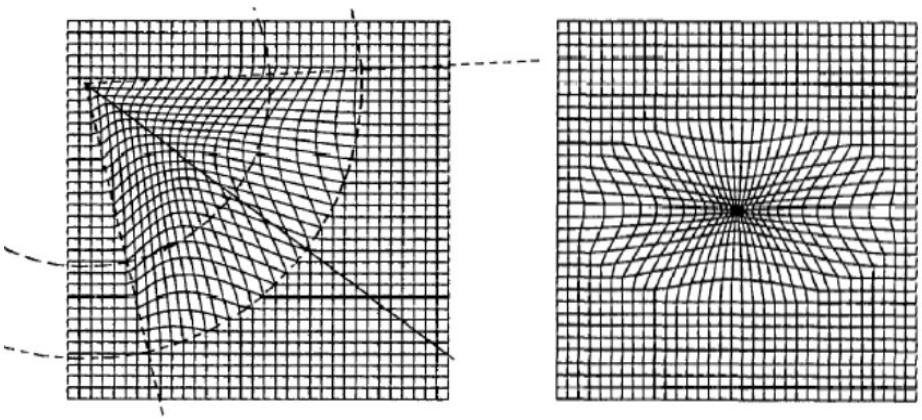

Abbildung 2.12: Kontraktion eines linearen und eines zirkularen Muskels

Das Modell wird von Parametern gesteuert, die aus Daten eingelesen werden, die Muskeln, Kieferrotationen und Fokussierung der Augen initialisieren. Integriert wurden zehn Muskeln, die wichtige AUs des FACS (vgl. 2.3.1) repräsentieren.

2.3.3.4 Physikalische Simulation

Physikalische Modelle simulieren die elastischen Hauteigenschaften sowie die darunterliegende Fett- und Muskelstruktur durch Feder-Masse Systeme. Dadurch wird versucht, die Anatomie des Gesichts so realistisch wie möglich durch physikalische Muskelkontraktionen zu simulieren. Terzopoulos und Waters [20] entwickelten ein dreidimensionales Gesichtsmodell, das aus drei Feder-Masse-Schichten besteht: Hautgewebe, subkutanes Fettgewebe und Muskeln. Eine Vereinfachung dieses Systems entwickelten Lee et al. [21] durch die Verwendung zweier Schichten und einer Verbindung dieser zu einem darunterliegenden Schädelmodell. Ähnlich zu [21] ist das Muskelmodell in [22], das für die Animation isotonische Kontraktionen durch Verkürzung linearer Segmente simuliert.
In [22] werden drei Schichten für die Animation des Gesichts verwendet. Eine Hautgewebe-Schicht als Epidermis zusammen mit einer subkutanen Fettschicht, eine Muskelschicht als Verbindung zwischen Haut- und Schädelstrukturen sowie eine Knochenschicht bestehend aus Schädel und Kiefer. In ihrem System wurden zwei Arten von Muskeln umgesetzt: lineare und zirkulare Muskeln (Anhang B). Die linearen Muskelfasern können zu Blattmuskeln, die sich z.B. an der Stirn befinden, kombiniert werden. Eine Eigenschaft der Muskeln ist die in Schichten angeordnete Struktur, sodass einzelne Muskelstränge frei übereinander gleiten können. Außerdem können sich Muskeln gegenseitig beeinflussen, diese Eigenschaft ist besonders in der Mundregion zu sehen.

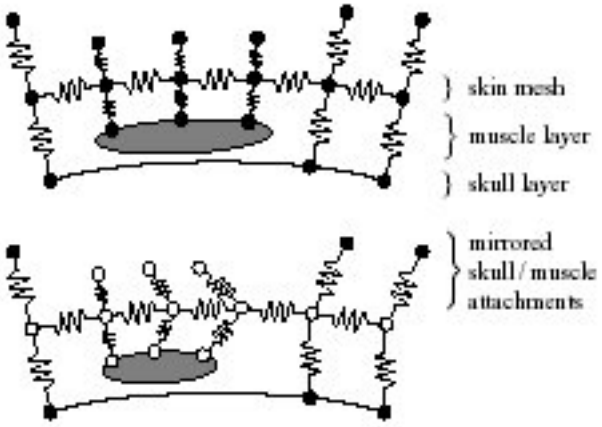

Abbildung 2.13: Feder-Masse Modell [22]

Abb. 2.13 zeigt die Verbindung zwischen Haut-, Muskel- und Knochenschicht als Feder-Masse Modell. Als Eingabedaten für die Hautsimulation wird ein 3D Scan verwendet, dessen Punkte und Kanten als initiales Federmodell dienen. Diese Federn sind biphasisch [23]: Bei hoher Spannung wird die Steifigkeit der Feder erhöht. Dadurch werden die nichtlinearen elastischen Eigenschaften der Haut simuliert.

Jeder Knotenpunkt der Oberfäche wird entweder mit der Knochenschicht oder mit der Muskelschicht durch eine Feder mit niedriger Steifigkeit verbunden. Dieses Prinzip simuliert die Eigenschaften der subkutanen Fettschicht, über die die Haut gleitet. Gespiegelte Federn werden jedem Knotenpunkt hinzugefügt, um die Vertizes nach außen zu ziehen. Dies hat den Vorteil, dass die Hautschicht während der Simulation nicht die darunter liegenden Schichten durchdringen kann. Daher sind normalerweise Methoden wie Volume Preservation oder Skull Penetration Constraints [22] nötig. Die Bewegungsgleichungen des Feder-Masse Systems werden über den Verlet Integrator numerisch integriert.
Die einzelnen Fasern der Muskeln bestehen aus linearen Segmenten, die mit Ellipsoiden überlagert werden. Eine Muskelfaser besteht aus n Kontrollpunkten $p_i \in \mathbf{R}^3$, die ein Kontrollpolygon P bilden. Eine Muskelkontraktion wird über einen Parameter $c \in [0, 1]$ berechnet. Ist der Parameter $c = 0$, so wird das Muskelpolygon nicht kontrahiert. Bei $c = 1$ entsteht volle Kontraktion und ein neues Kontrollpolygon $Q = \{q_i\}_{i=0}^{n-1}$ wird gebildet (Abb. 2.14).

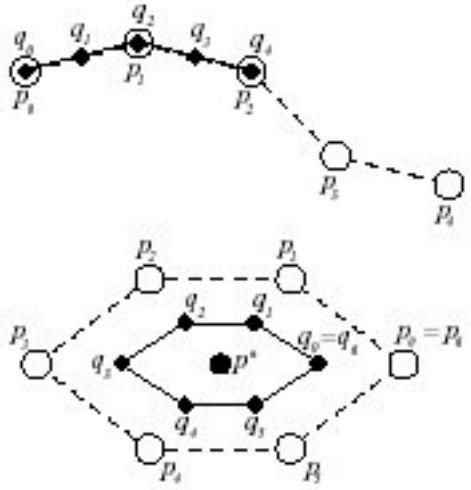

Abbildung 2.14: Kontraktionsmodell [22]

Durch lineare Interpolation werden die neuen Kontrollpunkte q_i berechnet. Bei Sphinktern werden die Segmente in Richtung des Zentrums $p^* \in \mathbf{R}^3$ kontrahiert. Bei einer gegebenen Kontraktion schwillt das Zentrum des linearen sowie die Zentren der Segmente des zirkularen Muskels durch eine Skalierung der Höhe jedes Muskelsegmentes an. Das simuliert sie Verdickung des Muskels bei einer Anspannung und die Streckung bei einer Relaxation.

2.4 Constraints

Im Folgenden werden verschiedene Techniken zur Realisierung von Constraints vermittelt. Abschnitt 2.4.1 beschreibt Constraints, die in physikalischen Simulationen Verwendung finden. Eine ähnliche Technik wird in Abschnitt 2.4.2 vermittelt, die speziell bei parametrisierten Objekten angewendet wird. Der letzte Abschnitt beschreibt Constraints in interaktiven Motion Editing Prozessen.

2.4.1 Physikalische Constraints

Barzel und Barr beschreiben in [24] physikalisch basierte Verfahren zur Kontrolle von computergrafischen Modellen. Die individuellen Elemente sind Rigid Bodies, auf die physikalische Kräfte der Newtonschen Mechanik wirken. Rigid Bodies sind primitive geometrische Körper wie Kugeln, Quader oder Zylinder. Jedes Objekt definiert spezielle Quantitäten, die für eine physikalische Simulation wichtig sind. Durch die Generalisierung auf primitive Objekte können geometrische Constraints spezifiziert werden, die durch Constraint-Kräfte auf die Objekte wirken und entsprechende Reaktionen hervorrufen. Während der Animation des Modells werden die Constraint-Kräfte kontinuierlich berechnet, sodass der Constraint kontinuierlich erfüllt ist. Das physikalische Modell besteht aus der Verbindung und Kontrolle der primitiven Körper durch Constraints sowie der Beeinflussung des Verhaltens durch explizite Anwendung externer Kräfte wie Gravitation, Federn oder Dämpfungskräfte.
Beispiele für Constraints sind u.a. fixieren eines Punktes des Körpers an einen vom Benutzer definierten Punkt (Point-to-Nail constraint) oder die Verbindung eines Punktes des Körpers mit einem Pfad, den das Objekt verfolgen soll (Point-to-Path constraint).

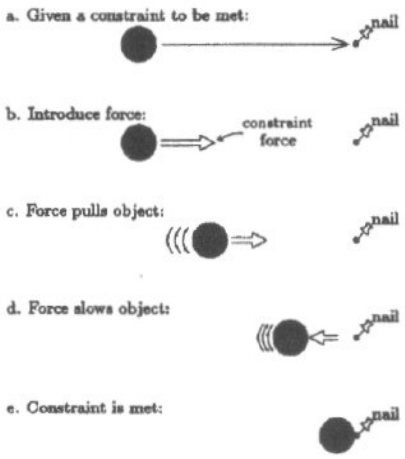

Abbildung 2.15: Die Constraint-Kraft zieht am Ball in Richtung Nagel und hält den Ball an dieser Stelle fest

Die Kräfte, die das Verhalten der Objekte beeinflussen, werden durch ein *forward dynamics* Problem gelöst. Um jedoch einen Constraint zu erfüllen muss das *inverse dynamics* Problem gelöst werden: Durch eine Beschreibung des erwünschten Verhaltens müssen Kräfte bestimmt werden, die ein angemessenes Verhalten erzeugen. Die Aufstellung und Lösung einer Constraint Force Equation berechnet die verschiedenen Constraint-Kräfte durch $MF_c + B = 0$, wobei F_c eine Menge unbekannter Constraint-Kräfte darstellt. Jeder Constraint wird durch ein Abweichungsmaß $\vec{D}$ definiert, sodass $\vec{D} = 0$ gilt, wenn ein Constraint getroffen wurde. $\vec{D}$ ist eine differenzierbare Funktion der Orientierungen und Positionen der eingeschränkten Rigid Bodies. Die Constraint Force Equation ist ein mehrdimensionales Gleichungssystem, das in [24] mittels Singlulärwertzerlegung gelöst wurde.

2.4.2 Energy Constraints

Ähnlich dem im vorherigen Abschnitt beschriebenen Verfahren ist das der Energy Constraints. In [25] wird ein einfaches und allgemeines Verfahren zur Aufstellung und Lösung von Constraints auf parametrisierte Objekte beschrieben. Constraints werden durch Energiefunktionen und Energiegradienten definiert. Diese Constraints verhalten sich wie Kräfte, die das Modell ziehen und parametrisch deformieren, um es in die gewünschte Konfiguration zu bringen. Die Energiefunktionen sind Funktionen über dem Parameterraum des verwendeten Modells. Diese nicht negativen Funktionen sind an bestimmten Punkten gleich Null, wenn ein Constraint erfüllt ist. Die Energiefunktionen werden summiert um eine skalare Funktion zu gewinnen, die anschließend im Parameterraum minimiert wird.
Eine Modellhierarchie definiert die Geometrie des Modells durch eine Sammlung von Funktionen. Eine Funktion $P(u, v)$ (parametric position function) liefert einen 3D Punkt für jedes (u, v), $N(u, v)$ (surface normal function) gibt eine Oberflächennormale zurück und $I(X)$ (inside-outside function) liefert einen skalaren Wert durch einen inside-outside Test des Testpunktes X. Für jedes Blatt der Baumstruktur sind diese Funktionen definiert. Jedes Primitiv beinhaltet reale Werte zur Beschreibung von Quantitäten wie Radius, Translationsvektor, etc. Die oben genannten Funktionen sind abhängig von diesen Werten: Wenn diese sich verändern, bewegt sich die Objektoberfläche. Das Objekt wird komplett von der Parameterbelegung Ψ bestimmt. Die Constraint-Gleichungen werden nicht algebraisch gelöst, sondern als Energiefunktionen formuliert und durch eine gradientenbasierte Methode gelöst. Die Lösung einer Menge von n Constraints sind Werte aus Ψ :

$$E(\Psi) = \sum_{i}^{n} E_i(\Psi) = 0.$$

Geometrische Constraints können durch Energiefunktionen modelliert werden. Eine Befestigung eines Punktes im Raum würde über den Energieterm $E =$

$|P^a(u_a, v_a) - Q|^2$ definiert werden, wobei a das Objekt und Q den spezifischen Punkt definiert.

2.4.3 Spacetime Constraints

Die Idee von Spacetime Constraints wird in [26] beschrieben. Physikalische Simulationsmethoden sind durch die initialen Positionen und Beschleunigungen der Objekte bestimmt sowie durch die Kräfte, die während der Simulation auf die Objekte wirken (initial value problem). Der animierende Benutzer setzt sich direkt mit der Bewegungskurve der Objekte auseinander. Probleme dieser Form, in der Anfangs- und Endbedingungen teilweise oder komplett eingeschränkt werden, sind two-point boundary Probleme. Die zentrale Charakteristik von Spacetime Constraints ist die Lösung der Objektbewegung über das gesamte Zeitintervall, anstatt sequentiell über die Zeit zu integrieren. Dadurch besteht die Möglichkeit, Bewegungen hinsichtlich ihrer Ziele direkt zu kontrollieren.
Spacetime Constraints werden in Motion Editing Umgebungen verwendet. In [27] werden bereits existierende Animationen, die neue Bedürfnisse erfüllen sollen, in einem Editing-Prozess durch direkte Manipulation verändert. Ein Spacetime Constraints Solver findet manipulierte Bewegungen, bezieht aber die gesamte Bewegung in die Berechnungen ein. Der Benutzer definiert Constraints in der Animation und verwendet einen Solver, der die 'beste' Bewegung aus den neuen Anforderungen und der existierenden Bewegung findet.

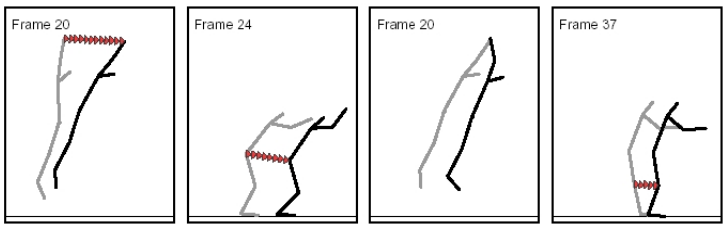

Abbildung 2.16: Editierung einer bestehenden Animation. Die Graue Figur zeigt die Position des Charakters vor der Dragging Operation [27]

Eine bestehende Sprungbewegung eines Characters soll beispielsweise um einen höheren Sprung erweitert werden (Abb. 2.16). Die traditionelle Methode für diese Aufgabe ist die Verwendung von Inverser Kinematik, bei der der Benutzer die Bewegungen durch Endeffektoren editieren kann. Der IK-Solver berechnet die beste Pose des Charakters, die diese neue Positionierung ausführt nur für den aktuellen Keyframe.
Der Ansatz verbindet die Verwendung von Spacetime Constraints mit sogenannten Motion Displacement Maps (Motion Warps). Dadurch können Bewegungen durch Addition einer neuen Bewegungskurve unabhängig voneinander editiert werden. Alle Spacetime Verfahren werden durch eine numerische, eingeschränkte

Optimierungsmethode definiert. Für eine Menge von Constraints, bestehend aus Constraints, die die existierende Animation beibehalten und Constraints, die die erwünschten Veränderungen spezifizieren, findet eine Minimierung einer Funktion statt. Das Problem ist definiert als

$$minimize \quad g(x) \quad subject \quad to \quad f(x) = c,$$

wobei x ein Vektor ist, der die Parameter der Bewegung repräsentiert, g ist eine skalare Funktion von x und f ist eine Vektorfunktion der Constraints. Durch eine interaktive Dragging-Operation findet ein Update einiger Elemente des Vektors c statt und die Gleichungen werden neu gelöst.
Im Motion Editing kann eine existierende Bewegung durch die Zeit t und einen Vekor x_0, der die Parameterbelegung der Bewegung darstellt, durch die Funktion $m_0(t, x_0)$ definiert werden. In Motion Displacement Techniken wird die existierende (initiale) Bewegung als Konstante angenommen und eine neue Bewegungskurve d addiert:

$$m(t, x) = m_0(t) + d(t, x).$$

Die in diesem Ansatz verwendeten Constraints sind größtenteils kinematisch und schränken die Konfiguration eines Charakters zu einem bestimmten Zeitpunkt ein. Diese Costraints sind definiert als

$$f(m(t_c, x))\Delta c,$$

wobei Δ eine Relation definiert, t_c ist die Zeit, in der der Constraint erfüllt ist, c ist ein Skalar und f ist die Constraint Funktion. Alle Constraint Funktionen werden in einer einzelnen Vektorfunktion gruppiert. Die Minimierung der Funktion ist eine Minimierung der Differenz zwischen der originalen und der editierten Bewegung, wodurch z.B. Endeffektoren, deren Velocities oder deren Beschleunigungen gematcht werden können.
Die möglichen Constraints dieses Systems lassen sich in drei Kategorien einteilen:

1. Constraints auf den Charakter (z.B. Winkel der Gelenke),
2. Constraints, die Informationen der initialen Bewegung beibehalten,
3. Constraints, die die initiale Bewegung regulieren.

Kapitel 3

Konzeption

Dieses Kapitel gliedert sich in zwei Hauptabschnitte. Zunächst wird ein Überblick über das allgemeine Gesamtsystem verschafft, das aus der Problemstellung in der Motivation und den Zielen dieser Arbeit abgeleitet wurde. Daraus resultieren zwei Hauptkonzepte, die tiefergehend in darauf folgenden Abschnitten entwickelt werden. Die Problemstellung der Personalisierbarkeit wird zunächst behandelt und vermittelt verschiedene Lösungen, aus denen das erste Konzept erarbeitet wird. Im zweiten Abschnitt wird, auf dieser Vorgehensweise aufbauend, das Konzept eines Constraintsystems entwickelt.

3.1 Gesamtüberblick

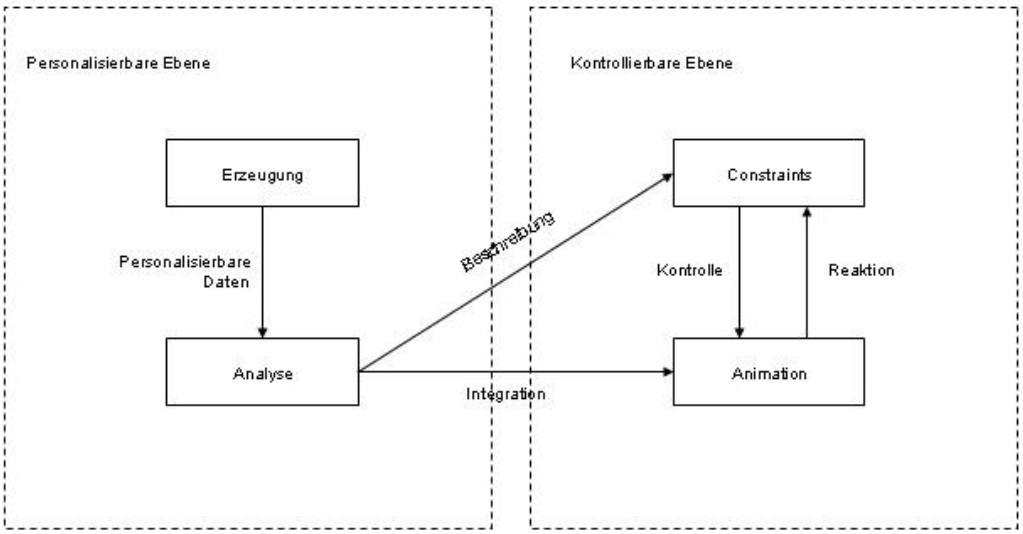

Abbildung 3.1: Abstrahiertes Gesamtsystem

Ein abstrahiertes Gesamtsystem wird in Abb. 3.1 gezeigt. Die Architektur besteht aus zwei Hauptkomponenten:

Eine Personalisierbare Ebene erzeugt, analysiert, und integriert personalisierbare Daten.

Eine Kontrollierbare Ebene verwendet, kontrolliert und reagiert auf personalisierbare Daten.

3.2 Personalisierbarkeit

Personalisierbarkeit ist entscheidend, um realistische Gesichtsanimationen zu erzeugen. Personalisierbare Informationen sind individuelle Gesichtszüge und Ausdrücke, die über die Verwendung möglichst realer Daten in den Animationsprozess integriert werden. Reale Daten sind Bild-, Video- oder Audiomaterial, die in einem analytischen Prozess ausgewertet werden. Personalisierbare Information wird unter anderem durch geometrische Daten realer Personen aquiriert. Es werden verschiedene Techniken zur Erzeugung personalisierbarer Morphtargets für ein generisches Gesichtsmodell vermittelt. Zunächst leitet ein genereller Überblick in die Thematik und erläutert die Vorgehensweise anhand einer allgemeinen Pipeline.

3.2.1 Überblick

Die Verfahren bestehen aus mehreren Teilschritten, um ein realistisches Morphing zwischen zwei Geometrien zu ermöglichen. Die im Folgenden dargestellten Techniken bilden eine Pipeline ähnlich [11](Abb. 3.2):

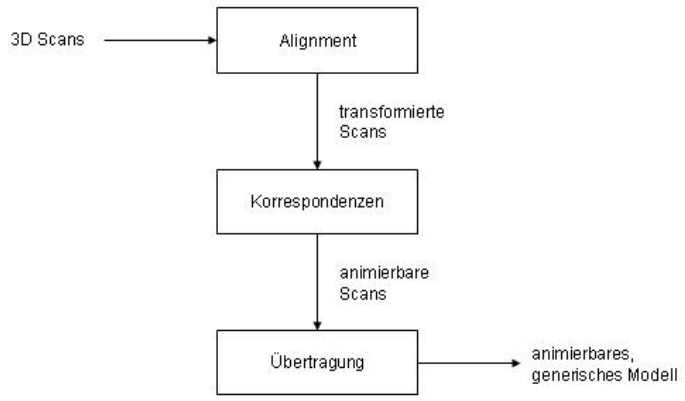

Abbildung 3.2: Pipeline zur Erstellung von Morphtargets

1. Erzeugung der Datenbasis,

2. Globales Alignment,
3. Berechnung von Punkt-zu-Punkt Korrespondenzen,
4. Übertragung auf einen generisches Modell.

Die Basis für ein solches System ist die Generierung realistisch aussehender und hochaufgelöster Meshes sowie Texturen. Dazu ermöglichen 3D-Scanverfahren die Rekonstruktion personalisierbarer Gesichtsausdrücke (Anhang A). Damit alle Modelle im Koordinatensystem in der gleichen Ausrichtung zueinander liegen, werden Alignment-Verfahren auf die Objekte angewendet. Der nächste Schritt ist die automatische Berechnung von Punkt-zu-Punkt Korrespondenzen zwischen dem neutralen Kopfmodell und allen Gesichtsausdrücken. Damit wird für die lineare Interpolation des Morphings ein gleichmäßiger Übergang zwischen Start- und Endpunkt korrespondierender Strukturen in den Gesichtsmodellen geschaffen [12]. Im letzten Schritt werden die erzeugten Modelle auf ein generisches Gesichtsmodell übertragen. Die Übertragung hat den Vorteil, dass die Mesh-Topolgie des generischen Modells viel strukturierter ist als die der hochaufgelösten Scanmodelle. Das Mesh des generischen Kopfes ist in Regionen, in denen starke oder unterschiedliche Bewegungen stattfinden können, feiner trianguliert als in statischen Regionen.

3.2.2 Alignment

Zur gemeinsamen Darstellung und Analyse von geometrischen (Mess-)Daten wird häufig eine Anpassung der Daten und die Darstellung in einem gemeinsamen Koordinatensystem notwendig.
Die rigide Registrierung ist ein einfaches, aber ein für viele Anwendungen ausreichendes Matchingverfahren. Außerdem ist sie die am meisten erforschte und am häufigsten verwendete Registrierungstechnik. Die mathematisch ausgereiftesten Registrierungsverfahren matchen korrespondierende Punkte zweier Volumen und werden als rigide Least Square Registrierungsverfahren bezeichnet. Wird der rigide Suchraum (Rotation/Translation) um den Skalierungsfaktor erweitert, wird die Registrierung affin und hat für viele verschiedene Anwendungen große Relevanz.
Eine weitere Möglichkeit ist die Verwendung von Methoden der analytischen Fotogrammetrie, bei denen über die Beziehung zwischen Kamera- und Objektkoordinatensystemen Informationen über die Objektorientierung berechnet werden. Im Folgenden wird eine Auswahl wichtiger Alignmenttechniken vorgestellt.

3.2.2.1 Iterative Closest Point

Besl und McKay [28] entwickelten einen einfachen und effizienten Registrierungsalgorithmus für 2D und 3D Formen.

Die Methode konvergiert immer monoton zum nächsten lokalen Minimum einer durchschnittlichen quadratischen Distanzmetrik, die zwischen nächstgelegenen Punkten zweier Geometrien berechnet wird. Falls das Minimum des durchschnittlichen quadratischen Fehlers (Mean Square Error, MSE) nur ein lokales Minimum ist, wurde eine fehlerhafte Transformation berechnet. Um dies zu verhindern sollten die Geometriedaten grob vorregistriert sowie statistische Ausreißer in der Geometrie beseitigt werden. Der Iterative Closest Point berücksichtigt sechs Freiheitsgrade, sodass die optimale Rotation und Translation mit einem iterativen Verfahren durch Minimierung von Distanzen abgeschätzt werden können. Geometrische Daten wie Punktmengen, Liniensegmente, implizite und parametrische Kurven, Dreiecksmengen sowie implizite und parametrische Flächen können für die Registrierung verwendet werden. Alle Closest Point Algorithmen sind entweder Quaternionen-basiert oder verwenden die Singulärwertzerlegung (SVD). Beide Verfahren erzeugen eine Kreuzkovarianz Matrix relativ zum Schwerpunkt zweier Punkteverteilungen, durch die die optimale Rotation und Translation berechnet werden können. Die Kreuzkovarianz Matrix ist das Analogon zur Kovarianz Matrix einer Punktemenge, aus der sich deren Hauptachsensystem berechnen lässt. Mit der SVD Methode kann der Algorithmus auf n-dimensionale Anwendungen verallgemeinert werden. Für $n \leq 3$ ist allerdings die Quaternionen-basierte Methode ausreichend.
Dem Algorithmus werden zwei abstrakte geometrische Formen zur Verfügung gestellt: Eine Datenform P, die so positioniert wird, dass sie sich in der besten Ausrichtung mit einer Modellform X befindet. Da die Daten in unterschiedlichen geometrischen Repräsentierungen vorliegen können, müssen diese zunächst in Punktmengen zerlegt werden. Es sind $P = \{p_i\}$ die Punkte der Datenform und $X = \{x_i\}$ die Punkte der Modellform. Die Anzahl der Punkte der Datenform bzw. der Modellform ist N_p bzw. N_x. Der ICP Algorithmus berechnet in jeder Iteration einen Vektor $\vec{q} = [\vec{q}_R|\vec{q}_T]$, der den entsprechenden Registrierungszustand der Rotationsquaternions (vgl. 2.2.2) $\vec{q}_R = [q_0 q_1 q_2 q_3]^T$ und der Translation $\vec{q}_T = [q_4 q_5 q_6]^T$ beinhaltet. Die zu minimierende mittlere quadratische Funktion ist:

$$f(\vec{q}) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|\vec{x}_i - R(\vec{q}_R)\vec{p}_i - \vec{q}_T\|^2.$$

Zunächst wird die Kreuz Kovarianzmatrix Σ_{px}

$$\Sigma_{px} = \frac{1}{N_p} \sum_{i=1}^{N_p} [(\vec{p}_i - \vec{\mu}_p)(\vec{x}_i - \vec{\mu}_x)^t] = \frac{1}{N_p} \sum_{i=1}^{N_p} [\vec{p}_i \vec{x}_i^t] - \vec{\mu}_p \vec{\mu}_x^t$$

mit den Schwerpunkten

$$\vec{\mu}_p = \frac{1}{N_p} \sum_{i=1}^{N_p} \vec{p}_i, \quad \vec{\mu}_x = \frac{1}{N_x} \sum_{i=1}^{N_x} \vec{x}_i$$

der beiden Punktmengen berechnet. Die zyklischen Komponenten einer neuen antisymmetrischen Matrix $A_{ij} = (\sum_{px} - \sum_{px}^T)_{ij}$ bilden eine Vektor $\Delta = [A_{23} \quad A_{31} \quad A_{12}]^T$, mit dem folgende symmetrische Matrix $Q(\Sigma_{px})$ gebildet wird:

$$Q(\Sigma_{px}) = \begin{bmatrix} tr(\sum_{px}) & \Delta^T \\ \Delta & \Sigma_{px} + \Sigma_{px}^T - tr(\Sigma_{px}) I_3 \end{bmatrix}.$$

Die optimale Rotation ist der Eigenvektor $\vec{q}_R = [q_0 q_1 q_2 q_3]^T$ mit dem maximalen Eigenwert der Matrix $Q(\sum_{px})$. Die optimale Translation $\vec{q}_T$ wird durch

$$\vec{q}_T = \vec{\mu}_x - R(\vec{q}_R)\vec{\mu}_p$$

gefunden.
Zur Berechnung der nächstgelegenen Punkte zweier Punktmengen wird die euklidische Distanz verwendet. Zwischen zwei Punkten $\vec{r}_1 = (x_1, y_1, z_1)$ und $\vec{r}_2 = (x_2, y_2, z_2)$ gilt

$$d(\vec{r}_1, \vec{r}_2) = \|\vec{r}_1 - \vec{r}_2\| = \sqrt{(x_2 - x_1)^2 + (y_2 - y1)^2 + (z_2 - z_1)^2}.$$

Die Distanz zwischen einem Punkt $\vec{p} \in P$ und der Modellform X ist

$$d(\vec{p}, X) = min_{i \in \{1,\ldots,N_x\}} d(\vec{p}, \vec{x}_i).$$

Die resultierende Menge an nächstgelegenen Punkten Y ist also die Berechnung aller nächstgelegenen Punkte zwischen P und X .

$$Y = C(P, X).$$

Wenn die Menge Y berechnet wurde, kann die Registrierung berechnet werden mit

$$(q, d) = Q(P, Y).$$

Die Positionen der Punkte der Datenform werden aktualisiert:

$$P = q(P).$$

Eine Punktmenge P mit N_p Punkten $\vec{p}_i$ der Datenform und die Modellform X mit N_x Punkten sind gegeben. Die Iteration wird mit $P_0 = P$, $\vec{q}_0 = [1, 0, 0, 0, 0, 0, 0]^t$ und $k = 0$ initialisiert. Der Algorithmus beginnt mit der Berechnung der nächstgelegenen Punkte zwischen den Modellen. Daraus lässt sich die Registrierung $\vec{q}$ berechnen, die auf das Datenmodell angewendet wird. Zuletzt wird in jeder Iteration die Veränderung des MSE zwischen der letzten und der aktuellen Iteration berechnet. Falls dieser unter einen Schwellwert $\tau > 0$ fällt, wird die Iteration beendet.
Es existieren verschiedene Varianten des Algorithmus, die versuchen, Geschwindigkeit und Genauigkeit des ICP zu verbessern.

3.2.2.2 Trimmed Iterative Closest Point

In [29] wird eine Erweiterung des Iterative Closest Point Algorithmus präsentiert. Diese robustere Version kann auf vorregistrierte, sich teilweise überschneidende und verrauschte geometrische Daten angewendet werden und versucht, Robustheit, Schnelligkeit und Konvergenz des originalen ICP zu verbessern. Der kritische Faktor ist sicherlich die Robustheit, da der ICP ausreißerfreie Punktmengen benötigt und annimmt, dass die Datenform P eine Untermenge der Modellform M ist, sodass für alle $\vec{p}_i \in P$ eine zulässige Korrespondenz in M gefunden werden kann.
Der Grundidee des TrICP beruht auf der Verwendung von Least Trimmed Squares (LTS). Mittels LTS werden die quadratischen Fehler aufsteigend sortiert und die Summe einer bestimmten Anzahl kleinerer Werte minimiert. Wie beim ICP werden auch hier zwei 3D Punktmengen $P = \{\vec{p}_i\}_1^{N_p}$ und $M = \{\vec{m}_i\}_1^{N_m}$gematcht, indem eine euklidische Transformation berechnet wird. Angenommen wird eine unterschiedliche Anzahl von Punkten $N_p \neq N_m$ und eine bestimmte Untermenge in P besitzt keine Korrespondenzen in M. Dennoch existiert ein sicherer Anteil korrespondierender Punkte. Dieser Anteil wird als minimale Überschneidungsrate (minimal overlap rate) ξ bezeichnet. Somit ergibt sich die Anzahl der Datenpunkte mit Korrespondenz in M aus $N_{po} = \xi N_p$. Nach dieser Bedingung wird die optimale euklidische Transformation berechnet, die die Untermenge mit N_{po} Punkten von P in die beste Ausrichtung mit M bringt.
Definiert wird eine individuelle Distanz von einem Datenpunkt $p_i(R,t)$ zur Modellform M als eine Distanz zum nächstgelegenen Punkt in M:

$$m_{cl}(i,R,t) = argmin_{m \in M} \|m - p_i(R,t)\|$$

$$d_i(R,t) = \|m_{cl}(i,R,t) - p_i(R,t)\|.$$

Durch Minimierung der Summe der kleinsten N_{po} Distanzen d_i^2 wird die Transformation (R,t) berechnet.
Die Struktur des Algorithmus ist dem originalen ICP ähnlich:

1. Berechne für jeden Punkt $p \in P$ den nächstgelegenen Punkt in M und die individuelle Distanz d_i^2,
2. Sortiere d_i^2 in aufsteigender Reihenfolge, selektiere N_{po} kleinste Werte und berechne ihre Summe S_{TS},
3. Falls eine Abbruchbedingung gilt, beende Iteration; ansonsten setze $S'_{TS} - S_{TS}$,
4. Berechne für N_{po} selektierte Paare die optimale Transformation (R,t), die S_{TS} minimiert,
5. Transformiere P mit (R,t).

3.2.2.3 Absolute 3D-3D Orientation

Die analytische Fotogrammetrie beschäftigt sich mit Techniken, die von Messungen einer oder mehrerer 2D perspektivischen Projektionen eines 3D Objekts auf die 3D Position und Orientierung des beobachteten 3D Objekts schließen lassen. Diese Probleme können durch nichtlineare Optimierungsmethoden konstruiert werden. Absolute Orientation bestimmt sieben Parameter eines Stereo Modells in einem Weltkoordinatensystem: Dies sind die Skalierung, die drei Translations- und die drei Rotationsparameter. Dies wird durch die 3D Koordinaten zentraler Punkte erreicht, deren Position auf dem Stereo-Bild berechnet werden können.
Der Algorithmus Absolute 3D-3D Orientation [30] behandelt das Problem des Alignments auf Basis des Kamera- und Modell-Koordinatensystems und berechnet die Transformation aus korrespondierenden Punkten. Gegeben sind N 3D-Punkte $p_1, \ldots, p_n$ relativ zum Kamera-Koordinatensystem. Dieselben N 3D-Punkte liegen im Koordinatensystem des Modells an den Stellen $q_1, \ldots, q_n$. Um zu bestimmen, wo sich das 3D Objekt befindet, muss die Rotationsmatrix R und der Translationsvektor t im Bezug auf das Kamera-Koordinatensystem berechnet werden.

$$p_n = Rq_n + t, \quad n = 1, \ldots, N.$$

Die 3D-3D Absolute Orientation oder auch 3D-3D Posenschätzung folgert R und t aus $q_1, \ldots, q_n$ und $p_1, \ldots, p_n$. Eine Verallgemeinerung der Posenschätzung ist die Annahme einer unbekannten Skalierung.

$$p_n = sRq_n + t, \quad n = 1, \ldots, N,$$

dabei wird der Skalierungsfaktor s als 1 angenommen.
Für die Bestimmung von R und t wird eine eingeschränkte Methode der kleinsten Quadrate verwendet, die die gewichtete Summe

$$\sum_{n=1}^{N} w_n \|p_n - (Rq_n + t)\|^2$$

minimiert mit der Einschränkung, dass R eine Rotationsmatrix ist, also $R' = R^{-1}$. Eingeschränkte Minimierungs- bzw. Maximierungsmethoden werden mit Lagrang'schen Multiplikatoren gelöst. Durch eine Singulärwertzerlegung wird ein lineares Gleichungssystem nach R aufgelöst. Falls die Determinante von R -1 ist, beinhaltet R zusätzlich eine Spiegelung. Das geschieht, wenn nur wenige oder verrauschte Beobachtungspunkte verwendet wurden. Ist dies der Fall, müssen berechnungsintensivere Lösungen mittels Quaternionen oder nichtlinearen Optimierungsmethoden angewendet werden.

3.2.3 Punkt-zu-Punkt Korrespondenzen

Der zentrale Schritt in der Erstellung eines Morphing Modells ist die Definition von Punkt-zu-Punkt Korrespondenzen zwischen jedem Scan und einem Re-

ferenzscan. Bei einer kontinuierlichen Veränderung des Parameters a_i zur Gewichtung der Shape-Vektoren (vgl. 2.3.3.1) wird ein glatter Übergang generiert, sodass sich jeder Punkt des Startobjekts auf einen Punkt des Zielobjekts bewegt. Intermediäre Zustände des Morphs können Artefakte produzieren, wenn Start- und Zielpunkte keine korrespondierenden Strukturen des Gesichts (z.B. Nasenspitze) sind. Im Folgenden werden interessante Techniken vermittelt, um Punkt-zu-Punkt Korrespondenzen zwischen Gesichtsmodellen zu bestimmen. Die vorgestellten Verfahren basieren auf sehr unterschiedlichen Ideen. Dennoch können die ersten beiden Techniken als Modell-basierte Verfahren bezeichnet werden, da sie nur geometrische Informationen zur Berechnung der Korrespondenzen heranziehen. Das letzte Verfahren ist ein hybrider Ansatz, der Bild- und 3D Information in die Berechnungen integriert.

3.2.3.1 Netze radialer Basisfunktionen

Noh und Neumann verwenden in [31] einen Ansatz zur automatischen Korrespondenzsuche, basierend auf der Selektion von weniger als zehn korrespondierenden Punkten zwischen zwei Modellen durch den Benutzer. Diese anfänglichen Korrespondenzen werden durch Volume Morphing mit radialen Basisfunktionen interpoliert, wodurch bestimmte Featurepoints des ursprünglichen Modells (Augenpartie, Nase, Kinn) grob an das Zielmodell angepasst werden. Die relativ geringe Menge der selektierten Korrespondenzen produziert allerdings noch keine perfekte Übereinstimmung der beiden Meshes. Damit alle Punkte des Ursprungsmodells im Zielmodell liegen, wird zusätzlich eine zylindrische Projektion des grob gemorphten Modells auf das Zielmodell angewendet.
Radiale Basisfunktionen sind für ihre besondere Interpolationsfähigkeit bekannt. Durch die Eingabe der selektierten Punkte wird ein Netz von RBFs trainiert. RBF-Netze sind neuronale Netze [32] und bestehen aus einer Eingabeschicht, einer Ausgabeschicht und einer verdeckten Schicht dazwischen. Die grundlegende Aufgabe eines RBF-Netzes ist, aus gegebenen (mehrdimensionalen) Stützstellen und deren Ausgaben durch weitere Eingabemuster approximierte Ausgaben zu berechnen. Mathematisch ist die Berechnung des Netzes eine Abbildung $f : R^n \rightarrow R$.
Zur Berechnung der Funktion sind n Stützstellen gegeben. Jede Stützstelle ist ein n-dimensionaler Vektor $X_i = (x_{i1}, ..., x_{in})$ mit den zugeordneten Funktionswerten Y_i und einer Interpolationsbedingung: $f(X_i) = Y_i$ mit $i \in 1, ..., n$. Radialsymmetrische Basisfunktionen $h_i : R^+ \rightarrow R$, $h_i(\|X - X_i\|)$, die von den Stützstellen und vom Abstand des Eingabevektors von den Stützstellen abhängig sind, werden verwendet, um die genannte Funktion zu interpolieren. Abb. 3.3 zeigt ein Beispiel für eine interpolierte Funktion (durchgezogene Linie), die durch vier radialsymmetrische Basisfunktionen $h_1, ..., h_4$ (gestrichelte Linie) approximiert wurde. Die Funktionen sind unterschiedlich gewichtet, besitzen verschiedene Stützstellen und liegen als Gauß'sche Glockenkurven vor.

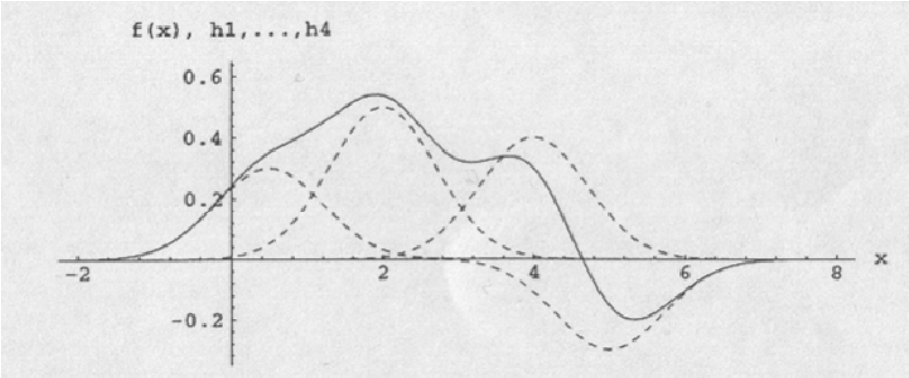

Abbildung 3.3: Radiale Basisfunktionen [32]

Die Eingabeschicht des neuronalen Netzes besteht aus n Neuronen entsprechend dem Eingaberaum. Die verdeckte Schicht (Hidden Layer) bildet n Neuronen, die der Anzahl der vorgegebenen Trainingsmuster entspricht. Die Anzahl der Neuronen in der Ausgabeschicht wird durch die Größe des Ausgaberaumes definiert. Im einfachsten Fall sind Stützstellen, die gewünschte Ausgabe und Zentrumsfunktionen gegeben. Daher müssen über ein lineares Gleichungssystem mit n Gleichungen die Gewichtungen der Ausgabeschicht berechnet werden. Die Eingabevektoren werden an die verdeckte Schicht übergeben, deren Aktivierungsfunktionen gleich den verwendeten Zentrumsfunktionen ist. In der verdeckten Schicht werden die Eingabevektoren mit der Aktivierungsfunktion ausgewertet. Die gewichteten Ausgaben dieser Neuronen werden berechnet und an die Ausgabeschicht übermittelt, die die entsprechende Summenfunktion $f(X)$ berechnet. Abb. 3.4 zeigt dieses Prinzip.

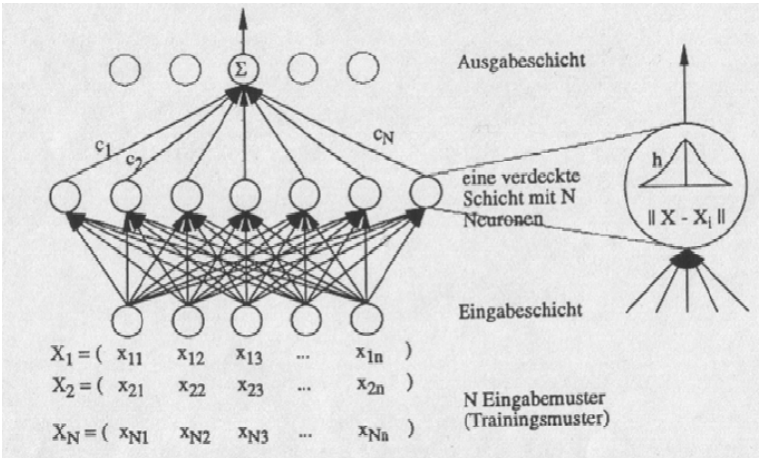

Abbildung 3.4: RBF-Netz [32]

Für das in [31] verwendete RBF-Netz gilt die Funktion:

$$f(X) = \sum_{j=1}^{N} w_j h_j(X)$$

mit den Basisfunktionen $h_j(X) = \sqrt{\|X - X_j\|^2 + s_j^2}$ und den zu bestimmenden Gewichtungen w_j. Die Distanz s_j wird zwischen $\vec{x}_j$ und dem nächstgelegenen x_j berechnet, $s_j = min_{i \neq j} \|x_i - x_j\|$. Durch die Distanzberechnung werden weit entfernte Feature Points weniger deformiert als nah beieinander liegende. Dieses Netz wird dreimal mit den 3D Koordinaten der selektierten Korrespondenzen des Startmodells ($\vec{x_i}$) und denen des Zielmodells ($\vec{y_i}$) trainiert.
Nach der Deformation durch die RBF Funktionen wird das Startmodell auf das Zielmodell projiziert. Eine vertikale Linie durch den Schwerpunkt des Modells wird als Mittelachse für die zylindrische Projektion verwendet. Zur Mittelachse senkrechte Strahlen werden durch jeden Vertex des Startmodells auf das Zielmodell verfolgt. Der erste gefundene Schnittpunkt mit dem Zielmodell wird durch baryzentrische Koordinaten berechnet. Dies wird durch Lösen eines linearen Gleichungssystems erreicht:

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_2 & z_2 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix},$$

mit $0 \leq b_1, b_2, b_3 \leq 1$. Abb. 3.5 zeigt das Ergebnis der Deformation.

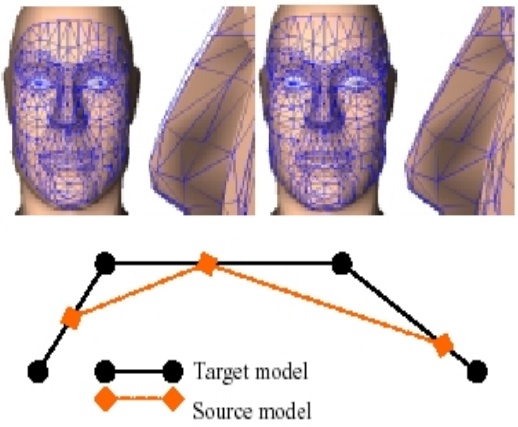

Abbildung 3.5: Ergebnis des Volume Morphings (oben) mit zylindrischer Projektion. 2D Ansicht der zwei Modelle nach der Projektion (unten) [31]

3.2.3.2 Harmonic Maps

Ein verbreitetes Verfahren zur Bestimmung von Punkt-zu-Punkt Korrespondenzen ist das sogenannte Harmonic Mapping [33]. Die grundlegende Idee ist die

Definition mehrerer Referenz-Shapes mit Vertex-zu-Vertex Korrespondenzen zwischen zwei Meshes. Eine Referenz-Shape definiert eine Partition des originalen Meshes. Die Harmonic Map bettet jede Partition in eine polygonale Region in der Ebene ein, das sogenannte Harmonic Shape Image [harmoninc maps applications]. Harmonic Mapping ist ein Partitionierungsalgorithmus, der die wichtigsten metrischen Strukturen (Winkel, Längen, etc.) in einer Partition eines Meshes erhält. Durch diese neue Mesh-Topologie können die Korrespondenzen durch Überlagerung der beiden Meshes bestimmt werden. Die Anzahl der Partitionen zwischen den beiden Meshes muss gleich sein, sodass sie paarweise auf dieselbe Ebene gemappt werden können.
Ziel des Harmonic Mappings ist die Konstruktion einer Parametrisierung einer (topologischen) Disc D, die eine Teilmenge des originalen Meshes M ist (Abb. 3.6). Die Parametrisierung erfolgt in einer konvexen polygonalen Region $P \subset R$. Die Harmonic Map $h : D \rightarrow P$ bestimmt das Mapping zwischen den verschiedenen Topologien. Das Harmonic Shape Image kodiert die 3D Information der topolgischen Disk in der 2D Domäne. Harmonic Maps sind Lösungen partieller Differenzialgleichungen [35]. Aufgrund hoher Berechnungskosten werden Harmonic Maps in zwei Schritten approximiert. Zunächst werden die Ränder eines 3D-Oberflächenpatches des originalen Meshes auf die Ränder eines gleichseitigen Dreiecks gemappt, das als 2D Zieldomäne ausgewählt wurde. Im zweiten Schritt

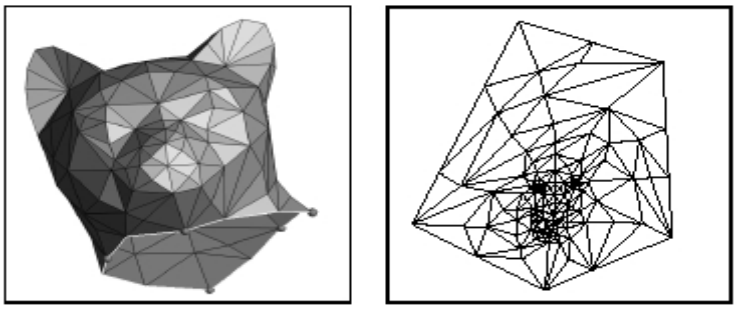

Abbildung 3.6: Originales Mesh und Harmonic Map aus [33]. Eckpunkte des Polygons sind durch kleine Bälle markiert

wird die innere Geometrie des Oberflächenpatches auf das Innere des gleichseitigen Dreiecks gemappt, mit dem Randmapping als Constraint. Das innere Mapping erfolgt durch Integration über jedes Face, wobei die Funktion E_{harm} als die Energie eines Federsystems interpretiert werden kann. Jede Feder wird mit einer Kante in D assoziiert:

$$E_{harm}(h) = \frac{1}{2} \sum_{i,j \in Edges(D)} k_{i,j} \|h(i) - h(j)\|.$$

Die Federkonstante $k_{i,j}$ wird abhängig von der Kantenlänge und des Flächeninhalts des Dreiecks in D berechnet. Das Mapping von D nach P ist eine Regulierung der Federlängen, die über die Ränder von P gespannt werden.

In [34] wird das Harmonic Mapping für die Bestimmung von Punkt-zu-Punkt Korrespondenzen verwendet. Abb. 3.7 zeigt die Vorgehensweise zur Erstellung einer Korrespondenz Map. Aus zwei hochaufgelösten Kopfmodellen werden durch einen Vereinfachungsalgorithmus (MAPS) zwei grobe Modelle berechnet. Der MAPS

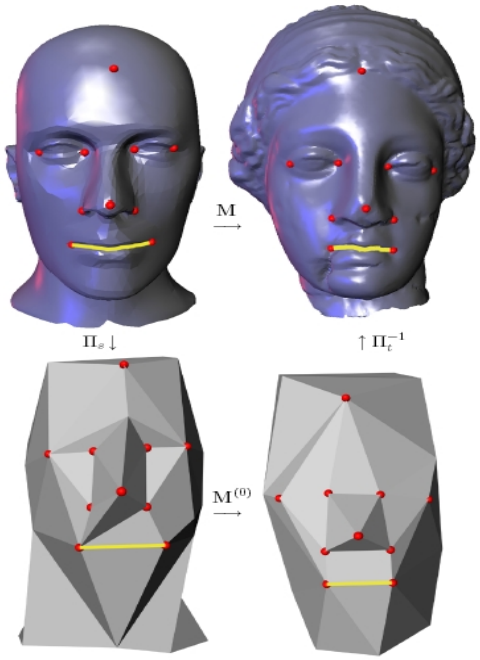

Abbildung 3.7: Erstellung einer Korrespondenz-Map [34]

Algorithmus verwendet selektierte Feature Points und Feature Lines auf den originalen Modellen. Unabhängige Vertexgruppen werden durch MAPS aus dem Mesh entfernt und entstehende Löcher neu trianguliert, wobei die selektierten Feature Points nicht gelöscht werden. Bei U selektierten Punkten in beiden Meshes mit s bzw. t Vertices gilt folgende Parametrisierung:

$$\Pi_s(s_i) = s_i, \quad \Pi_t(t_i) = t_i \quad 1 \leq i \leq U.$$

Aus dieser Mapping Funktion zwischen originalen Modellen und Basisdomäne wird eine erste Korrespondenz Map $M^0(s_i) = t_i$ berechnet. Weiterhin müssen Korrespondenzen zwischen den Vertizes der Basisdomäne berechnet werden, die keine Features sind. Ein weiteres Verfahren richtet die Basisdomänen zueinander aus und projiziert die Quell-Basisdomäne auf die Ziel-Basisdomäne durch eine iterative Relaxationsmethode auf dem Mesh, um das anfängliche Mapping zu verbessern. Durch die Relaxationsmethode werden die Vertizes der gemappten Quell-Basisdomäne auf der Ziel-Basisdomäne gleichmäßig verteilt. Daraus ergibt sich das Mapping $M^{(0)}$ für alle Punkte von $S^{(0)}$. Um alle Punkte der Basisdomäne zu integrieren, wird jedes Dreieck der Quell-Basisdomäne durch $M^{(0)}$ auf die

Ziel-Basisdomäne gemappt. Die Punkte eines Dreiecks I, J, K liegen nicht in einem einzelnen Dreieck der Ziel-Basisdomäne. Durch Shortest Path Berechnungen mit $\overline{IJ}$, $\overline{JK}$ und $\overline{KI}$ wird eine Region auf der Ziel-Basisdomäne definiert (Abb. 3.8). Eine Harmonic Map wird berechnet, die die Region I, J, K als Randpunkte

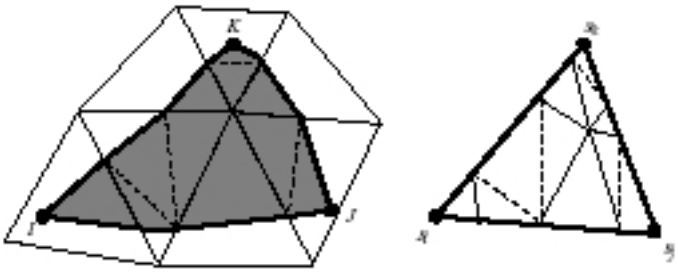

Abbildung 3.8: Harmonic Mapping auf ein Dreieck [34]

verwendet und I, J, K auf s_i, s_j und s_k des korrespondierenden Quell-Dreiecks mappt. Dieser Vorgang wird auf jedes Dreieck der Quell-Basisdomäne angewendet. Dadurch entsteht die Map $M^{(0)}$ für jeden Punkt der Quell-Basisdomäne.

3.2.3.3 Optical Flow

Zur Bestimmung von Korrespondenzen wird in [11]-[14] der optische Fluss mit einer Verallgemeinerung auf 3D Objekte verwendet. Obwohl der originale Algorithmus auf eine zeitliche Bildfolge desselben dargestellten Objektes angewendet wird, kann der optische Fluss auch bei verschiedenen Objekten gute Resultate erzielen. In [13] wurden als Datenbasis für den Algorithmus 35 statische Lasers-

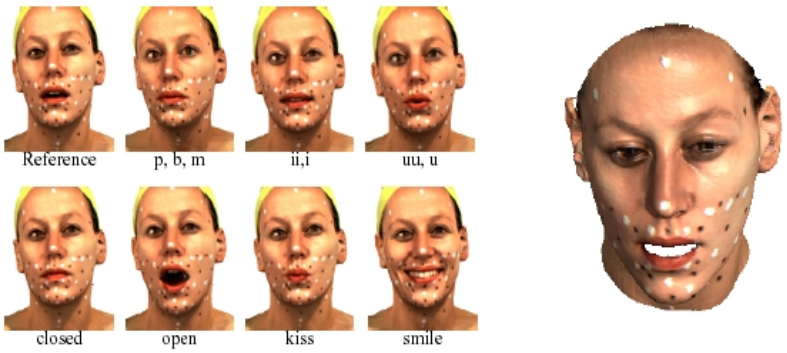

Abbildung 3.9: Datenbasis mit verschiedenen Mundkonfigurationen und das Referenzmodell [13]

cans einer Person mit wichtigen Visemen aufgenommen. Das Morphing Modell basiert auf einem Referenzscan, der zwei Bedingungen erfüllt:

1. Der Referenzscan muss so ähnlich wie möglich zu allen anderen Scans sein,
2. Nur die Oberflächenregionen, die Teil des Referenzscans sind, können in neuen Linearkombinationen repräsentiert werden.

Der Referenzscan ist ein komplettes Modell mit einer intermediären Mundkonfiguration. Zusätzlich wurden Zähne in das Referenzmodell integriert und Punkte auf die Haut der gescannten Person für ein präzises 3D Alignment gemalt.
In diesen Ansätzen werden die Messdaten des Scanners für jeden Punkt kombiniert:

$$I(h,\phi) = (r(h,\phi), R(h,\phi), G(h,\phi), B(h,phi))^T$$
$$(h,\phi) \in (0,...,511).$$

Die Repräsentierung des Kopfmodells in zylindrischen Koordinaten ermöglicht die Parametrisierung der Oberfläche des Gesichts durch die zwei Parameter h und ϕ (vgl. 2.3.2). Der optische Fluss (vgl. 2.1.3) berechnet die Korrespondenzen als Vektorfeld

$$v(h,\phi) = (\Delta h(h,\phi), \Delta\phi(h,\phi))^T.$$

Jeder Punkt des Scans $I_1(h,\phi)$ besitzt einen korrespondierenden Punkt auf dem zweiten Scan $I_2(h+\Delta h, \phi+\Delta\phi)$ (Abb. 3.10). Um den optischen Fluss auf 3D

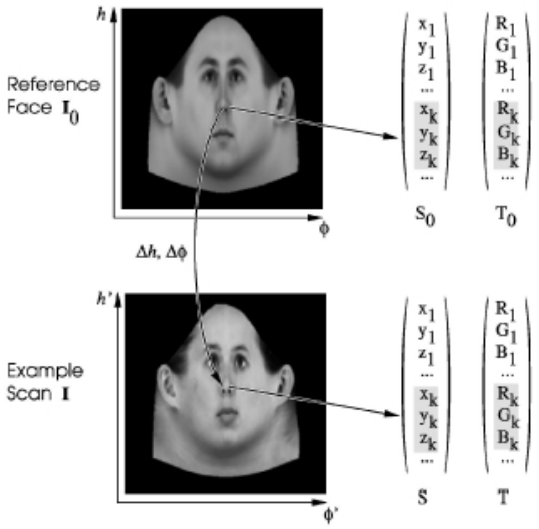

Abbildung 3.10: Korrespondenzen zwischen zwei Laserscans [11]

Laserscans anwenden zu können, wurde die Fehlerminimierung des originalen, gradientenbasierten Algorithmus (vgl. 2.1.3.2) modifiziert:

$$E = \sum_{(h,\phi)\in R} \|v_h \frac{\delta I(h,\phi)}{\delta h} + v_\phi \frac{\delta I(h,\phi)}{\delta \phi} + \Delta I\|^2.$$

Die Norm mit den Gewichtungen w_r, w_R, w_G, w_B

$$\|I\|^2 = w_r r^2 + w_R R^2 + w_G G^2 + w_B B^2$$

integriert die Komponenten der Messdaten. Die Gewichtungen kontrollieren dabei die Integration der Objekt- im Gegensatz zur Texturinformation. Da in [11] ein Morphing Modell erstellt wird, das aus verschiedenen Individuen besteht (mit verschiedenen Beleuchtungen), wurde der hierarchische, gradientenbasierte Algorithmus mit Laplace Pyramiden verwendet.
Der optische Fluss erzeugt fehlerhafte Werte in Regionen des Gesichts, in denen Objekt- und Farbinformation einheitlich sind. Ein spezieller Glättungs- und Interpolations- algorithmus verbessert das Vektorfeld auf jeder Ebene der Pyramide. Deshalb wird das Vektorfeld von Regionen mit hoher Varianz (Augen, Mund) zu Regionen mit niedriger Varianz interpoliert. Dieses geglättete Vektorfeld v_s wird durch die Minimierung einer Energiefunktion berechnet [11]:

$$E = \eta E_c + \sum h, \phi E_0(h, \phi).$$

E_c ist dabei ein Energieterm, der die Summe der Differenz aller benachbarten Beschleunigungswerte von v_s berechnet. Dieser wird gewichtet in die obige Gleichung eingebracht. Der zweite Energieterm E_0 definiert das Ausmaß einer Verbindung zwischen dem originalen Beschleunigungsfeld v_0 und v_s basierend auf dem Rang der Matrix W (vgl. 2.1.3.2). Der Rang bestimmt, ob das Gleichungssystem eine eindeutige Lösung hat oder nicht. Die Energieminimierung wird durch die konjugierte Gradientenmethode aus [36] berechnet und liefert als Ergebnis das geglättete und interpolierte Beschleunigungsfeld v_s.
Nachdem das finale Beschleunigungsfeld berechnet wurde, wird der Laserscan mit Hilfe des Referenzscans neu definiert. Um einen neuen Scan I_1 zum Morphing Modell hinzuzufügen, wird das Beschleunigungsfeld zwischen Refernzscan I_0 und I_1 berechnet. Auf Basis des Beschleunigungsfeldes werden die Shape- und Texturvektoren umsortiert. Für jeden Vertex $k \in 1, ..., n$ des Meshs an der Position $(h_k, \phi_k, r(h_k, \phi_k))$ wird die Beschleunigung an diesem Punkt addiert:

$$h'_k = h_k + (\Delta h_k, \phi_k), \quad \phi'_k = \phi_k + v_\phi(h_k, \phi_k).$$

3.2.4 Übertragung

Die Übertragung von Gesichtsausdrücken ermöglicht die Animation verschiedener Gesichtsmodelle mit demselben Gesichtsausdruck.
In [31] werden sogenannte Bewegungsvektoren eines Quell-Gesichtsmodells auf ein Ziel-Gesichtsmodell mit unterschiedlicher Vertex-Anzahl und Konnektivität transferiert (Abb. 3.11). Da die Gesichtsgeometrien und Proportionen der Modelle stark variieren können, werden Quell-Bewegungsvektoren in Richtung und Länge auf das Zielmodell angepasst (Motion Vector Direction Adjustment). Dabei wird

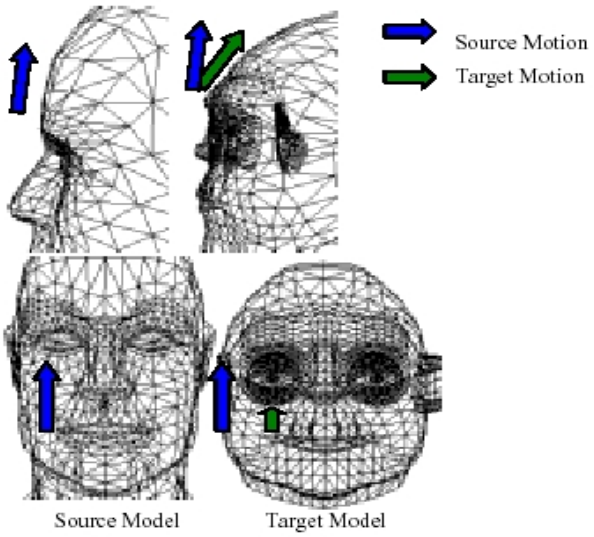

Abbildung 3.11: Anpassung der Richtung und der Länge von Bewegungsvektoren [31]

die generelle Form der Modelle beibehalten. Bei einem Motion Vector Transfer befinden sich beide Modelle durch das in Abschnitt 3.2.3.1 beschriebene Verfahren der RBF-Netze in voller Korrespondenz. Jedem Vertex des Quell- und des deformierten Zielmodells wird ein lokales Koordinatensystem zugeordnet. Die Transformation zwischen diesen beiden Koordinatensystemen definiert das Motion Vector Direction Adjustment. Für einen gegebenen Motion Vector $\vec{m}$ wird der deformierte Vektor $\vec{m'}$ durch die Transformationsmatrizen zwischen den beiden lokalen Koordinatensystemen und dem Weltkoordinatensystem berechnet. Die Anpassung der Länge der Bewegungsvektoren kann bei relativ ähnlichen Gesichtsmodellen über eine einfache Skalierung berechnet werden. Bei Modellen mit großen geometrischen Unterschieden wird die Länge jedes Motion Vectors durch einen lokalen Skalierungsfaktor innerhalb eines globalen Schwellwerts berechnet. Die lokale Skalierung wird durch eine Bounding Box um alle Polygone des gemeinsamen Vertex bestimmt. Die Quell-Bounding Box wird durch die vorher berechnete Rotationsmatrix transformiert. Außerdem wird für alle Vertizes des Quellmodells in einer Bounding Box die rotierte Position der Deformation berechnet. Die lokale Skalierungsveränderung ist das Verhältnis zwischen der Bounding Box des Quellmodells und der deformierten Bounding Box.
Ein anderes Verfahren wird in [37] verwendet. In diesem Ansatz werden Gesichtsausdrücke durch Muskelparameter in einem physikalischen Modell animiert. Das sogenannte Expression Retargetting (oder auch Muscle Parameter Transfer) basiert auf der Annahme, dass das Zielmodell das gleiche Muskelprofil besitzt wie das Quellmodell. Daher können die Muskelparameter zwischen den verschiedenen Modellen ausgetauscht werden.

3.2.5 Analyse und Zusammenfassung

Aufgrund der Anforderungen an das System müssen die Verfahren analytisch betrachtet werden. Das im System umzusetzende Verfahren führt die Berechnungen offline aus. Daher wurden die Ansätze nicht nach ihrer Schnelligkeit bewertet. Wichtige Faktoren für die Wahl der geeigneten Algorithmen sind vor allem

- Automatisierung,
- Anpassungsmöglichkeit an gegebene Datensätze,
- Robustheit.

Der verwendete Scanprozess (Anhang A) liefert 3D Daten, die transformiert im Raum liegen. Ein geeignetes Alignment-Verfahren soll die hochaufgelösten Meshes so transformieren, dass die unbeweglichen Regionen des Gesichts (z.B. Stirn, Nasenspitze) im 3D Raum übereinander liegen. Daher ist eine vollständige Registrierung auf beide Modelle nicht nötig, das Alignment muss nur auf das gesamte Referenzmodell und eine statische Region des Ausdrucksmodells angewendet werden.
Blanz und Vetter verwenden in ihrem Ansatz das Alignmentverfahren Absolute 3D-3D Orientation (vgl. 3.2.2.3). Absolute Orientation bestimmt die Rotationsmatrix R und den Translationsvektor t durch korrespondierende Punktepaare p_i und q_i. Dies wird durch mehrere nicht-kollineare Referenzpunkte erreicht.
Der Iterative Closest Point Algorithmus und der Trimmed Iterative Closest Point können auf Basis eines *Initial Guess* große Datensätze automatisch registrieren. Dabei ist eine aufwendige Selektierung von Featurepaaren nicht nötig.
Der Iterative Closest Point (ICP) Algorithmus registriert ein Objekt A auf ein Objekt B, indem er eine geeignete Transformation schätzt. Ein Vorteil dieses Algorithmus ist, dass er unabhängig von der Darstellung der Form ist. Er registriert sowohl Punkte, Linien, Kurven, als auch Dreiecke und Polygone. Es ist möglich, dass der Algorithmus nicht die korrekte Transformation bei der Registrierung finden kann. Er konvergiert gegen ein lokales Minimum der Fehlerfunktion, das nicht das globale Minimum sein muss.
Der Trimmed ICP Algorithmus ist ein schnelles, für fehlerhafte Daten robustes Verfahren zur Registrierung von Objekten. Er kann bei Überschneidungen der Objekte von bis zu 50% angewendet werden. Er benutzt die Summe der LTS der individuellen Abstände, um mit den Formdefekten und den teilweisen Überschneidungen zurecht zu kommen. Er braucht mehrere Iterationen als der ICP Algorithmus ist aber robuster und registriert die Objekte mit einer größeren Genauigkeit.
Zur Bestimmung von Punkt-zu-Punkt Korrespondenzen wurden drei komplexe

Verfahren erläutert, die in der Gesichtsanimation für Morphing Modelle verwendet werden. Die verbreiteten RBF-Vefahren und das Harmonic Mapping sind modellbasiert und berechnen die Korrespondenzen ausschließlich durch die Geometrie der Modelle. Die Methode von Noh und Neumann ist eine effiziente Technik. Das Volume Morphing mit dem interpolierenden RBF-Netz berechnet ein initiales Oberflächenmatching, das durch eine zylindrische Projektion verfeinert wird.
Beim Harmonic Mapping entstehen Schwierigkeiten, wenn spezielle Punkte zwischen den Modellen gematcht werden müssen. Ein naives Harmonic Mapping könnte die Polygone einfach vertauschen, wenn der Benutzer die Nasenspitzen oder Mundwinkel der Modelle matchen will [31]. Daher ist es wichtig, die Modelle zu partitionieren oder zu simplifizieren, bevor das Harmonic Mapping angewendet wird (vgl. 3.2.3.2). Harmonic Mapping und das Volume Morphing durch RBF-Netze sind Verfahren, die zwar auf sehr unterschiedliche Geometrien angewendet werden können, die Idee basiert aber auf dem Vorhandensein von Animationsdaten für das Quell-Basismodell. Die Punkt-zu-Punkt Korrespondenzen werden zunächst auf Basis der neutralen Modelle berechnet. Danach werden die Animationsdaten als Bewegungsvektoren des Quell-Basismodells an das Zielmodell angepasst (vgl. 3.2.4).
Der optische Fluss von Blanz und Vetter ist ein hybrides Verfahren, das geometrische und texturelle Information für die Berechnung der Korrespondenzen integriert. Werden ähnliche Modelle verwendet, in denen geometrische und farbliche Information nicht zu stark variieren, kann durch die Verwendung von Gauß Pyramiden ein detailliertes Beschleunigungsfeld berechnet werden. Außerdem besteht die Möglichkeit, durch Laplace Pyramiden, in denen nur Kanteninformationen enthalten sind, Korrespondenzen zwischen verschiedenen Modellen zu bestimmen. Zudem ist eine manuelle Spezifikation korrespondierender Punkte nicht nötig.

Da die Modelle im Vorverarbeitungschritt manuell vorregistriert werden müssen, ist die Wahrscheinlichkeit für einen falschen Match innherhalb der Alignmentphase sehr gering. Zudem wird das Alignment auf eine Region angewendet, die in beiden Modellen annähernd gleich ist. Diese statische Region kann als die benötigte Untermenge für den ICP verwendet werden. Der Iterative Closest Point Algorithmus liefert daher eine verlässliche Basis für eine Registrierung.
Die Bestimmung von Punkt-zu-Punkt Korrespondenzen zwischen den registrierten Modellen sollte automatisiert sein. Bei einer Verwendung von Harmonic Mapping oder RBF Netzen müssen für jeden Gesichtsausdruck korrespondierende Punkte selektiert werden. Da der Scanprozess hochaufgelöste Meshes und Texturen liefert, bietet sich ein hybrides Verfahren an, das Geometrie und Textur in den Berechnungen berücksichtigt. Der optische Fluss kann außerdem durch verschiedene Filtertechniken das Problem der Übertragung lösen, wobei allerdings

eine manuelle Skalierung des zu übertragenen Scans auf das generische Zielmodell nötig ist.
Die analytische Betrachtung der vorgestellten Verfahren hat ergeben, dass der Ansatz von Blanz und Vetter in höchstem Maße die Anforderungen an das zu implementierende System erfüllt. Zudem wird dieser Ansatz durch eine Registrierung der Modelle durch den Iterative Closest Point Algorithmus unterstützt. Daher orientiert sich die folgende Entwicklung eines Konzepts an diesen Ansätzen.

3.2.6 Konzept

Aus den Schlussfolgerungen der analytischen Betrachtung der verschiedenen Techniken zur Generierung personalisierbarer Morphtargets wird nun ein Konzept entwickelt. Die umgesetzten Verfahren orientieren sich an den Ansätzen von Blanz und Vetter, die in den vorherigen Abschnitten vorgestellt wurden. Die konzeptionelle Pipeline zur Erstellung und Verarbeitung von 3D Scans bis hin zum Morphing Modell beinhaltet manuelle sowie automatische Vorverarbeitungsschritte, in denen die Rohdaten auf den Kernprozess vorbereitet werden. Die generelle Pipeline aus Abschnitt 3.2.1 wird nun durch die spezifischen Komponenten der Algorithmen erweitert.

Erzeugung der Datenbasis und Vorverarbeitung

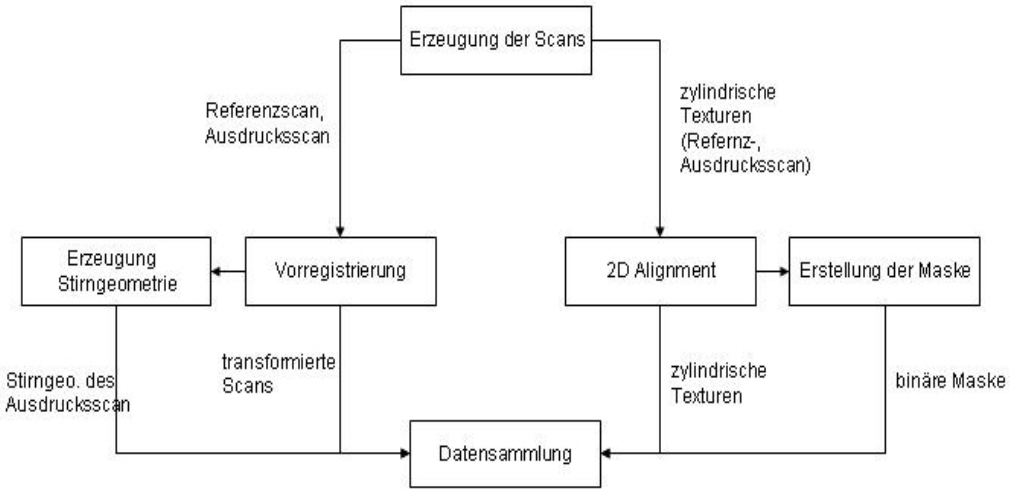

Abbildung 3.12: Datenbasis und Vorverarbeitung

In der Erzeugung der Datenbasis werden personalisierbare Kopfmodelle einer Person produziert. Das Resulat dieses Moduls ist ein Referenzmodell als komplettes Kopfmodell sowie Gesichtsmodelle mit verschiedenen Gesichtsausdrücken. Für

jedes Modell existiert zudem eine hochaufgelöste zylindrische Textur. Innerhalb der Vorverarbeitungsphase finden ein manuelles 2D-Alignment und eine manuelle Vorregistrierung der 3D Modelle statt. Das Vorverarbeitungsmodul sichert ein robustes Verhalten des anschließenden Registrierungsverfahrens und der Zuordnung von Korrespondenzen.

Globales Alignment und Korrespondenzen

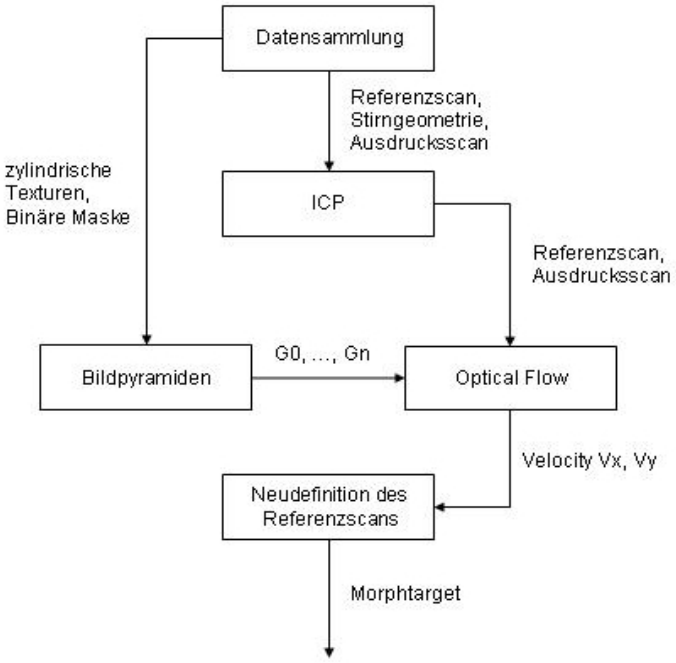

Abbildung 3.13: Registrierung und Punkt-zu-Punkt Korrespondenzen

Das Alignment- und das Korrespondenzmodul werden automatisch nacheinander auf die Modelle angewendet. Das globale Alignmentverfahren (ICP) erwartet das Referenzmodell, einen Ausdrucksscan und die erzeugte statische Region des Ausdrucksscans als Eingabe. In einem iterativen Verfahren wird die optimale Rotation und Translation ermittelt, die zuletzt auf den Ausdrucksscan angewendet wird. Der optische Fluss erwartet alle zylindrischen Texturen, Referenz- und Ausdrucksmodell sowie eine binäre Maske als Eingabe. Bildpyramiden werden aus den Eingabebildern konstruiert und an das Korrespondenzmodul übergeben. Darin werden die Punkt-zu-Punkt Korrespondenzen zwischen den Modellen und Texturen berechnet. Der letzte Schritt ist eine Neudefinition des Referenzscans auf Basis der berechneten Korrespondenzen. Aus der Neudefinition entsteht das entsprechende Morphtarget, das dem Morphing Modell hinzugefügt werden kann.

Übertragung auf generische Modelle

Um die Gesichtsausdrücke auf ein generisches Modell zu übertragen, durchläuft das generische Modell dieselbe Pipeline, die im letzten Abschnitt vorgestellt wurde. Dabei wird das generische Modell als Referenzmodell verwendet. Im ersten Schritt wird der neutrale Referenzscan auf das generische Referenzmodell übertragen. Dadurch entsteht ein generisches Modell, das die Form des neutralen Referenzscans annimmt. Im zweiten Schritt werden die Korrespondenzen zu allen Morphtargets berechnet.

3.3 Kontrollierbarkeit

In Abschnitt 2.4 der theoretischen Grundlagen wurden grundlegende Constraint-Techniken vermittelt, die in der Animation Verwendung finden. Constraints werden in Echtzeit während der Animation in physikalischen Simulationen (vgl. 2.4.1) oder interaktiv in einem Motion Editing Prozess (vgl. 2.4.3) eingesetzt.
Dieser Abschnitt beschäftigt sich mit der Konzipierung eines Constraintsystems für die Gesichtsanimation. Ausgehend von dem im letzten Kapitel entwickelten Konzept, ist die Basis weiterer Ausführungen ein Scan-basiertes oder generisches Morphing Modell.
Im Folgenden werden zunächst generelle Möglichkeiten zur Kontrolle von Animationsvorgängen genannt. Anschließend wird spezieller auf Kontrollmechanismen im Tracking eingegangen. Das Gesichtstracking war der Auslöser für die Entwicklung weiterer analytischer, bildbasierter Verfahren, die in Abschnitt 3.3.2 näher betrachtet werden. Darauf aufbauend wird schließlich ein Konzept entwickelt.

3.3.1 Kontrolle von Animationen

Die kontrollierte Animation wird durch Kontroll-Schemata umgesetzt, die die Animation parametrisch beschreiben. Der Animationsprozess wird durch spezifierung von Kontrollparametern in einer zeitlichen Funktion ausgeführt. Die automatische Erzeugung von Kontrollparametern basiert auf unterschiedlichen Verfahren: Diese sind Regel-, Analyse- und Performance-basiert [9]. Durch Integration dieser Verfahren in ein Animationssystem findet eine Kontrollierbarkeit in der *Erzeugung* der Animation statt.

Regel-basierte Verfahren fundieren auf linguistischen und psychologischen Studien. Eine Menge von Regeln beschreibt die Verbindung zwischen Intonation, Emotionen und anderen Gesichtsausdrücken [38]. Durch Multi-Layer Strukturen können verschiedene synchrone Effekte wie Lippenbewegungen, konversationale Signale und Emotionen charakterisiert werden[39].

Analyse-basierte Methoden bestehen aus der Extraktion von Informationen aus Videosequenzen oder Bildern und der Eingabe dieser Informationen in das Animationssystem. Solche Informationen sind Muskelkontraktionen oder Action Units des FACS (vgl. 2.3.1). Die automatische Bestimmung der Gesichtsparameter ist schwierig, aufgrund von Feinheiten und Komplexität der Deformationen des Gesichts und Korrelationen zwischen Muskeln.

Performance-basierte Verfahren tracken diverse Punkte des Gesichts in Videosequenzen und übertragen in einem Sythesemodul diese Parameter auf ein Gesichtsmodell.

Kontrollarchitekturen, die als Kontrollmechanismen in der Gesichtsanimtion verwendet werden, generieren die kontrollierte Animation des Gesichts *zur Laufzeit*: *Muskelmodelle* definieren abstrakte Muskeln, die Gesichtsausdrücke durch Kontrollparameter (Position, Einflussregion, Kontraktionsprofile) geometrisch kontrollieren (vgl. 2.3.3.2).
Physikalische Modelle bewegen sich auf mehreren Abstraktionsebenen basierend auf der Psychologie der menschlichen Gesichtsausdrücke, der Anatomie der Gesichtsmuskeln, der Biomechanik der Gewebestrukturen und auf kinematischen Eigenschaften (vgl. 2.3.3.4). Auf der höchsten Abstraktionsebene befindet sich der Gesichtsausdruck. Das Gesichtsmodell führt Ausdrucks- und Phonemkommandos in einem definierten Zeitintervall aus. Die zweite Ebene stellt eine Kontrollebene dar, in der eine Untermenge der AUs des FACS in einen Muskelkontrollprozess einbezogen wird. Der Kontrollprozess übersetzt Ausdrucks- oder Phoneminstruktionen in eine koordinierte Aktivierung von Aktuatorgruppen im Gesichtsmodell. Die Muskelebene ist der Animationsmechanismus des Modells. Die physikalische Ebene realisiert die Approximation der Hauteigenschaften durch Massepunkte und nichtlineare Federn. Die Geometrieebene repräsentiert die geometrische Form des Gesichtsmodells [40].
Linguistische Modelle generieren Gesichtsausdrücke, Kopf- und Augenbewegungen automatisch aus gesprochener Eingabe. Eine gesprochener Äußerung mit assoziierter Intonation und Emotion kann unabhängig vom Gesichtsmodell berechnet werden. Über daraus extrahierte AUs des FACS wird das Gesichtsmodell animiert. Phoneme werden durch ihre Deformierbarkeit charakerisiert. Ein Koartikulationsmodell [41] untersucht benachbarte Phoneme in der Phonemsequenz, die das aktuelle Visem in seiner Form beeinflusst (Look-Ahead Model). Gleichzeitig werden Sequenzen von Muskelkontraktionen angepasst, wenn Bewegungen von Synergisten (Anhang B) aufeinander folgen. Außerdem wird automatisch erkannt, ob ein Muskel vor oder nach einem Phonem genug Zeit für eine Kontraktion oder Relaxation hat. Falls die Zeit zwischen zwei aufeinander folgenden Phonemen kleiner als die Kontraktionszeit eines Muskels ist, wird das vorherige Phonem durch die Kontraktionen des aktuellen Phonems beeinflusst. Daraus wird ein funktionale Gruppe entwickelt, die aus Lippenformen, koversationalen

Signalen, Punktuatoren, Regulatoren und Manipulatoren besteht. Verschiedene Algorithmen synchronisieren daraus Bewegungen. Am Ende dieser Berechnungen entsteht eine Liste von AUs für jedes Phonem [38].

3.3.1.1 Kontrollgeometrie

Performance-basierte Verfahren wie Tracking sind häufig verwendete Ansätze, um Gesichtsausdruck und Sprache aus Videomaterial zu extrahieren.
Der einfachste Weg, die Features des Gesichts zu tracken, ist, die Marker direkt auf dem Gesicht des Akteurs zu platzieren. Durch Aufnahme der Gesichtsausdrücke können bildbasierte Methoden die Standorte der Marker in der Bildebene berechnen [42, 43, 44]. Zwei grundlegende Probleme sind in der Synthese zu lösen [44]:

1. Das sogenannte Retargetting der Motion Capture-Daten ist eine Anpassung der Daten auf das Zielmodell,

2. Die Generierung von Bewegungsvektoren für alle Vertizes im Zielmodell.

Retargetting-Verfahren von Motion-Capture (MoCap) Daten verwenden Radiale Basisfunktionen als eine verteilte Daten-Interpolationsmethode (Scattered Data Interpolation). RBFs liefern ein Mapping zwischen MoCap-Daten und Zielmodell. Durch eine Energieminimierung wird die optimale Platzierung der MoCap-Punkte auf die Oberfläche des Zielmeshes definiert.
Für die Animation wird ein Kontrollmesh verwendet, das aus einer Triangulierung der MoCap-Punkte gewonnen wird [44]. Diese Modelle werden als einfache

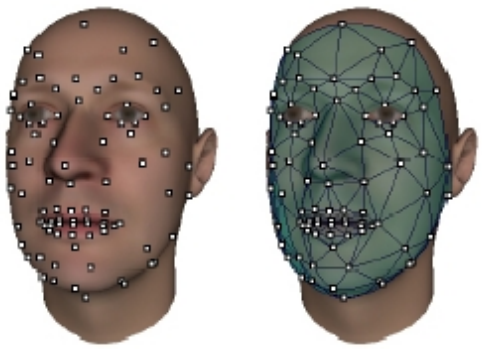

Abbildung 3.14: Kontrollstruktur: MoCap Punkte (links) und Kontrollmesh (rechts) [43]

Kontrollmodelle in die Gesichtsgeometrie integriert und deformieren (warpen) die feiner strukturierte Geometrie durch die Methode der sogenannten *Planar Bone Deformations*[44] oder der *Bezier-triangle Induced Deformation (BID)* [43]. Planar Bones Deformations basiert auf einer linearen Achsendeformation (Bone Deformation), verwendet aber dreieckige Kontrollelemente, um die Deformation auf eine darunterliegende Geometrie auszuüben. Die Kontrollmeshes sind definiert durch ihre Dreiecke als Kontrollelemente (Bones) und die Verschiebungen der einzelnen Punkte. Die feinere Geometrie wird partitioniert, um zu bestimmen, welche Vertices von den Bones beeinflusst werden. Dem Kontrollelement wird ein umgebendes Volumen hinzugefügt, mit dessen Hilfe die Partitionierung durchgeführt wird. Um eine Distanz zwischen Bone und Vertex bestimmen zu können, müssen mehrere Fälle betrachtet werden. Eine initiale Klassifizierung wird mittels baryzentrischer Koordinaten durch eine Projektion der Vertizes auf die Kontrollelemente berechnet und somit einer Region des Bones zugeordnet. Aus dem Kontrollelement, dem verschobenen Kontrollelement und dem projizierten Vertex lassen sich Transformationsmatrizen für den entsprechenden deformierten Vertex berechnen. Auch BIDs konstruieren aus den MoCap-Punkten eine Triangulierung der Kontrollpunkte und definieren ein eins-zu-eins Mapping der Zielgeometrie auf das Kontrollmesh. Über die Normale des nächstgelegen Vertizes wird das Zielmodell auf das Kontrollmodell projiziert. In einer Rekonstruktionsphase wird das deformierte Zielmodell durch die veschobenen MoCap-Punkte und der Verformung des Kontrollmeshes entsprechend angepasst.

3.3.2 Interessante analysebasierte Verfahren

Parke [9] unterteilt analysebasierte Verfahren in zwei Kategorien: Die *modellbasierte* Analyse nimmt an, dass eine Geometrie vorhanden ist, auf die das Bild gematcht wird. *Bildbasierte* Verfahren fundieren ausschließlich auf den Parametern der Bildebene.
Das Prinzip der Deformable Templates basiert auf Features des Gesichts, die durch eine Bildanalyse eindeutig identifizierbar sind. Ein Template kann die Form durch Modifikation der Parameter, die das Template geometrisch beschreiben, definieren. In einer Matchingprozedur wird die Parameterbelegung für das Template aus einem Bild durch Kanten, Valleys und Peaks aus den Bildintensitäten extrahiert. Durch Kombination dieser Werte entsteht eine Energiefunktion über den Parametern des Templates, die anschließend minimiert wird.
Die Methode der Active Contours ähnelt dem Deformable Template, reduziert das Problem aber auf zweidimensionale Kurven oder Splines, die Linien und Konturen im Bild tracken. Discrete Deformable Contours sind durch eine Menge von Knotenpunkten in der Bildebene definiert, die durch Federn miteinander verbunden sind. Die deformierbare Kontur wird durch Kräfte aus einer zeitlichen Bildfolge in Form und Bewegung beeinflusst. Die Kräfte des Bildes basieren auf der Berechnung der Länge von lokalen und temporären Gradienten (vgl. 2.1.2)

der Bildintensitäten. Aus den deformierbaren Konturen werden Kontraktionswerte sowie Kieferrotationen für die Animation extrahiert.
Essa [10] verwendet den optischen Fluss (vgl. 2.1.3) zum tracken von Pixeln oder Gruppen von Pixeln von Frame zu Frame. Das resultierende Velocity-Feld erlaubt es, Muskelaktionen zu schätzen und zusätzlich Gesichtsausdrücke zu klassifizieren. Daraus wurde ein einfacher Gesichtserkenner entwickelt, der bestimmte Muster von Bewegungsenergien identifiziert und mit einem Gesichtsausdruck assoziiert.
Kenji Mase [45] veröffentlichte eine weitere Arbeit zur Erkennung von Gesichtsausdrücken durch Optical Flow. Aus Grauwertbildern werden automatisch Muskelbewegungen extrahiert, die als Action Units in Kombination mit Feature Positionierungen interpretiert werden können. In diesem Ansatz wurden Action Units aus dem Facial Action Coding System als eine geeignete Basis für die Kategorisierung von Gesichtsausdrücken unter Verwendung des Wissens über die Anatomie des Gesichts integriert. Da Gesichtsausdrücke durch Kontraktionen der Gesichtsmuskeln geformt werden, enthält die Deformation der Haut wichtige Informationen über Muskelaktionen. Muskelkontraktionen werden durch einen Nervimpuls getriggert, die durch Emotionen geneneriert werden. Da viele Muskeln, die für Gesichtsausdrücke verantwortlich sind, mit bestimmten Features im Gesicht verbunden sind (Lippen, Augenpartie), werden entsprechende Features verformt. Mase verwendet den gradientenbasierten Ansatz von Horn und Schunck (vgl. 2.1.3.1), um Gesichtsbewegungen in den Feature-Regionen der Grauwertbilder zu extrahieren. Obwohl die Haut ein nicht-rigides, deformierbares Objekt ist, kann das lokale Bewegungsfeld als glatt angenommen werden.
Folgende Gesichtsmusklen wurden primär untersucht:

1. M. Orbicularis Oris um den Mund,
2. M. Venter Frontalis an der Stirn,
3. M. Buccinator,
4. M. Zygomaticus,
5. M. Anguli Oris.

Damit der optische Fluss Muskelbewegungen berechnen kann, werden auf den Grauwertbildern Fenster definiert, die die Hauptrichtung von Muskelkontraktionen bzw. Hautdeformationen beschreiben. Jedes Fenster definiert seine Größe S als Anzahl der überdeckten Pixel und einen Richtungsvektor n:

$$m_{i,t} = \frac{1}{S_i} \int < u_t(x), n_i > dx,$$

wobei $u_t(x) = (u_t(x,y), v_t(x,y))$ das berechnete Beschleunigungsfeld zwischen zwei aufeinander folgenden Frames zur Zeit t und $t+1$ ist.
Die Fenster wurden manuell auf den Bildern durch bestimmte Features des Gesichts lokalisiert. m_i ist die geschätzte Muskelbewegung im i-ten Fenster.

3.3.3 Analyse und Zusammenfassung

Auf Basis der Behandlung von Constraints sowie der Ansätze zur Realisierung von Kontrollmechanismen in der Animation wird nun eine analytische Betrachtung der beschrieben Techniken durchgeführt.
Die in der Motivation aufgestellte Definition von Constraints ist entscheidend für die weitere Konzeptionierung. Die Anforderungen an das zu implementierende Verfahren sind

- Schnelligkeit,
- Integrierbarkeit,
- Dynamik.

Das Constraintsystem wird während der Animation aktiviert und sollte den Animationsprozess nicht negativ durch niedrige Frameraten beeinträchtigen. Ein weiterer wichtiger Aspekt ist die Integrierbarkeit in die verwendete Animationstechnik, sodass eine geometrische Korrektur der Animation zur Laufzeit möglich ist. Die Dynamik setzt voraus, dass sich das Constraintsystem dynamisch an den Animationsprozess, der eine Vielzahl von Morphtargets verwenden kann, anpasst und automatisch realistische Grenzen erzeugt.
Die Definition von Constraints im physikalischen Sinne oder die interaktive Gestaltung von Constraints liefern interessante Informationen über verschiedene Lösungswege. Generell werden Constraints durch physikalisch realistische Bewegungsgleichungen definiert. Die Gesichtsanimation unterscheidet sich stark von den üblichen computergrafischen Modellen, da z.B. keine physikalischen Bewegungsgleichungen verwendet werden.
Die bisher verwendeten Constraint-Techniken sind aufgrund der Definition von Constraints in der Motivation nicht verwendbar. Es werden Verfahren benötigt, die automatisch während des Morphings in die Animation eingreifen und bei Erreichen eines Constraints das Verhalten des Morphings beeinflussen. Zudem müssen realistische Grenzen gefunden werden, die an personalisierbaren Informationen festgemacht werden können. Innerhalb der vorhandenen Animationsmethoden und bildbasierten Verfahren ist ein Kontrollmechanismus und eine Möglichkeit der Kombination zu finden, die in den Morphingprozess integriert werden können.
Realistische Animation wird innerhalb natürlicher Grenzen durch Muskelsimulationen (vgl. 2.3.3.4) generiert. Die natürlichen Grenzen des Gesichts sind die Gesichtsmuskeln selbst: Die Kombination von Kontraktionen verschiedener Muskeln und Muskelgruppen erzeugen Emotionen. Die Kombination von Kieferrotation und Muskelkontraktionen erzeugen Viseme.
Die Verwendung von Kontrollmodellen zur Übertragung der MoCap-Daten auf die feiner strukturierte Geometrie (vgl. 3.3.1.1) ist ein guter Ansatzpunkt für

das zu implementierende System. Ein Feature- oder Muskel-basiertes Kontrollmodell kann mit personalisierbarer Information initialisiert werden und während der Animation entsprechend diesen Informationen reagieren. In physikalischen Muskelsimulationen wird die Deformation des Gesichts durch Spezifizierung der Muskelkontraktion kontrolliert. Da eine physikalische Simulation von Muskeln hinter einem Morphingmodell zu berechungsintensiv wäre, liefern parametrische Verfahren (vgl. 2.3.3.3) den gewollten Effekt, wobei nicht die Erzeugung von Animationen, sondern der Kontrollmechanismus, der aus der Animation Informationen über den aktuellen Gesichtsausdruck beziehen und diesen gegebenenfalls einschränken kann, im Vordergrund steht.
Die Idee der Muskelvektoren mit entsprechenden Einflussregionen auf dem Gewebe kann in einem muskelbasierten Kontrollmodell umgesetzt werden, das während der Animation pro Frame aktualisiert wird. Das Kontrollmodell benötigt eine realistische Datenbasis, um zwischen erlaubten und falschen Gesichtsausdrücken unterscheiden zu können. Diese Basis kann aus Mases Erkennungsverfahren durch Optical Flow realisiert werden. Dieses System ist FACS-basiert und erzeugt für jedes Muskelfenster Kontraktionswerte, die auf ein Muskelmodell übertragen werden können. Weiterhin kann das System durch Regeln (vgl. 3.3.1) erweitert werden, die das Verhalten einzelner Muskeln regulieren können.
Aufgrund der Definition von Constraints entspricht ein parametrisches Muskelmodell in Kombination mit dem analysebasierten Verfahren von Mase den beschriebenen Anforderungen.

3.3.4 Konzept

Auf Basis der analytischen Betrachtung des letzten Abschnitts wird im Folgenden das Konzept des Constraintsystems entwickelt. Das System gliedert sich in drei Module, die anhand von Flussdiagrammen dargestellt werden:

1. Update der Datenbasis,
2. Initialisierung,
3. Animationsprozess.

Das Modul *Update der Datenbasis* kann jederzeit offline zur Integration neuer personalisierbarer Constraints ausgeführt werden und stellt die Realisierung des analysebasierten Verfahrens von Mase dar. Die *Initialisierung* speichert Constraints für den schnellen Zugriff und integriert das parametrische Kontrollmodell in das Kopfmodell. Nach der Initialisierung startet die Animation und das Constraintsystem wird aktiviert.

Update der Datenbasis

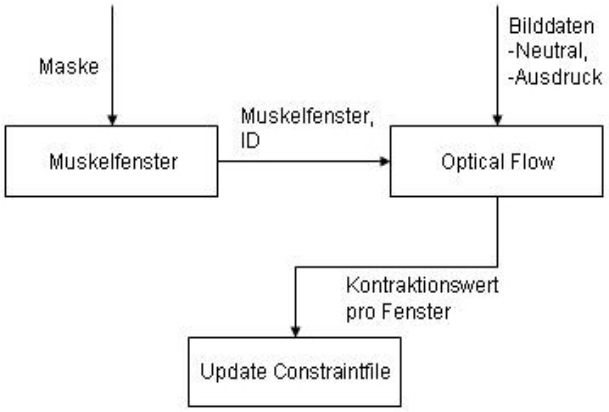

Abbildung 3.15: Updatemodul zur Erzeugung personalisierbarer Constraints

Das Updatemodul betrachtet pro Constraint zwei Grauwertbilder derselben Person mit neutralem und einem beliebigen Gesichtsausdruck. Zusätzlich werden durch eine spezielle Maske orientierte Muskelfenster definiert, die im Featurebasierten Verfahren die Parameter für das Constraintsystem liefern (vgl. 3.3.2). Die berechneten Kontraktionswerte werden anschließend der Datenbasis hinzugefügt.

Initialisierung

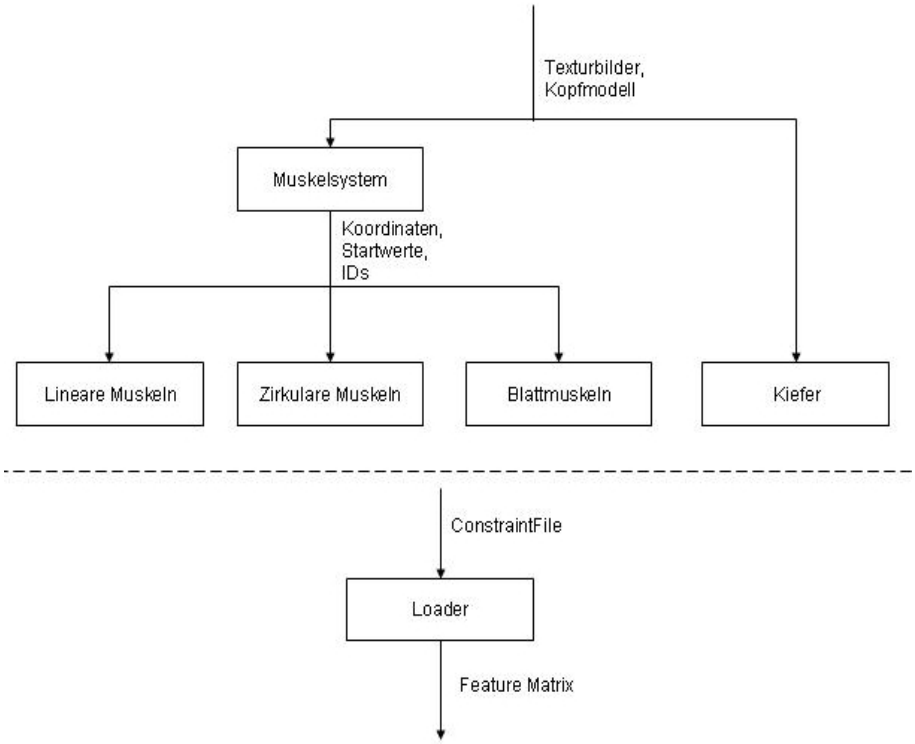

Abbildung 3.16: Initialisierungsmodul zur Integration der Muskeln in das Modell

Das Initialisierungsmodul erwartet mehrere Texturbilder sowie die Datenbasis des Constraintsystems als Eingabe. Aus den Texturbildern wird das dreidimensionale Muskelmodell innerhalb eines Muskelsystems berechnet und mit IDs und Startwerten initialisiert. Das Muskelsystem ist für die Verwaltung von linearen und zirkularen Muskeln sowie Blattmuskeln zuständig. Gleichzeitig wird die Datenbasis geparst und für einen schnellen Zugriff als eine Feature-Matrix gespeichert.

Animationsprozess

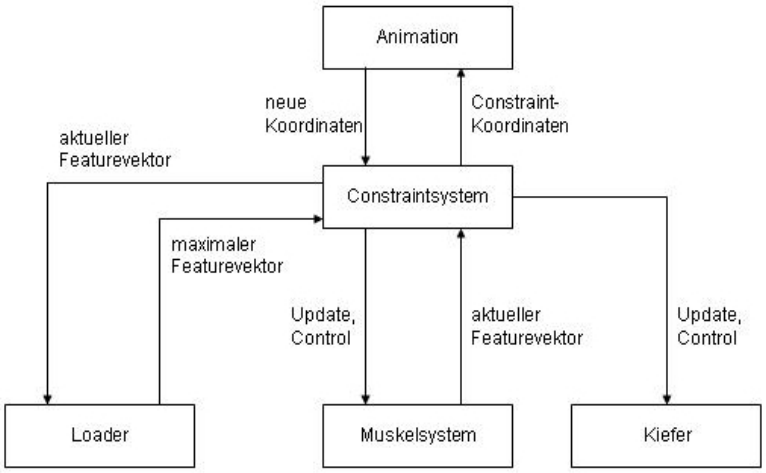

Abbildung 3.17: Kontrolle während der Animation

Das Constraintsystem wird während einer Animation pro Frame einmal aufgerufen. In jedem Frame werden zunächst alle durch das Morphing geänderten Parameter des Kontrollmodells neu berechnet. Das Constraintsystem führt einen Update des Muskelsystems durch und baut einen Featurevektor der *aktuellen* Muskelkonfiguration auf. Die aktuelle Muskelkonfiguration wird aus den Muskelpunkten der verschiedenen Muskeln im Mesh berechnet. Dieser Vektor wird mit denen aus der Feature-Matrix verglichen und liefert einen nächsten Featurevektor, dessen Werte minimale oder maximale Zustände der einzelnen Muskeln definieren. Mit diesen Werten wird eine Kontrollfunktion aktualisiert, die bei einer Überschreitung der Werte in den Animationsprozess eingreift.

Kapitel 4

Realisierung

Nach der theoretischen Auseinandersetzung in Kapitel 2 folgt nun die praktische Umsetzung der in Abschnitt 3.2.6 und 3.3.4 entwickelten Konzepte. Das Kapitel besteht aus zwei Hauptabschnitten, die die beiden Systemkomponenten separat beschreiben.
Die Implementierung der Module basiert auf OpenGl Performer und C++. Alle Klassen wurden in das Character Animation System JOSH integriert, das auf der VE Entwicklungsumgebung Avango (Anhang C) basiert.

4.1 Modul Personalisierbarkeit

Das Modul Personalisierbarkeit wird offline durchgeführt und enthält alle Methoden zur Generierung personalisierbarer Morphtargets. Das Resultat dieses Moduls ist laut Konzept (vgl. 3.2.6) ein Morphtarget, das aus zwei 3D Scans berechnet wird und auf ein generisches Modell übertragen werden kann. Im Folgenden werden Details zur Implementierung und Vorgehensweisen vermittelt.

4.1.1 Vorverarbeitung

In der Vorverarbeitungsphase werden alle nötigen Eingabedaten auf die Bestimmung von Punkt-zu-Punkt Korrespondenzen und die Generierung von Morphtargets vorbereitet.
Für die Vorverarbeitung wird ein vollständiges Datenset benötigt, bestehend aus zwei triangulierten 3D Scans (neutral und Ausdruck) und entsprechende zylindrische Texturen. Die Vorverarbeitungskette besteht aus einem manuellen und einem automatisierten Prozess. Zunächst müssen die 3D Modelle grob vorregistriert werden. Zusätzlich werden die Texturen in Adobe Photoshop passend übereinander gelegt und in Maya über den Texture Editor erneut auf die Modelle angepasst. Wie im Konzept bereits erwähnt wurde, erfolgt die Registrierung auf statische Elemente des Gesichts. Daher wird die Stirngeometrie des transformier-

ten Ausdrucksmodells ausgeschnitten. Alle 3D Daten werden als Wavefront OBJ exportiert.
Um die Daten in JOSH zu integrieren, müssen die Scans zusätzlich in Inventor Daten konvertiert und in einem Scheme Skript als .iv und .obj definiert werden. Die Texturbilder werden in RGB (Sgi) konvertiert und dem Scheme Skript hinzugefügt. In einer weiteren Textdatei wird der Ladevorgang der Scans und der Bilder definiert. Diese wird beim Laden des JOSH-Systems von einem Loader geparst und baut eine neutrale Scanbase sowie zwei hierarchische Strukturen aller übrigen Scans und Texturbilder auf.

4.1.2 Globales Alignment

Wie im Konzept beschrieben, ist die Aufgabe des globalen Alignments die Registrierung der Modelle in einem iterativen Prozess. Der Quaternionen-basierte Iterative Closest Point Algorithmus (vgl. 3.2.2.1) berechnet in wenigen Iterationsschritten die optimale Transformation zwischen der Stirngeometrie des Ausdrucksscans und der neutralen Scanbase. Die Iteration wird beendet, wenn die Veränderung des MSE (Mean Square Error) zwischen der letzten und der aktuellen Iteration unterhalb des Schwellwertes τ liegt. Der Parameter τ definiert die verlangte Präzision der Registrierung. Folgende Schritte werden bis zur Konvergenz innerhalb eines Toleranzwertes τ ausgeführt:

1. Berechnung der nächstgelegenen Punkte $Y_k = C(P_k, X)$,
2. Berechnung der Registrierung $(\vec{q}_k, d_k) = Q(P_0, Y_k)$,
3. Anwendung der Registrierung $P_k + 1 = \vec{q}_k(P_0)$,
4. Beendung der Iteration, falls $d_k - d_{k+1} < \tau$ mit $\tau > 0$.

Nach Abschluss der Berechnungen wird die Transformation auf das Ausdrucksmodell angewendet.
Da die Modelle in den Vorverarbeitungsschritten bereits vorregistriert wurden, ist das Konvergenzkriterium erfüllt und falsche Berechnungen der Transformationen sind ausgeschlossen. Die Genauigkeit wird durch die Ähnlichkeit der beiden Modelle (bzw. der Stirngeometrien) gewährleistet. Der 'best Match' wird nur auf die Stirn angewendet und liefert eine Transformation mit geringem Fehler.

4.1.3 Korrespondenzen

Das Korrespondenzmodul liefert laut Konzept ein Morphtarget, das aus einem Referenzscan und einem Ausdrucksscan gewonnen wird. Für die grundlegende Berechnung des optischen Flusses wurde das hierarchische, gradientenbasierte Verfahren gewählt (vgl. 2.1.3.2). Im Gegensatz zu nicht-hierarchischen Verfahren können relativ große Verschiebungen, die besonders im Mundbereich auftreten,

lokalisiert werden.
Das hierarchische Verfahren benötigt als Eingabe zwei zylindrische Texturbilder im RGB Format (Sgi) und die entsprechenden 3D Modelle als Inventor und als Wavefront.
Zur Generierung der Bildhierarchie wurde eine Klasse entwickelt, die verschiedene Funktionen für Laplace und Gauß Pyramiden enthält. Die Klasse verwendet Standardfunktionen für Bilder aus der Klasse joImage zum Zugriff auf Bilddaten. Die Pyramidenoperationen (vgl. 2.1.1) werden sowohl auf Bilder (Integer) als auch auf Beschleunigungsfelder (Floating Point) angewendet:

1. `Reduce`, zur Reduzierung des Eingabebildes/Feldes auf ein Viertel der originalen Größe,

2. `Expand`, zur Vergrößerung des Eingabebildes/Feldes auf vier mal die Größe des Originalbildes,

3. `Smooth`, zur Anwendung eines Gauß Filters auf das Beschleunigungsfeld.

`Reduce`, `Expand` und `Smooth` verwenden eine separierbare Gauß Maske G, um die Anzahl der Multiplikationen im Berechnungsschritt zu minimieren. In der Implementierung wurde eine 5x5 Filtermaske verwendet:

$$[G_x] = [G_y]^T = \frac{1}{16}[1 \quad 4 \quad 6 \quad 4 \quad 1].$$

Die Gauß Pyramide wird durch mehrfache Anwendung der `reduce`-Funktion konstruiert. Die Tiefe der Hierarchie wird durch den Parameter t bestimmt, der von der Größe der Eingabebilder abhängig ist. Je größer das Bild ist, desto tiefer ist die resultierende Bildpyramide.

Um die 3D Scans zu verarbeiten, wurde das gradientenbasierte Verfahren an den Algorithmus auf Laserscans angepasst (vgl. 3.2.3.3).
In der Implementierung werden Objekt- und Texturinformation in einem Datentyp als 6-dimensionaler Vektor kombiniert. Da eine Registrierung der Modelle stattgefunden hat, können die 3D Koordinaten als Quantitäten verwendet werden.

$$I(x,y) = (x(x,y), y(x,y), z(x,y), R(x,y), G(x,y), B(x,y))^T.$$

Der optische Fluss ist nun durch die Minimierung der quadratischen Funktion

$$E = \sum_{(x,y)\in R} \|v_x \frac{\delta I(x,y)}{\delta x} + v_y \frac{\delta I(x,y)}{\delta y} + \Delta I\|^2$$

definiert. Die Texturbilder des neutralen Modells I^0 und des Ausdrucksmodells I^1, sowie die entspechenden Meshes werden an die Hauptfunktion übergeben. Der hierarchische Algorithmus berechnet die Bewegung zwischen den Eingabedaten

und gibt das Velocity-Feld in zwei bildähnlichen Datenstrukturen Dx und Dy aus.

```
Berechne die Bildpyramiden G^0_0,...,G^0_t und G^1_0,...,G^1_t
der beiden Texturen I^0 und I^1
for(l = t; l >= 0; l = l-1){
     if (l == t){
        Vx_l = 0.0;
        Vy_l = 0.0;
     } else {
       Vx_l = expand(Vx_{l+1})
       Vy_l = expand(Vy_{l+1})
       Vx_l = 2 * Vx_l
       Vy_l = 2 * Vy_l
     }
     Berechne Warp W der Textur G^0_l mit Vx_l und Vy_l
     Schaetze Bewegung zwischen W, Mesh_1 und G^1_l, Mesh_2
     und berechne residuale Bewegung Vx_res und Vy_res
     Update:  Vx_l = Vx_l + Vx_res
              Vy_l = Vy_l + Vy_res
     Smoothing:  smooth(Vx_l)
                 smooth(Vy_l)
```

Abbildung 4.1: Pseudo-Code der Hauptfunktion basierend auf [3]

Auf alle Ebenen der Bildpyramide G_l^0 wird ein Warping durch die berechneten Velocities angewendet (Abb. 4.1). Das gewarpte Texturbild W und G_l^1 werden nacheinander an die Funktion übergeben, die durch das gradientenbasierte Verfahren die residuale Bewegung berechnet. Das Verfahren wird nur auf bestimmte Regionen der Texturen bzw. der Modelle angewendet, die durch eine binäre Maske definiert werden. Das gradientenbasierte Verfahren erzeugt für jedes Pixel der Eingabebilder unter der Maske den oben beschriebenen Datentyp. Da nicht für jedes Pixel auch ein Vertex zur Verfügung steht, werden exakte 3D Koordinaten auf dem Mesh berechnet. Über eine Umrechnung der Bildkoordinaten in Texturkoordinaten werden im Texture Space durch 2D Dreieckstests die Texturkoordinaten bestimmt, die das umgebende Dreieck bilden. Durch Lösen eines Gleichungssystems kann der exakte 3D Punkt berechnet und in den Datentyp I mit den RGB-Werten des Pixels integriert werden.
Damit werden für jede Pixelkoordinate (x, y) des gewarpten Texturbildes W lokale und temporäre Gradienten berechnet:

$$\delta_x I(x, y) = (I_0(x + 1, y) - I_0(x - 1, y))/2.0$$

$$\delta_y I(x,y) = (I_0(x,y+1) - I_0(x,y-1))/2.0$$
$$\Delta I(x,y) = I_0(x,y) - I_1(x,y).$$

Entsprechend [11] werden diese Eingabedaten mit Gewichtungen normiert, um 3D und 2D Information gleichmäßig in den Algorithmus zu integrieren. Je höher die Gewichtung gewählt wird, desto größer ist der Einfluss des entsprechenden Parameters auf das resultierende Velocity-Feld:

$$\|I\|^2 = w_x x^2 + w_y y^2 + w_z z^2 + w_R R^2 + w_G G^2 + w_B B^2.$$

Um die residuale Beschleunigung zu berechnen wird das lineare Gleichungssystem $Wv = \gamma$ durch Diagonalisierung gelöst (vgl. 2.1.3.2). Die Summierung erfolgt in einer 5x5 Pixelnachbarschaft.
Die berechnete Velocity zwischen I_0 und I_1 wird im letzten Schritt dazu verwendet, das neutrale Referenzmodell neu zu definieren. In [11] wurde eine Umsortierung der 3D Koordinaten durch die Vektoren vorgenommen.
Über die 3D Koordinaten des neutralen Modells werden die Texturwerte abgefragt. Die gefundenen Texturkoordinaten (u, v) werden in Pixelkoordinaten (x, y) umgerechnet. An der Position des berechneten Pixels wird ein in einer 5×5 Pixelnachbarschaft gemittelter Velocity-Vektor addiert $(x + v_x, y + v_y)$. Die Koordinaten des neuen Pixels werden wiederum in Texturkoordinaten umgerechnet und mit der oben genannten Methode der exakte 3D Punkt im Ausdrucksmodell berechnet. Der 3D Punkt im neutralen Modell wird durch diesen neuen 3D Punkt ersetzt. Durch dieses Verfahren nimmt das neutrale Modell die Form des Ausdrucksscans an. Zuletzt werden die neuen 3D- und Texturkoordinaten in einer Textdatei abgespeichert.
Wie im Konzept beschrieben, wird die Übertragung der Scanmodelle auf ein generisches durch den optischen Fluss gelöst. In der Vorverarbeitung werden die Morphtargets auf das generische Modell gelegt und die Texturen an das generische Modell angepasst. Die Integration der Daten in JOSH gleicht dem bisher beschrieben Vorgang. Der optische Fluss wird auf die Berechnung von Laplace Pyramiden umgestellt, da die Texturen der beiden Modelle farblich sehr unterschiedlich sein können.

4.2 Modul Kontrollierbarkeit

Das Modul Kontrollierbarkeit ist offline zum Update bzw. Aufbau einer Constraint-Datenbasis sowie während einer Animation zur Kontrolle der Gesichtsausdrücke zuständig.
Getestet wurde das System mit einem männlichen, generischen Kopfmodell. Zusätzlich wurde eine Muskeltextur für diesen Kopf verwendet, in der viele wichtige Muskeln des menschlichen Gesichts eingezeichnet sind. Die Muskelfenster wurden ähnlich zu [45] definiert und die entsprechenden Muskeln werden über die

Muskeltextur in den generischen Kopf integriert. Dieses Prinzip zeigt Abb. 4.2. Die Tabelle 4.1 zeigt den Zusammenhang zwischen den Muskelfenstern, den Muskeln und den entsprechenden AUs.

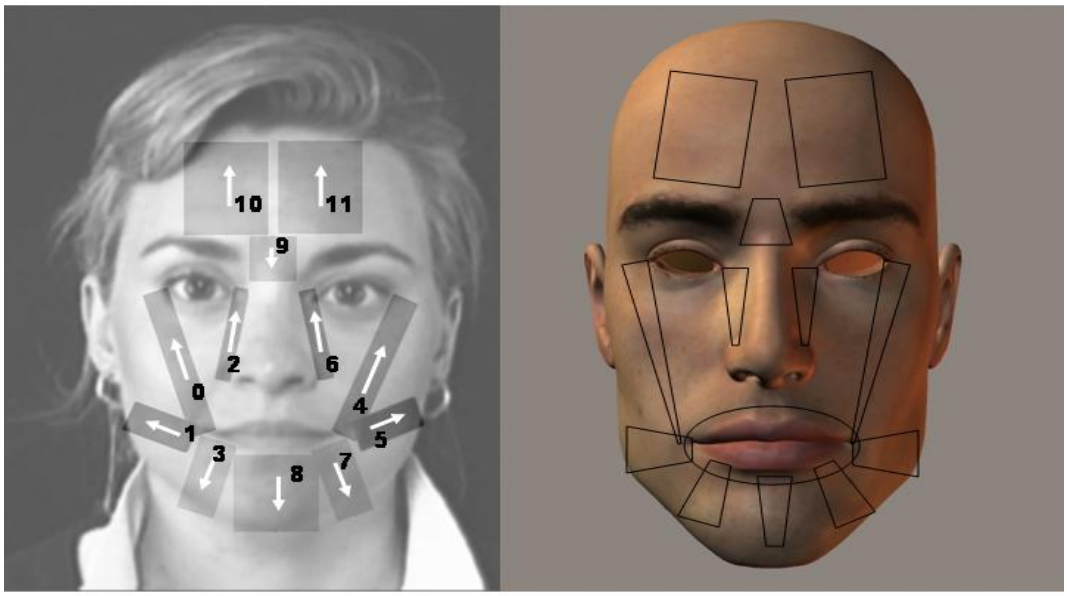

Abbildung 4.2: Muskelfenster und Abbildung auf das generische Modell

Muskelfenster	Muskeln	AU
m0 m4	M. Zygomaticus Major	cheek raiser
m1 m5	M. Zygomaticus Major M. Buccinator	lip corner puller
m2 m6	M. Levator Labii	upper lip raiser
m3 m7	M. Depressor anguli oris	lip corner depressor
m8	M. Mentalis Kieferrotation M. Depressor Labii	chin raiser jaw drops lips part
m9	M. Depressor Supercilii M. Levator Labii	nose wrinkler
m10 m11	M. Venter Frontalis M. Depressor Supercilii	inner brow raiser brow lowerer

Tabelle 4.1: Muskelfenster, Muskeln und Action Units

4.2.1 Aufbau und Update der Datenbasis

Die Klasse `ConstraintBuilder` ist für Aufbau und Update der Constraint-Datenbasis zuständig.
Durch den optischen Fluss auf Grauwertbilder und Mases Ansatz werden Muskelkontraktionen aus Bildern extrahiert. Die Datenbasis ist eine Textdatei, die durch Verwendung des `ConstraintBuilders` jederzeit aktualisiert werden kann. Sie enthält für alle analysierten Bilder eine Muskelfenster-Identifikationsnummer sowie minimale oder maximale Kontraktionswerte.
An den `ConstraintBuilder` werden vier Grauwertbilder als RGB übergeben. Das erste Bild ist ein neutraler Gesichtsausdruck, das zweite Bild zeigt dieselbe Person mit einem Gesichtsausdruck oder einer Kombination aus Gesichtsausdruck und Visem. Das dritte Bild ist eine binäre Maske, das die Regionen definiert, unter denen der optische Fluss berechnet werden soll. Das vierte Bild ist eine Art Maske, die die spezifischen Muskelfenster für die dargestellte Person definiert. Jedes Muskelfenster ist durch Parameter festgelegt, die in den RGB-Kanälen der Maske kodiert werden:

1. ID,
2. Orientierung,
3. Größe.

Im `CostraintBuilder` werden diese Informationen aus der Maske extrahiert. Der Grauwert eines Fensters liefert die entsprechende Identifikationsnummer. Die Orientierung wird in den Muskelfenstern im G-Kanal zweier Pixel definiert. Diese beiden Pixel werden in jedem Fenster gesucht und liefern zwei Pixelkoordinaten, aus denen ein normierter Vektor berechnet wird, der die Orientierung n_i des Fensters definiert. Die Größe S_i eines Fensters ist die Anzahl der überdeckten Pixel.
Der erste Schritt ist die Berechnung des optischen Flusses durch die zwei Bilder und die binäre Maske. Die daraus gewonnenen Beschleunigungsfelder V_x und V_y werden im zweiten Schritt entsprechend der in Abschnitt 3.3.2 aufgeführten Formel zur Berechnung der Kontraktionen verwendet. Die Kontraktionswerte jedes Fensters m_i werden zusammen mit ihrer ID (Abb. 4.3) anschließend in einer Textdatei (ConstraintFile) abgespeichert.

```
{
window_50 0.02
window_60 0.01
window_80 0.00
window_90 0.00
window_100 0.01
window_110 0.01
window_140 1.00
window_150 1.02
window_160 -0.59
window_170 0.012
window_180 0.018
window_190 -0.014
}
```

Abbildung 4.3: Auszug aus der ConstraintFile

4.2.2 Initialisierung

In der Initialisierungsphase werden einerseits die Constraints aus der Datenbasis auf den schnellen Zugriff für die Animation vorbereitet, andererseits wird das parametrische Muskelsystem in das Mesh des generischen Gesichtsmodells eingefügt.
Die Initialisierung der Constraints wird durch einen `ConstraintLoader` durchgeführt. Dieser parst die ConstraintFile und speichert die Parameterbelegungen der Muskelfenster in einem zweidimensionalen STL Vektor. Die gebildete $m \times n$ Matrix enthält in jeder Zeile einen Featurevektor der Länge m, entsprechend der Anzahl der Muskelfenster und Spaltenvektoren der Länge n, entsprechend der Anzahl der analysierten Bilddaten.
Die Initialisierung des Muskelsystems basiert auf der Eingabe dreier Texturen:

- Eine Muskeltextur definiert alle linearen Muskeln durch jeweiliges Einzeichnen eines Insertionspunktes, eines Knochenpunktes und weiteren Muskelpunkten (lineare Segmente) dazwischen. Zirkulare Muskeln sind nur durch Insertionspunkte definiert.

- Eine Blattmuskel-Textur definiert zwei farblich gekennzeichnete Einflussregionen für die Blattmuskeln an der Stirn.

- Eine Kiefer-Textur definiert die farblich gekennzeichnete Einflussregion des Kiefers.

Auf der Muskeltextur wurden insgesamt zwölf lineare Muskeln sowie ein zirkularer Muskel um den Mund als lineare Segmente eingezeichnet. Zwei weitere Muskeln werden über die Blattmuskel-Textur in das Modell integriert.

Aus den Texturen werden initiale Parameterbelegungen in die Datentypen integriert. Die Parameter sind von den Eigenschaften des Datentyps abhängig. Aus der Muskeltextur werden alle Daten für lineare und zirkulare Muskeln extrahiert.

4.2.2.1 Lineare Muskeln

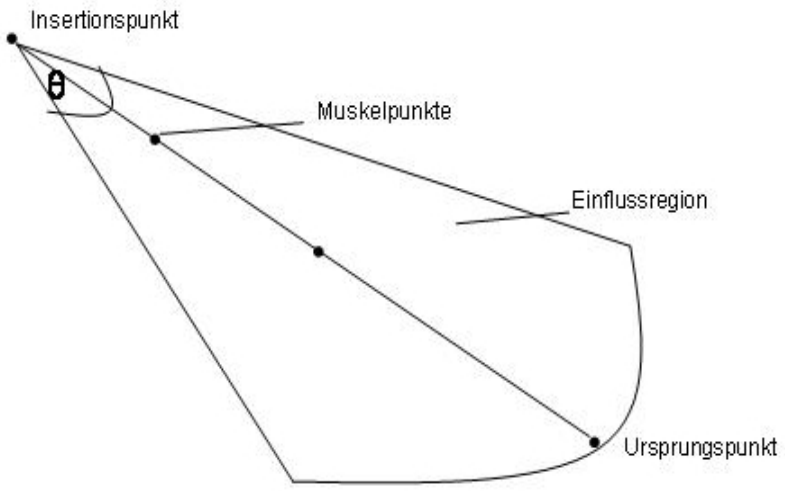

Abbildung 4.4: Umsetzung des linearen Muskels

Jeder lineare Muskel definiert folgende Parameter:

- Identifikationsnummer,
- Insertionspunkt und zusätzliche Muskelpunkte,
- Knochenpunkt,
- Einflussregion,
- Minimaler, maximaler und aktueller Kontraktionswert.

Der Insertionspunkt und alle Muskelpunkte werden durch nächstgelegene Texturkoordinaten und den daraus gewonnenen Vertizes des generischen Modells bestimmt. Lineare Muskeln sind entweder mit dem unbeweglichen Schädel oder mit dem rotierbaren Kiefer verbunden (Anhang B). Der Schädel- oder Kieferpunkt wird exakt durch Lösen eines linearen Gleichungssystems berechnet. Durch den Insertionspunkt und den Ursprungspunkt wird ein Muskelvektor definiert. Die Identifikationsnummer wird aus dem Grauwert der Verbindungen zwischen den Muskelpunkten extrahiert und gleicht einer entsprechenden ID der Muskelfenster. Die Einflussregion ist ein Kreissektor mit dem Muskelvektor als Radius und einem vordefinierten Winkel. Alle 3D Punkte der Einflussregion eines spezifischen

linearen Muskels werden in einem STL Vektor abgespeichert. Gleichzeitig wird für jeden dieser Punkte, einschließlich der Muskel- und Insertionspunkte, der Index abgespeichert, an dem sich der Punkt im Koordinatenarray des Meshes befindet. Die Grenzwerte (Min/Max) definieren ein Intervall, in dem sich der aktuelle Kontraktionswert bewegen darf. Diese Werte werden in der Initialisierungsphase nicht gesetzt, sondern verändern sich erst zur Laufzeit automatisch.

4.2.2.2 Zirkulare Muskeln

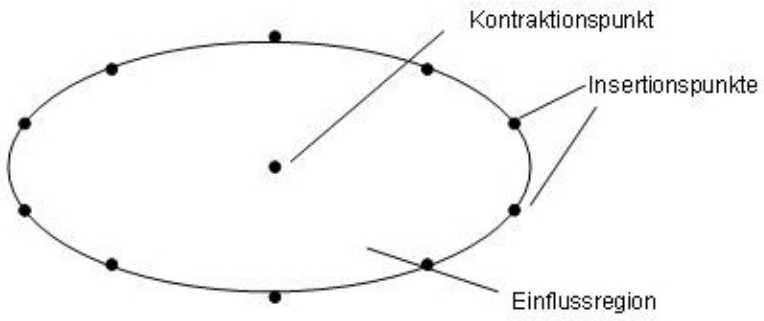

Abbildung 4.5: Korrespondenzen zwischen zwei Laserscans

Jeder zirkulare Muskel definiert folgende Parameter:

- Insertionspunkte,
- Einflussregion,
- Kontraktionspunkt,
- Minimaler, maximaler und aktueller Kontraktionswert.

Die zirkularen Muskeln unterscheiden sich von linearen Muskeln dadurch, dass sich alle Muskelpunkte im Mesh befinden und kreisförmig angeordnet sind. Alle Punkte (bzw. Indizes), die sich innerhalb der linearen Segmente befinden, werden in der zirkularen Einflussregion abgespeichert. Diese Punkte werden durch Inside-Outside Tests mit Hilfe der linearen Segmente bestimmt. Zirkulare Muskeln kontrahieren in Richtung eines bestimmten Punktes innerhalb seiner Einflussregion. Daher wird ein initialer Kontraktionspunkt durch den Schwerpunkt der Muskelpunkte berechnet.

4.2.2.3 Blattmuskeln und Kiefer

Blattmuskeln verwenden den gleichen Parametersatz wie lineare Muskeln. Die Einflussregion eines Blattmuskels wird durch die Blattmuskel-Textur definiert

und überdeckt statt eines Kreissektors großflächige Regionen der Stirn.
Der Kiefer wird durch die Kiefer-Textur definiert. Er besitzt einen Kontrollmechanismus, der die Länge zwischen dem Kontraktionspunkt und dem bezüglich der y-Achse niedrigsten Muskelpunkt des zirkularen Muskels um den Mund kontrolliert.

In der Initialisierungsphase besteht keine Verbindung zwischen den Constraints und den parametrischen Muskeln. Die wirkliche Zuordnung realer Daten und Muskeln findet erst während der Animation statt. Die Muskeln befinden sich bisher in einem initialen Zustand und es besteht eine geometrische Abhängigkeit durch ihre Integration in das generische Gesichtsmodell.

4.2.3 Animation

Nach der Initialisierungsphase wird das Constraintsystem mit einer gleichzeitigen Animation des generischen Modells gestartet. Der Kontrollmechanismus wird pro Frame aufgerufen und verläuft sequentiell in folgender Reihenfolge:

1. Update der linearen Muskeln,
2. Update der zirkularen Muskeln,
3. Update des Kiefers,
4. Aktualisierung der Min/Max Kontraktionswerte,
5. Control,
6. Animation,

4.2.3.1 Update

Die Update Funktionen passen die Parameter der Muskeln und des Kiefers an die bereits gemorphte Geometrie an. Das Update bestimmt die neuen Positionen der Muskelpunkte, des Insertionspunktes und der Einflussregion durch die in der Initialisierung gespeicherten Indizes. In einem weiteren Vektor wird der alte Zustand der Einflussregion gespeichert. Zusätzlich muss der alte Kontraktionswert gespeichert werden.
Auf Basis dieser Anpassungen kann der neue Kontraktionswert c des Muskels berechnet werden. Durch die Muskelpunkte m_i inklusive Insertionspunkt und den initialen Muskelpunkten m_start_i werden Vektoren berechnet, die die Verschiebung der gemorphten Punkte definieren. Ähnlich zu Mases Ansatz wird das Skalarprodukt zwischen jedem Vektor und der Muskelrichtung d berechnet:

$$c_{linear} = \frac{1}{S} \sum < (\vec{m}_i - \vec{m}\_start_i), \vec{d} >,$$

wobei S die Anzahl der Muskelpunkte ist.
Bei zirkularen Muskeln wird der Kontraktionspunkt durch den Schwerpunkt der Muskelpunkte aktualisiert. Der Kontraktionswert eines zirkularen Muskels wird durch die durchschnittliche Länge $mLength_i$ zwischen Muskelpunkten m_i und Kontraktionspunkt cp sowie die entsprechenden Längen der Startkonfiguration des Muskels $mOldLength_i$ berechnet.

$$c_{circular} = 1 - (\frac{1}{S} \sum \frac{mLength_i}{mOldLength_i}).$$

Zusätzlich findet eine Skalierung der Kontraktionswerte c_{linear} und $c_{circlar}$ statt.
Eine ähnliche Berechnung findet im Update des Kiefers statt. Das Verhältnis der Länge zwischen dem Muskelpunkt mit dem minimalsten y-Wert des zirkularen Muskels, des Kontraktionspunkts und des Startkontrollwertes liefert den aktuellen Kontrollwert für den Kiefer. Die Berechnungen für Blattmuskeln gleicht den Berechnungen für lineare Muskeln.

4.2.3.2 Aktualisierung der Min/Max-Werte

Nach dem Update-Prozess erfolgt eine Aktualisierung des minmalen bzw. maximalen Kontraktionswerts, um die erlaubten Grenzen auf den nächsten Morphing-Schritt vorzubereiten. Entsprechend der Anordnung der IDs in der Featurematrix wird ein aktueller Featurevektor aufgebaut, der aus allen aktuellen Kontraktionswerten der linearen Muskeln besteht. Ein Lookup in der Datenbasis berechnet den nächstgelegenen Vektor aus der Featurematrix. Abb. 4.6 verdeutlicht die Berechnung in einem vereinfachten 2D Fall.

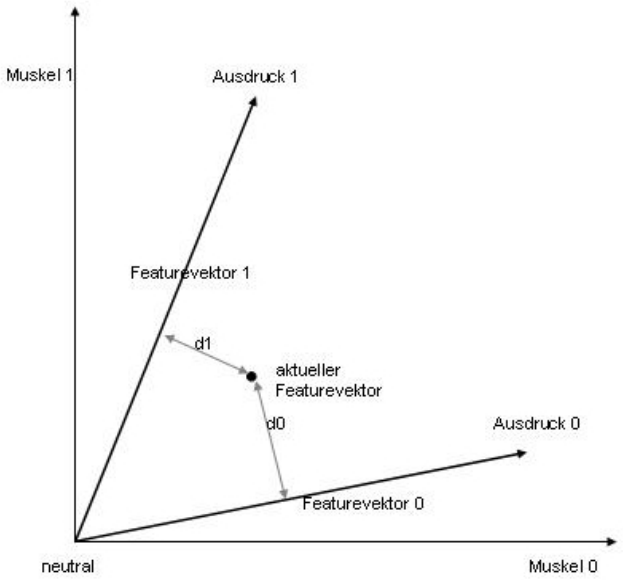

Abbildung 4.6: Featurevektoren und aktuelle Parameterbelegung

Jeder Featurevektor der Datenbasis ist eine spezielle Parameterbelegung für einen Gesichtsausdruck. Da sich der aktuelle Featurevektor annähernd neben oder auf der Strecke vom Ursprungs- (=neutral) bis zum Ausdruckspunkt eines Featurevektors befinden sollte, wird die Distanz zu einem Punkt auf der Strecke berechnet. Der Punkt auf der Strecke liegt auf einer zum Featurevektor senkrecht verlaufenden Linie durch den aktuellen Featurevektor. Die verwendete Distanzberechnung kann auf n-dimensionale Fälle verallgemeinert werden. Aus den Distanzen d_0 und d_1 wird die kleinste ausgewählt und der entprechende Featurevektor der Datenbasis wird an das Constraintsystem übergeben. Dort werden die minimalen und maximalen Kontraktionswerte der linearen Muskeln durch den Vektor aktualisiert. Der Kontraktionswert des Muskelfensters des M. Mentalis am Kinn (Abb. 4.2) liefert Informationen über eine vorhandene Kieferrotation. Falls der Kontraktionswert dieses Muskels negativ ist, wird der maximale Kontrollwert des Kiefers mit diesem Wert aktualisiert. Ist ein positiver Wert gegeben, wird stattdessen der maximale Kontraktionswert des Mentalis gesetzt.

4.2.3.3 Control

Der nächste Schritt ist eine Kontrollfunktion, die überprüft, ob sich die aktuelle Parameterbelegung aller Muskeln innerhalb der definierten Grenzen befindet. Falls der Kontraktionswert eines Muskels größer als der maximale bzw. kleiner als der minimale Wert ist, wechselt der Muskel in einen kontrollierten State. Dieser State verhindert, dass sich der Muskel und seine Einflussregion in der Animation weiterbewegt: Die zuletzt erlaubte Einflussregion (mit erlaubtem Kontraktionswert) wird mit Hilfe der abgespeicherten Indizes in das Koordinatenarray des generischen Modells eingefügt.
Da durch diesen Vorgang auch alle Muskelpunkte eingeschränkt werden, wird ein temporärer Kontraktionswert durch den Insertionspunkt und die Muskelpunkte berechnet. Der Insertionspunkt darf sich während des kontrollierten States weiterbewegen und liefert weitere Kontraktionsionswerte $c > MAX$ bzw. $c < MIN$. Erst wenn der temporäre Kontraktionswert gleich dem zuletzt erlaubten Kontraktionswert ist, wird der Kontrollstate aufgelöst, und der Muskel wird für die Animation freigegeben. Die Kontrollfunktion wird auf lineare und zirkulare Muskeln sowie Blattmuskeln und Kiefer, die durch ihre unterschiedlichen Einflussregionen verschiedene Auswirkungen auf das generische Modell haben, ausgeführt.

4.2.3.4 Definition von Regeln

Abhängigkeiten zwischen den verschiedenen Aktuatoren sowie zwischen Aktuatoren und Kiefer verlangen eine Definition von Regeln. Die wichtigsten Muskeln für Viseme und viele Gesichtsausdrücke sind Buccinator, Zygomatic und Orbicularis Oris (Anhang B). Die Muskelfasern dieser Muskeln greifen in der Mundregion ineinander und formen die intrinsische Lippenstruktur. Die Muskeln sind dadurch

verflochten und mischen sich zu gekoppelten Muskelaktionen [22]. Das Prinzip der verflochtenen Muskeln wurde in der Mundregion umgesetzt. Alle Muskeln, deren Insertionspunkt in der Einflussregion des Mundes liegt, bilden eine Constraintgruppe. Ist eine maximale bzw. minimale Muskelkonfiguration eines dieser Muskeln erreicht, wird der Constraint an die gesamte Constraintgruppe weitergeleitet. Bei z.B. einem maximalen Lächeln sind alle Muskeln der Constraintgruppe maximal kontrahiert. Eine weitere Bewegung eines Muskels und damit einer Region auf dem Mesh würde einen unrealistischen Gesichtsausdruck erzeugen. Löst ein Muskel der Gruppe einen Constraint aus, wird der aktuelle Kontraktionswert jedes Muskels dieser Gruppe kurzzeitig zu einem Maximalwert.
Eine weitere Regel ist für die Abhängigkeit zwischen linearen Muskeln sowie der Mundregion und der Rotation des Kiefers definiert. Eine maximale Rotation des Kiefers resultiert in einem gleichzeitigen Constraint der Mundregion.
Einige lineare Muskeln sind mit dem beweglichen Kiefer durch den Knochenpunkt verbunden. Diese Muskeln befinden sich in der Einflussregion des Kiefers und müssen gegebenenfalls an eine Kieferrotation durch eine Neuberechnung des Knochenpunktes angepasst werden.

Zusammenfassend ist das Resultat der realisierten Konzepte und Ideen eine Integration personalisierbarer Daten verschiedener Art in ein Animationssystem. Das Modul Personalisierbarkeit legt dabei den Grundbaustein für die Animation. Aufgrund der verwendeten Animationstechnik und ihrer Additivität liefert das Muskel-basierte Constraintsystem eine geometrische Kontrolle, die durch eine analytische Betrachtung personalisierbarer Daten unterstützt werden kann. Aus Erweiterungen des Systems durch Regeln, die durch anatomische Beziehungen im Parameterraum definiert werden können, lassen sich geometrische Beziehungen herstellen.

Kapitel 5

Ergebnisse

In diesem Kapitel werden die Ergebnisse der Arbeit vorgestellt. Der erste Abschnitt beschreibt die Resultate des Registrierungsverfahrens. Im zweiten und dritten Abschnitt wird die Erstellung der Morphtargets anhand weiblicher Scans und die Übertragung anhand eines männlichen Scans gezeigt. Der letzte Abschnitt beschreibt die Ergebnisse des Constraintsystems anhand einer beispielhaften Animation.

5.1 Alignment

Der Registrierungsprozess findet innerhalb des JOSH Systems statt und legt zwei Gesichtsmodelle übereinander. In den Tests wurden statische Regionen des Ausdrucksscans mit dem Referenzmodell registriert. Das neutrale Referenzmodell dient als die Modellform mit ca 13200 Vertizes, die statische Stirngeometrie besteht aus ca. 4000 Vertizes und wird als Datenform für den Algorithmus verwendet. Das Verfahren konvergiert zur korrekten Lösung innerhalb von 30 Iterationen mit einem RMS (Root Mean Square Error) von 0.49 mm. Dieser Wert beschreibt die Diskrepanz zwischen den beiden Modellen und ist ein Maß für die Exaktheit des Verfahrens. Obwohl der ICP Algorithmus nicht für Datenpunkte ohne Korrespondenzen in der Modellform designed wurde, kann man daraus folgern, dass eine geringe Menge falscher nächstgelegener Punktpaare nur einen geringen Effekt auf die endgültige Registrierung hat. Abb 5.1 zeigt die vorregistrierten Modelle (neutal/Mund geöffnet) und das Ergebnis der Registrierung. Abb 5.2 zeigt die Stirnregion vor und nach der Registrierung.

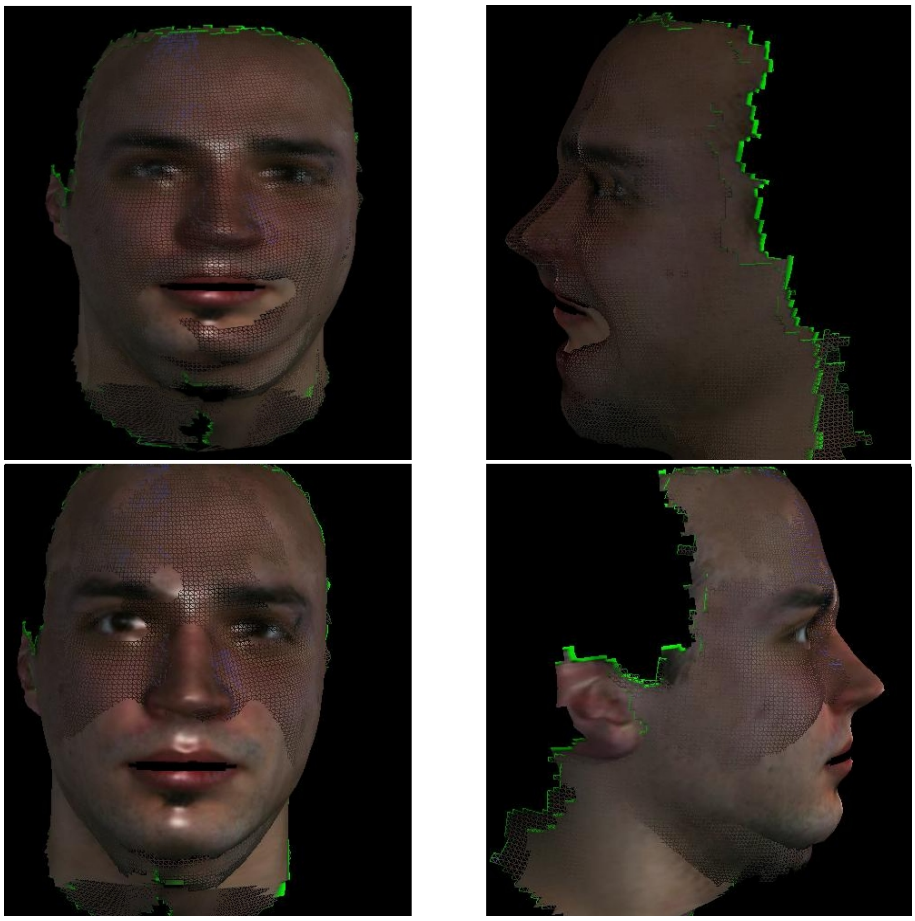

Abbildung 5.1: Scans vor (oben) und nach der Registrierung (unten)

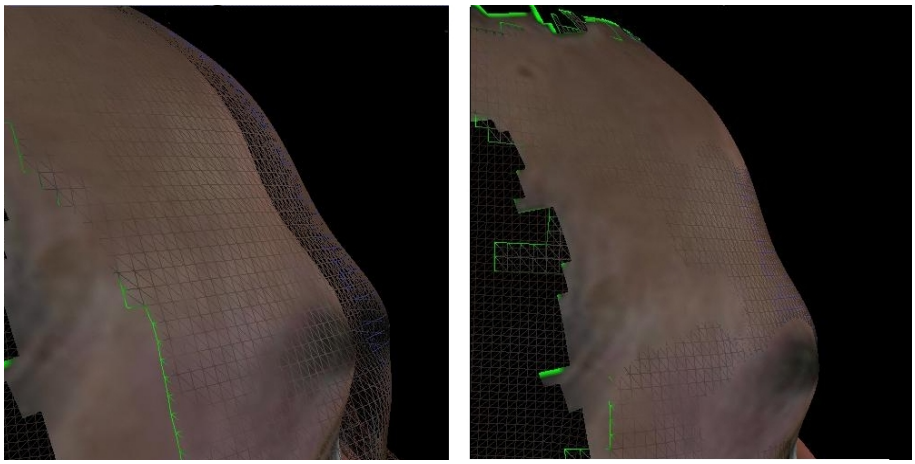

Abbildung 5.2: Stirnregion vor (links) und nach der Registrierung (rechts)

5.2 Scanbasiertes Morphing Modell

Die Ergebnisse der Erstellung personalisierbarer Morphtargets werden im Folgenden anhand eines Beispiels dargestellt. Entsprechend [13] wird eine intermediäre Mundkonfiguration als Referenz verwendet. Das Referenzmodell ist eine komplette Rekonstruktion eines weiblichen Kopfes. Als Ausdrucksscan wird ein Gesichtsmodell derselben Person verwendet, das nur aus 3-4 Bildern rekonstruiert wurde. Der Ausdrucksscan zeigt ein Lächeln, das durch die Berechnungen des optischen Flusses in das Referenzmodell integriert wird. Abb. 5.3 und Abb. 5.4 zeigen den verwendeten Datensatz bestehend aus den Modellen und den entsprechenden Texturen. In diesem Fall fiel die Wahl auf eine Gauß Pyramide, da die Farbinformationen der Texturen sehr ähnlich sind.

Abbildung 5.3: Referenzmodell und Ausdrucksscan

Abbildung 5.4: Zylindrische Texturen des Refernzmodells und des Ausdrucksscans

Die daraus berechneten Beschleunigungsfelder können in einem Nadeldiagramm visualisiert werden. Auf jeder Ebene der Bildpyramide wird durch die Textur des Referenzmodells und den berechneten Velocities ein Warpbild erzeugt. Abb. 5.5 zeigt das Nadeldiagramm und das berechnete Warpbild der untersten Ebene der Bildpyramide.

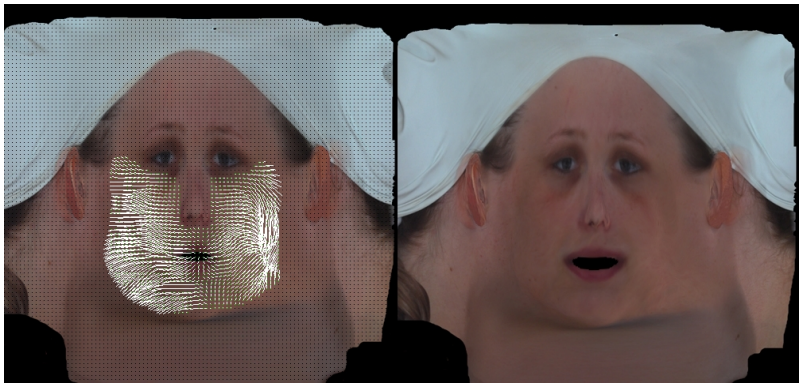

Abbildung 5.5: Nadeldiagramm und Warp

Im letzten Schritt erzeugt das Korrespondenzmodul das neu definierte Referenzmodell. Abb. 5.6 zeigt das entstandene Morphtarget im Vergleich zum eingegeben Ausdrucksscan sowie das texturierte Morphtarget mit der Textur des Referenzmodells.

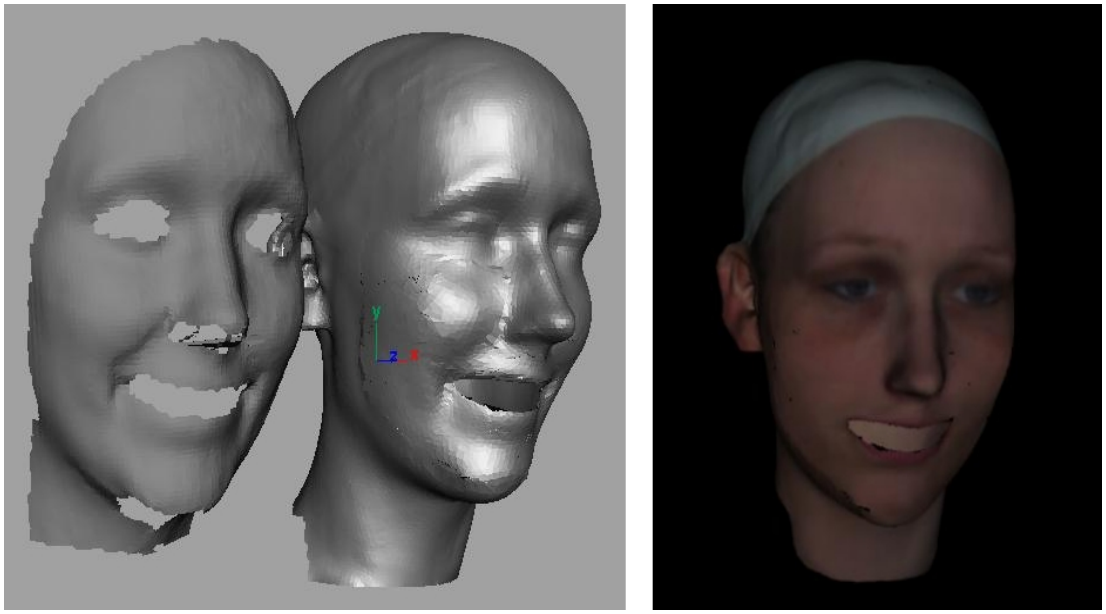

Abbildung 5.6: Ausdrucksscan, Morphtarget und texturiertes Morphtarget

5.3 Übertragung eines männlichen Gesichsscans

Die Übertragung wird anhand eines anderen Beispiels gezeigt. Hierfür wurde ein männlicher Gesichtsscan mit neutralem Gesichtsausdruck auf das generische Modell übertragen. Da in diesem Fall die Farbinformation zwischen den Texturen sehr unterschiedlich ist, wurde eine Laplace Pyramide für die Berechnungen der Korrespondenzen verwendet. Die Eingabedaten sind in Abb. 5.7 und 5.8 zu sehen.

Abbildung 5.7: Generisches Modell und neutrales Scanmodell

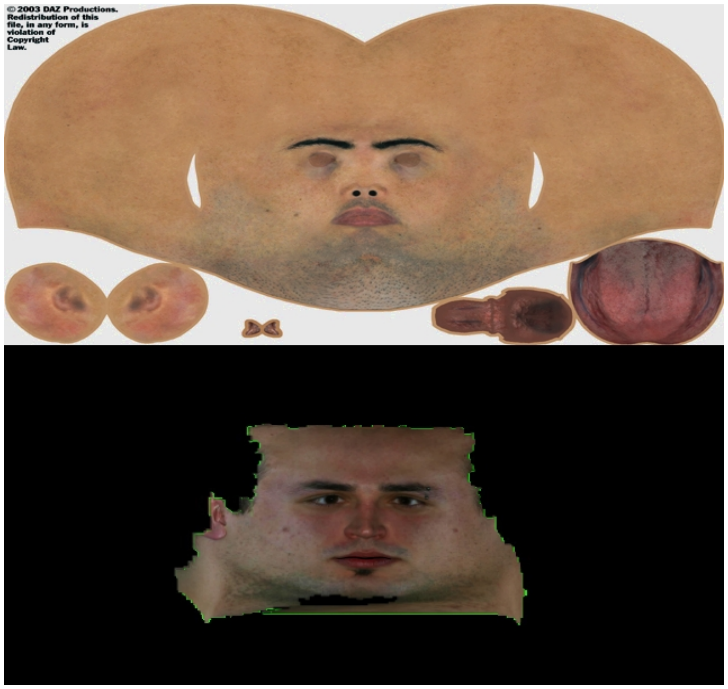

Abbildung 5.8: Zylindrische Texturen der Modelle

Das Scanmodell wurde während der Vorverarbeitung in Maya auf die Größe des

generischen Modells skaliert. Zudem wurden die Kontraste der Texturen erhöht. Abb. 5.9 zeigt das resultierende Morphtarget.

Abbildung 5.9: Alle Punkte des generischen Modells liegen im neutralen Scanmodell (links). Resultat der Übertragung (rechts).

5.4 Constraints

Das Constraintsystem wurde mit drei Featurevektoren initialisiert. Der optische Fluss für Grauwertbilder wurde auf drei Bildpaare angewendet. Die daraus berechneten Nadeldiagramme und die entsprechenden Muskelfenster sind in Abb. 5.10 zu sehen. Das erstellte ConstraintFile besteht aus drei verschiedenen Parameterbelegungen für Muskelkontraktionen, die während der Animation Constraints auf

- Lächeln,
- Heben der Augenbrauen,
- Überrascht

legen.

Abbildung 5.10: Vorberechnete Constraints

Um das Constraintsystem zu testen, wurde eine Animation erstellt, die einzelne Morphtargets und Kombinationen auf dem generischen Kopf abspielt. Im Folgenden wird das Constraintsystem anhand von zwei Beispielen verdeutlicht:

1. Smile,
2. Visem A + Visem O.

Die Gewichtungen der Morphtargets wurden so gewählt, dass unrealistische Gesichtsausdrücke in der originalen Animation entstehen. Nach der Initialisierung des Constraintsystems werden kontinuierlich Distanzen zwischen den aktuellen Parametern des Muskelsystems und den Featurevektoren der Datenbasis berechnet. Die folgenden Diagramme verdeutlichen den Start des wirkenden Featurevektors und zeigen, welche spezifischen Muskeln die verschiedenen Morphtargets beeinflussen. Die Diagramme beschreiben die aktuelle Parameterbelegung (schwarz)

und den maximalen Zustand des Muskels oder des Kiefers (weiß). Die Animation wurde jeweils mit Constraints (im Bild links) und ohne Constraints (Mitte) abgespielt. Der berechnete Featurevektor ist jeweils rechts zu sehen.

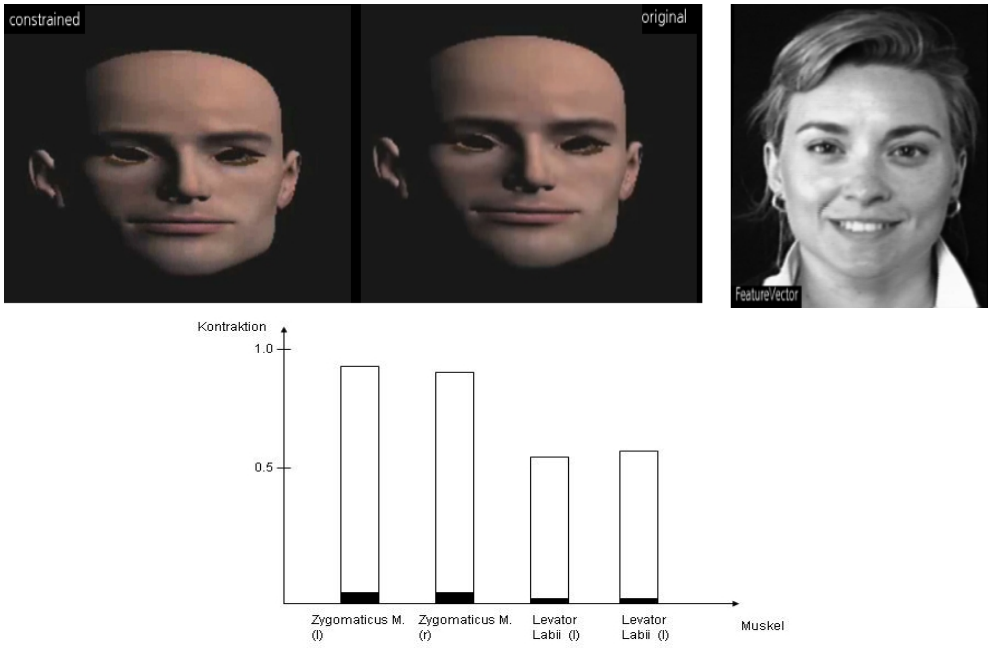

Abbildung 5.11: Featurevektor (Lächeln) ist gesetzt

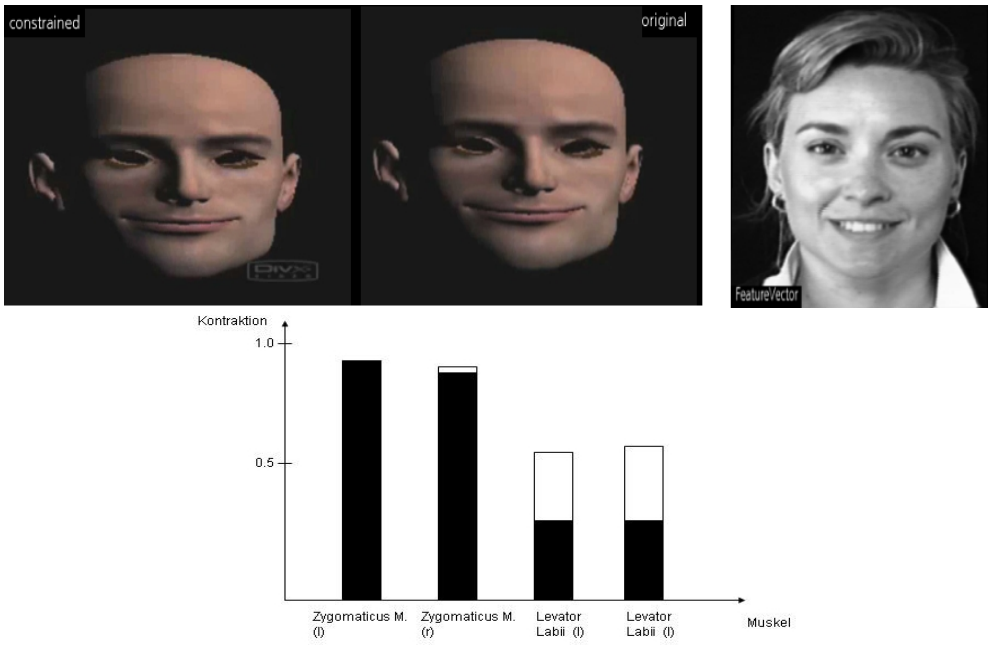

Abbildung 5.12: Maximaler Kontraktionswert ist erreicht

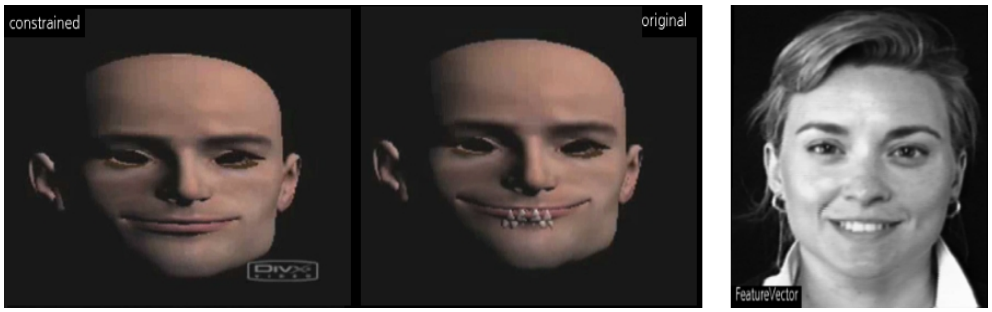

Abbildung 5.13: Constraints wirken

Das Morphtarget 'Lächeln' wirkt sich besonders auf den M. Zygomaticus Major und auf den M. Levator Labii aus. Der M. Zygomaticus erreicht einen maximalen Zustand und löst den Constraint der Constraintgruppe aus.

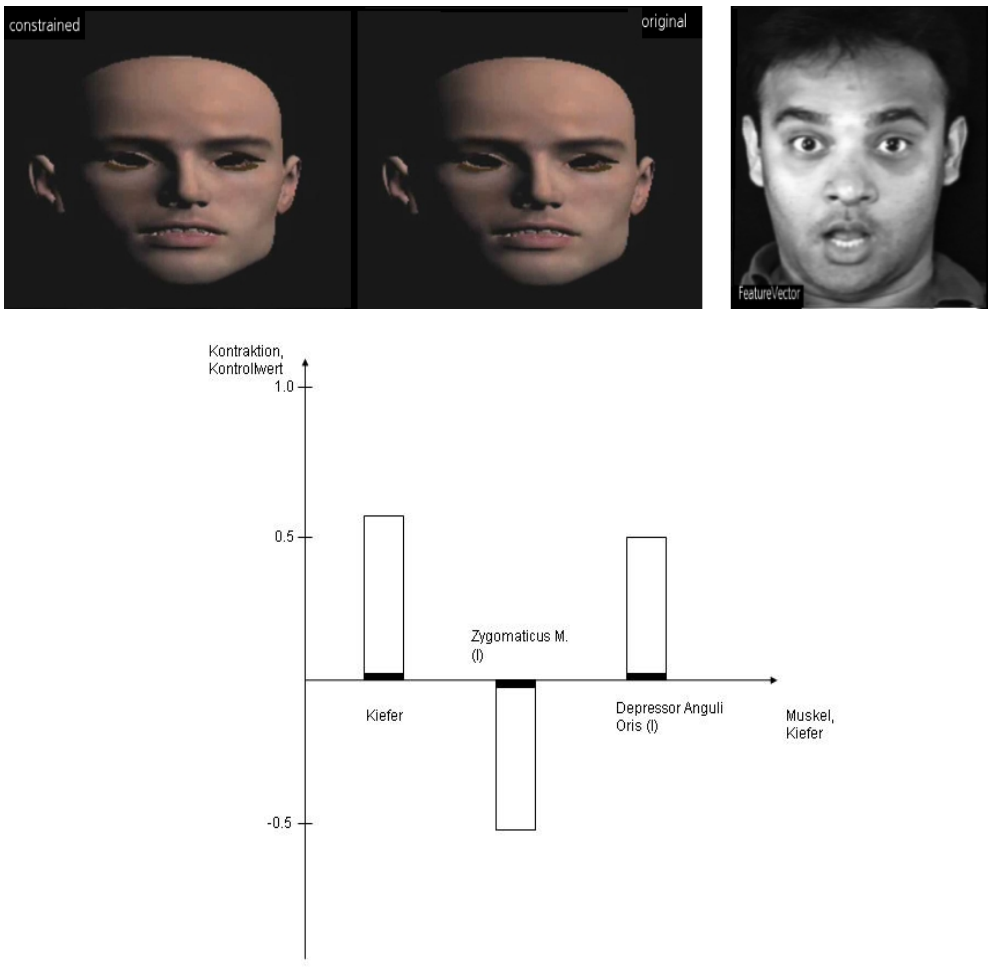

Abbildung 5.14: Featurevektor (Überrascht) ist gesetzt

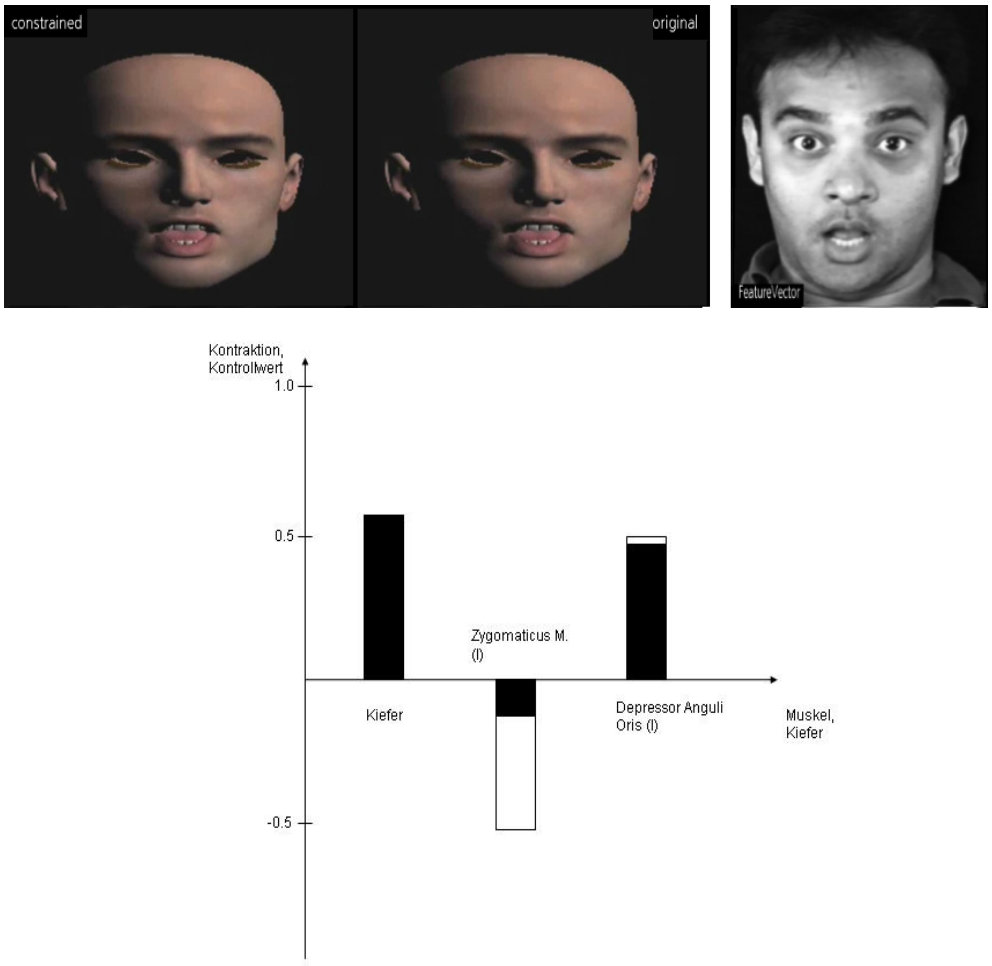

Abbildung 5.15: Maximaler Kontrollwert ist erreicht

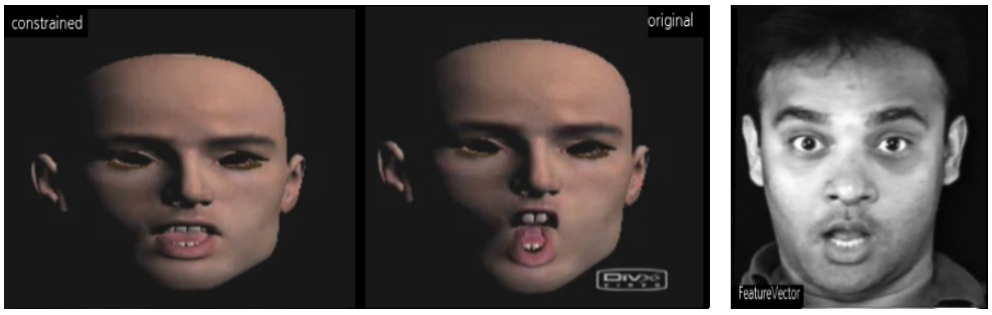

Abbildung 5.16: Constraints wirken

Die Viseme A und O erzeugen eine Kontraktionen des M. Depressor Anguli Oris. Der Kontraktionswert des M. Zygomaticus wird negativ, da sich der Muskelvektor

durch Kontraktion des Orbicularis Oris verlängern muss. In diesem Fall erzeugt der Kiefer einen Constraint.

Kapitel 6

Zusammenfassung und Ausblick

6.1 Zusammenfassung

Innerhalb dieser Arbeit konnte ein System zur Erstellung personalisierbarer Morphtargets aus 3D Scans, deren Übertragung auf generische Modelle und deren Integration in ein echtzeitfähiges Constraintsystem vorgestellt werden. Die Notwendigkeit für ein Constraintsystem liegt in den additiven Eigenschaften des Morphings.
Die Erstellung von Morphtargets ist ein Prozess, der in mehreren manuellen und automatisierten Teilschritten durchgeführt werden kann. Die Pipeline enthält einen Vorverarbeitungsprozess, einen Registrierungsprozess sowie die Bestimmung von Punkt-zu-Punkt Korrespondenzen zwischen den Modellen. Die einzelnen Prozesse können durch unterschiedliche Algorithmen gelöst werden, die innerhalb der Konzeptionierung diskutiert wurden. Die Evaluierung der Algorithmen ergab, dass die Registrierung mit dem Iterative Closest Point Algorithmus gelöst werden kann, da die Anwendung des Algorithmus nur auf eine statische Region des Datenmodells ausgeführt wird. Die Bestimmung von Punkt-zu-Punkt Korrespondenzen liefert der Optical Flow-Algorithmus durch eine Verallgemeinerung auf 3D Scans. Dieses Verfahren beruht auf der Verwendung eines gradientenbasierten Verfahrens und wurde sehr komplexen, geometrischen Verfahren gegenübergestellt. Die Übertragung der Scans auf ein generisches Modell konnte über dieselbe Pipeline gelöst werden.
Die Kontrolle von Animationen findet in der Literatur entweder während des Entstehungsprozesses der Animation oder durch das Animationsmodell selbst statt. Verschiedene Ansätze zur Kontrolle der Animation wurden in den theoretischen Grundlagen und in der Konzeption vorgestellt.
Das Constraintsystem basiert auf der Idee der Verbindung zweier Animationsmodelle. Das Prinzip der Morphtargets wurde um ein parametrisches Muskelmodell als Kontrollmodell erweitert. Das Constraintsystem erlaubt die Integration von Grenzen einer bestimmten oder verschiedener Personen. Dies ermöglicht die Auf-

nahme eines spezifischen Mimikrepertoires der Person(en) in die Animation, die auf die Darstellung der Features des Repertoires beschränkt wird. Personalisierbarkeit macht z.B. eine Asymmetrie in den Gesichtszügen oder ein spezifisches Lächeln aus. Das Constraintsystem *kennt* diese Ausdrücke und kontrolliert das Morphing auf Basis dieser Daten.
Während der Animation wird das Kontrollmodell mit realistischen Kontraktionsgrenzen aktualisiert, die aus einem FACS basierten Optical Flow Verfahren auf Grauwertbilder vorberechnet wurden. Bei einer gegebenen Über- bzw. Unterschreitung des maximalen bzw. minimalen Kontraktionswertes wird die entsprechende Einflussregion des Muskels in der Geometrie kurzzeitig angehalten. Durch die Definition von Regeln konnten Abhängigkeiten zwischen den Muskeln sowie zwischen Muskeln und Kiefer definiert werden, die der Anatomie des menschlichen Gesichts entsprechen.
Im letzten Kapitel wurden die Ergebnisse der verschiedenen Verfahren anhand von Beispielen gezeigt.

6.2 Fazit

Die in dieser Arbeit gewonnenen Erkenntnisse zeigen, dass durch Verwendung von personalisierbarer Information realistische Animationen erstellt werden können. Zudem können zur Laufzeit vorberechnete Constraints auf das Gesichtsmodell wirken, um die Gesichtsausdrücke und Viseme in dynamischen Grenzen zu halten. Weiterführend erlaubt das Constraintsystem die Überlagerung von Animationen aus verschiedenen Quellen, wodurch zum Beispiel Sprachanimationen mit Trackingdaten kombiniert werden können, ohne dass eine Anpassung der Animationsdaten nötig ist.
Der optische Fluss ist ein interessanter Algorithmus, der vielseitig eingesetzt werden kann. In dieser Arbeit liefert er die Informationen für Punkt-zu-Punkt Korrespondenzen durch eine Verallgemeinerung auf 3D Scans und repräsentiert das analytische Verfahren zur Berechnung von Muskelkontraktionen im Gesicht durch Grauwertbilder.
Das Korrespondenzmodul liefert gute Ergebnisse bei Texturen mit ähnlicher Farbinformation. Die Übertragung der Modelle ist schwieriger, da sich die Modelle in Textur und Geometrie stark unterscheiden und eine Anpassung der Gewichtungsparameter und der Bildinformation nötig ist.
Für die Korrespondenzen sind hohe Berechnungszeiten in Kauf zu nehmen, was auf die hochaufgelösten Modelle und Texturen zurückzuführen ist. Für jeden Pixel des Texturbildes müssen verschiedene Ableitungen berechnet werden, die auf den 3D Koordinaten des Modells basieren. Dies erschwert eine Durchführung von Tests immens.
Die Wahrscheinlichkeit der Entstehung von Ausreißern ist besonders in Regionen niedriger Varianz sehr hoch, dies wirkt sich negativ auf die geometrische Neude-

finition des Referenzmodells aus.
Das Constraintsystem lieferte innerhalb der Testumgebung gute Ergebnisse. Schwierigkeiten können sich durch die Smoothing Operation des 2D optischen Flusses ergeben. Durch die Faltung wird das Bewegungsfeld über das gesamte Bild geglättet. Bei einer Kopfbewegung der fotografierten Person oder einer Hintergrundbewegung zwischen den zwei Frames können falsche Kontraktionswerte entstehen.

6.3 Ausblick

Im Rahmen des Projektes Virtual Human sollen synthetische Charaktere entwickelt werden, die als persönliche Dialogpartner eingesetzt werden. Diese Charaktere sind z.B. in interaktiver Lernsoftware verwendbar und erfordern ein realistisches und personalisierbares Aussehen und Mimik. Durch die Natürlichkeit der dargestellten Person entsteht eine vertraute Verbindung zum Benutzer. Dadurch kann die Lerneffizenz erheblich gesteigert werden.
Die in dieser Arbeit entwickelten Module können durch folgende optimierende und weiterführende Verfahren ergänzt werden.

1. Der Vorverarbeitungsprozess ist durch seine manuellen Komponenten aufwendig. Ein sinnvoller Ausbau des automatisierten Prozesses wäre ein automatisches 2D Alignment. Da durch die Registrierung bereits die Vertizes übereinanderliegen, würde sich eine zylindrische Projektion über die vertikale Hauptachse der Köpfe anbieten.

2. Die Berechnungszeiten des Korrespondenzmoduls sind durch die Verwendung hochaufgelöster Modelle und Texturen sehr hoch und verlangen eine Optimierung. Die Berechnung von Velocities für jeden Pixel ist nicht zwingend nötig. Während der Neudefinition des Referenzscans wurde das Beschleunigungsfeld für jeden Vertex unter einer 5×5 Maske gemittelt. Eine Möglichkeit wäre, nur einen Beschleunigungsvektor zu verwenden, der sich am Texturpunkt des Vertizes befindet. Dadurch könnte die Anzahl der zu berechnenden Velocities stark reduziert werden.

3. Die Smoothing Funktion des optischen Flusses für Punkt-zu-Punkt Korrespondenzen ist eine einfache Filterung des Velocity Feldes mit einer Gauß Maske. Da die Velocity besonders in Regionen mit niedriger Varianz Ausreißer enthalten kann, ist ein interpoliertes und geglättetes Velocity Feld vielversprechend. Die Berechnung der Morphtargets basiert auf den Velocity Vektoren und kann in Regionen mit starken Ausreißern Fehler in der Geometrie erzeugen. Die Energieminimierungsmethode aus [11] würde bessere Ergebnisse erzielen und befindet sich derzeit in der Entwicklung.

4. Das Resultat der Übertragung eines Scans auf das generische Modell ist von der Skalierung des Scans auf das generische Modell abhängig. Das Scanmodell muss in der Vorverarbeitungsphase manuell an das generische Modell angepasst werden. Da die Gesichtszüge des Scans z.B. durch das in Abschnitt 3.2.3.1 vorgestellte Volume Morphing verloren gehen würden, könnte man den Vorgang durch eine affine Registrierung automatisieren.

5. Das Verhalten des Constraintsystems ist von der erzeugten Datenbasis abhängig. Die Muskeln, die sich nicht in einer Constraintgruppe befinden, sollten kombinierbar sein. Ein Feature Vektor mit einer Muskelkonfiguration mit maximalem Lächeln sollte z.B. mit dem Featurevektor eines maximalen Stirnrunzelns kombiniert werden können.

6. Die Constraintbehandlung kann durch Constraints auf die Muskelrichtung erweitert werden. Nebeneinander liegende, sich überschneidende Einflussregionen enthalten eine bestimmte Menge gleicher Punkte auf dem Mesh. Die Kontraktion eines Muskels wird für jeden Muskelpunkt eines linearen Muskels durch die Muskelrichtung berechnet. Dieser Vorgang sollte zusätzlich auf die Schnittmenge der linearen Muskelregionen angewendet werden, um die Richtung zu bestimmen, in die sich der Punkt bewegt. Der berechnete Wert kann darüber entscheiden, zu welcher Einflussregion der Punkt während eines Constraints zugeordnet werden soll, oder ob überhaupt ein Constraint für diesen Punkt nötig ist.

7. Zusätzliche Muskeln um die Augen und eine Erweiterung der Muskelfenster zur Berechnung von Kontraktionen zirkularer Muskeln würden das Constraintsystem sinnvoll erweitern. Die Berechnung zirkularer Kontraktionswerte kann nicht durch Muskelfenster realisiert werden. Zirkulare Muskeln kontrahieren in Richtung eines Kontraktionspunktes, der sich innerhalb der Einflussregion befindet. Eine Möglichkeit wäre die Beobachtung zirkular angeordneter Pixel und deren Bewegung im Grauwertbild, die eine zirkulare Region einschließen.

Anhang A

Erstellung von 3D Scans

Die 3D Scans werden mit einem bildbasierten Scanverfahren mit der ShapeCam von Eyetronics produziert. Die ShapeCam besteht aus einer Digitalkamera und einem speziellen Gitterblitz, die an einem Metallrahmen angebracht sind. Die Apparatur wird auf einem Stativ befestigt. Das zu scannende Objekt wird mit einer bestimmten Distanz vor der Kamera auf einem drehbaren Stuhl ausgerichtet. Ein im Rahmen integrierter Steuermechanismus synchronisiert die Digitalkamera mit dem Gitterblitz. Bei Betätigen des Auslösers wird zeitversetzt ein normales Bild und ein Gitterbild aufgenommen. Bevor das eigentliche Modell fotografiert wird, muss eine Standard Kalibrierungstechnik angewendet werden, bei der zunächst ein Gitterfoto eines Referenzobjekts aufgenommen wird (Abb. A.1). Um die Distanz zwischen Kamera und Objekt zu bestimmen, werden zwei

Abbildung A.1: Kalibrierung des Referenzobjekts

Laserpointer, die in beiden Seiten des Metallrahmens eingelassen sind, zu einem Punkt auf dem Objekt ausgerichtet. Wurde das Kalibrierungsbild aufgenommen, kann mit dem Scannen des Modells begonnen werden. Pro Morphtarget werden jeweils 3-10 Gitter- bzw. Texturbilder aufgenommen. Mit der mitgelieferten

Abbildung A.2: Scansession

Software (ShapeSnatcher, ShapeMatcher) werden die bekannten 3D-Punkte im Kalibrierungsbild gesucht. Daraus werden die internen Parameter der Kamera durch Bestimmung von 3D-2D Punktkorrespondenzen berechnet. Für die Rekonstruktion der Geometrien wird im ShapeSnatcher jedes Gitterbild analysiert. Aus den verzerrten Gitterstrukturen und der Kalibrierungsinformation wird für jedes Bild ein 3D-Patch rekonstruiert, das mit den Texturbildern entsprechend texturiert werden kann. Da die erkannten Gitterstrukturen oft Fehler aufweisen (an Ohren, Haaransatz, etc) und dadurch falsche Geometrie ensteht, müssen Teile des Gitters per Hand gesetzt werden. Enstehende Kanten werden aus den Patches entfernt.
Der ShapeMatcher fügt die Patches in ein Modell zusammen. Durch Blenden der verschiedenen Patches entsteht ein glatter Übergang zwischen den Einzelgeometrien. Durch einen weiteren Intergrationsvorgang wird das komplette Kopfmodell berechnet. Zur Erstellung einer zylindrischen Textur wird ein Zylinderobjekt um das Modell gelegt, auf das die geblendeten Texturen der Patches projiziert werden. Textur und Objekt können in verschiedenen Formaten exportiert werden.

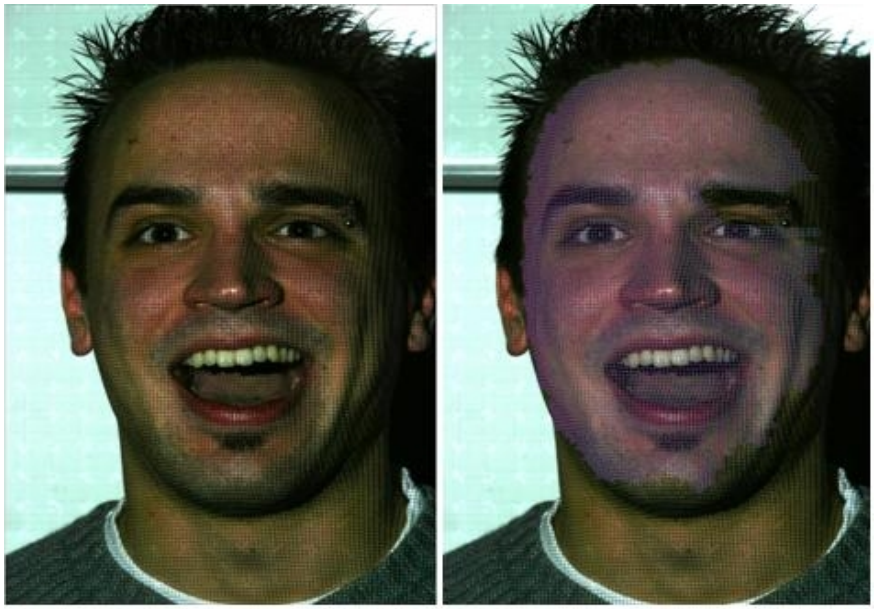

Abbildung A.3: Gitterprojektion und Erkennung

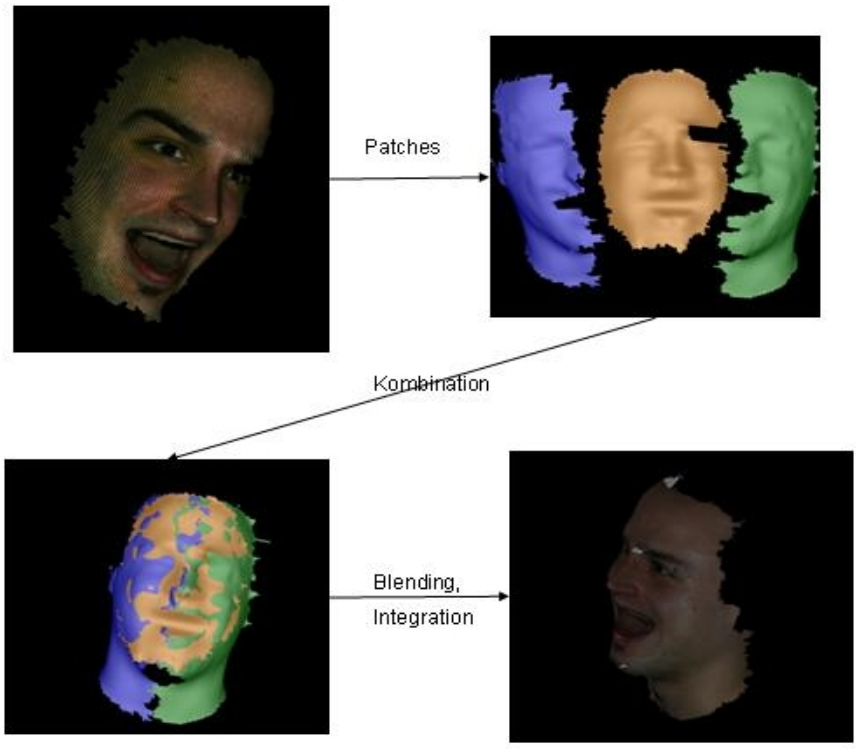

Abbildung A.4: Pipeline zur Erstellung von 3D Scans

Anhang B

Anatomie des Gesichts

Generell sind Muskeln Bewegungsorgane, durch deren Kontraktion bestimmte Regionen des Gesichts bewegt werden. Muskeln liegen zwischen festen oder beweglichen Punkten des Körpers, z.B. zwischen Knochen und Haut, zwischen zwei Knochen oder zwischen zwei Regionen der Haut. In der aktiven Phase eines Muskels findet eine Kontraktion statt, die Relaxation ist passiv. Ein Muskel wird von einem oder mehreren stimulierenden Nervimpulsen gesteuert, die die Kontraktion auslösen. Eine fehlende Stimulierung erzeugt die Relaxation des Muskels.
Die zwei Punkte eines Muskels sind nach [9]

Ursprungspunkte, die eine starre Verbindung zwischen Muskel und Punkt beschreiben sowie

Insertionspunkte, die eine bewegliche Verbindung zwischen Muskel und Punkt beschreiben.

Die Muskeln des Gesichts, die für Gesichtsausdrücke verantwortlich sind, besitzen eine Verbindung zu einer subkutanen Fett- und Hautschicht als Insertionspunkte und liegen oberflächlich in der Haut. Einige Muskeln haben eine Verbindung zur Haut in Ursprungs- und Insertionspunkt, wie der M. Orbicularis Oris.
Viele Muskeln im Gesicht arbeiten synergetisch und nicht unabhängig. Muskelgruppen funktionieren als eine Koordination und sind während der Kontraktion verflochten mit anderen Muskeln, wobei jeder einzelne eine bestimmte Funktion ausführt.
Die Muskeln des Gesichts werden aufgrund der Orientierung ihrer Muskelfasern definiert:

lineare/parallele Muskeln ziehen in eine bestimmte Richtung (z.B. M. Zygomaticus Major),

elliptische/zirkulare Muskeln kontrahieren in Richtung eines Punktes (z.B. M. Orbicularis Oris),

Blattmuskeln verhalten sich wie Reihen linearer Muskeln, die sich über eine Region ausbreiten (z.B. M. Frontalis).

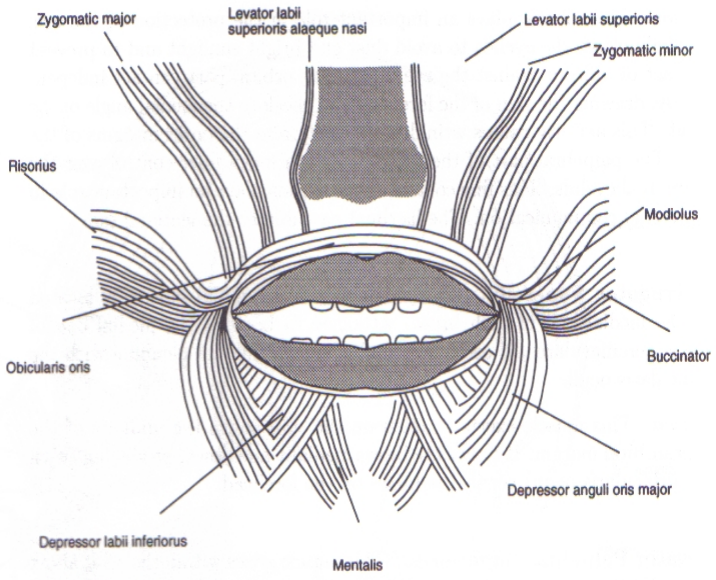

Abbildung B.1: Die Muskeln in der Mundregion aus [9]

Die zahlreichen Muskeln der Mundregion sind wichtige Muskeln für Gesichtsausdrücke:

Musculus Orbicularis Oris umgibt die Mundöffnung mit zahlreichen Muskelsträngen. Die linearen Muskeln M. Buccinator, M. Levator Anguli, M. Levator Labii Superioris, M. Zygomaticus Major und M. Depressor Labii Inferioris verbinden sich mit diesem. Er ist verantwortlich für die Kontrolle der zahlreichen Lippenformen.

Musculus Buccinator drückt die Wangen gegen die Zähne und verbindet sich im Insertionspunkt mit dem Orbicularis Oris.

Musculus Zygomaticus Major hat seinen Insertionspunkt in den Mundwinkeln und hebt diese bei einer gegebenen Kontraktion (z.B. Lächeln).

Musculus Levator Labii Superioris Alaeque Nasi fügt sich in die Haut des Nasenflügels ein. Er hebt die Oberlippe, vertieft die sogenannten Nasolabialfalten, die tangential zum M. Orbicularis Oris liegen, und erweitert die Nasenlöcher.

Musculus Levator Anguli Oris liegt tiefer im Hautgewebe als der M. Zygomaticus Major und verbindet sich mit diesem und dem M. Oribicularis Oris

in den Mundwinkeln. Auch dieser Muskel vertieft die Nasolabialfalten und hebt die Mundwinkel.

Musculus Depressor Anguli Oris und Musculus Depressor Labii Inferioris haben ihren Ursprungspunkt im Unterkiefer und fügen sich in die Mundwinkel ein. Sie ziehen die Mundwinkel nach unten und seitwärts.

Abb. B.2 zeigt die oberflächlichen Muskeln des gesamten Gesichts.

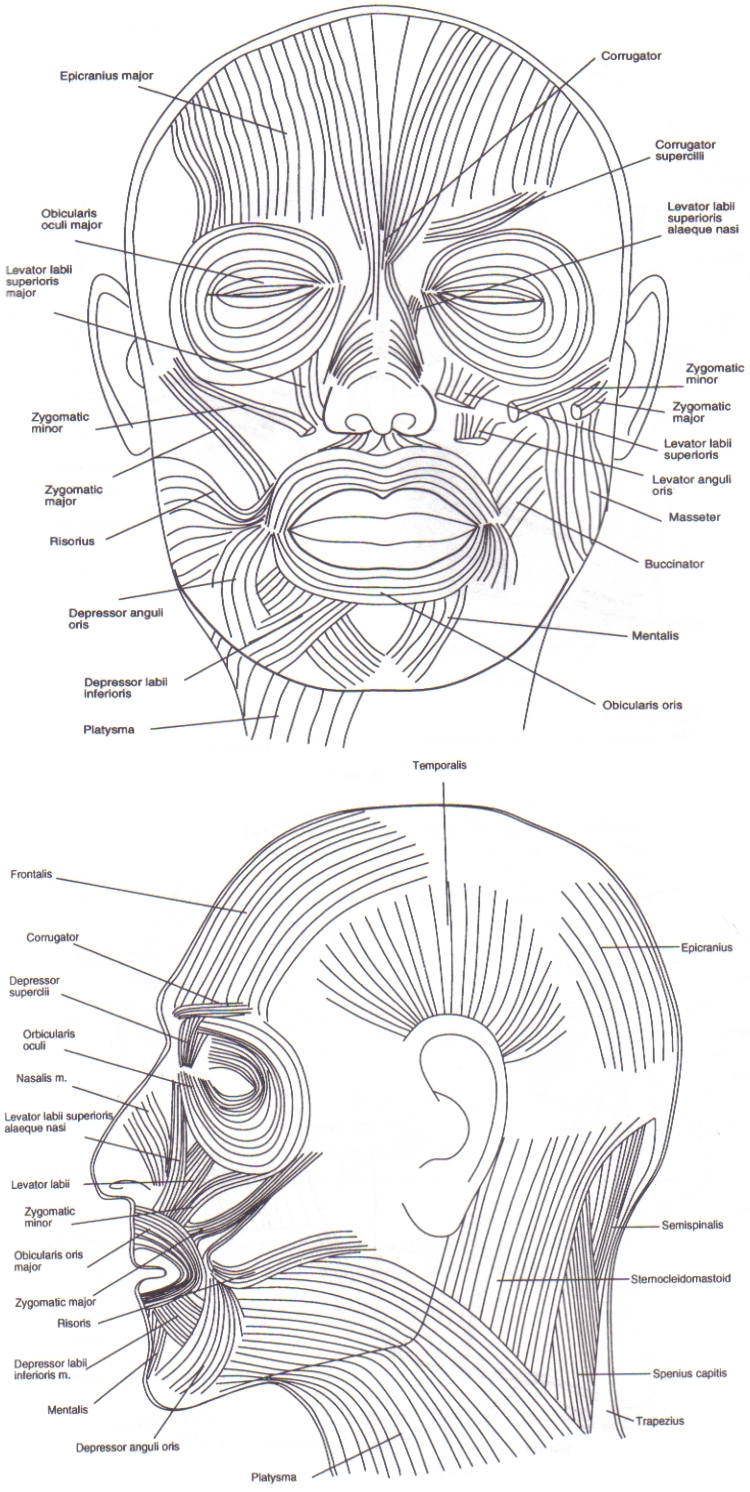

Abbildung B.2: Die oberflächlichen Muskeln des Gesichts [9]

Anhang C

Avango und JOSH

Avango ist ein Software Framework, das für verteilte 3D Anwendungen konzipiert wurde. Seit 1996 wird die Software innerhalb der Abteilung Virtual Environments des Instituts für Medienkommunikation entwickelt. Das Konzept umfasst die Visualisierung dreidimensionaler Daten durch verschiedene Display Konfigurationen, sowie die Möglichkeit der Interaktion durch eine Vielzahl von Eingabegeräten. Das System basiert auf OpenGl Performer und stellt eine Programmierschnittstelle in C++ sowie eine Anbindung an die Skriptsprache Scheme zur Verfügung. Scheme ist eine von Algol und Lisp abgeleitete Sprache. Durch die Skriptsprache ist eine Erstellung und Manipulation aller high-level Avango Objekte zur Laufzeit möglich. Die C++ API erlaubt die Definition zweier Objektklassen:

Nodes bieten eine objektorientierte Szenegraph API, die die Repräsentation und das Rendern komplexer Geometrien ermöglicht,

Sensors werden verwendet, um externe Geräte in eine Anwendung zu importieren.

Alle Avango Objekte werden als sogenannte Field Container bezeichnet, welche den Status von Objekten als eine Sammlung von Feldern repräsentieren. Zur Laufzeit kann durch Scheme auf diese Felder zugegriffen und ihr Inhalt überschrieben werden.
JOSH ist ein Character Animation System, das auf dem Avangosystem basiert. Es bietet viele Funktionen zur Umsetzung und Unterstützung von Charakteren und deren Animation.

Abbildungsverzeichnis

Literaturverzeichnis

[1] E.H. Adelson, P.J. Burt. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*, 1993.

[2] J.M. Ogden, E.H. Adelson, J.R. Bergen, P.J. Burt. Pyramid-based computer graphics, 1985.

[3] J.R. Bergen, R. Hingorani. Hierarchical Motion-Based Frame Rate Conversion. Technical Report, David Sarnoff Research Center Princeton, 1990.

[4] B. K. P. Horn, B.G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 185-203, 1981.

[5] Jean-Yves Bouguet Pyramidal Implementation of the Lucas Kanade Feature Tracker. Intel Corporation.

[6] OpenGl Performer Guide, Version 3.2.

[7] W. H. Leung, B. L. Tseng, Z. Shae, F. Hendricks, T. Chen. Realistic Video Avatar. IBM T.J. Watson Research Center.

[8] K. Shoemake. Quaternions. Department of Computer and Information Science, University of Pennsylvania.

[9] F.I. Parke, K. Waters. Computer Facial Animation. 1996.

[10] I. A. Essa. Analysis, Interpretation and Synthesis of Facial Expression. PhD thesis, MIT, 1995

[11] V. Blanz, T.Vetter. Face Recognition based on Fitting a 3D Morphable Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 9, 2003

[12] V. Blanz, T. Vetter. A Morphable Model for the Synthesis of 3D Faces. MPI Thübingen, 2003

[13] V. Blanz, C. Basso, T. Poggio, T. Vetter. Reanimating Faces in Images and Video. *EUROGRAPHICS*, Vol. 22, 2003

[14] V. Blanz, C. Basso, T. Vetter. Regularized 3D Morphable Models. MPI Saarbrücken, Universität Basel

[15] J. Noh, U. Neumann. A Survey of Facial Modeling and Animation Techniques. University South California

[16] S. Pasquariello, C. Pelachaud. Greta: A Simple Facial Animation System. University of Rome

[17] K. Waters. A Muscle Model for Animating Three-Dimensional Facial Expression. *Computer Graphics (SIGGRAPH '87)*, Vol. 21, 1987

[18] F. I. Parke. Parameterized Models for Facial Animation. NY, Institute of Technology

[19] S. M. Platt, N. Badler. Animating Facial Expression. *Computer Graphics*, Vol. 15, No. 3, 1981

[20] D. Terzopoulos, K. Waters. Physically-based Facial Modelling, Analysis, and Animation. *Journal of Visualization and Computer Animation*, 1990

[21] Y. Lee, D. Terzopoulos, K. Waters. Realistic Modeling for Facial Animation. *Computer Graphics (SIGGRAPH '95)*, 1995

[22] K. Kähler, J. Haber, H. Seidel. Geometry-based Muscle Modeling for Facial Animation. MPI Saarbrücken,

[23] A. Van Gelder. Approximate Simulation of Elastic Membranes by Triangulated Spring Meshes. *Journal of Graphics Tools*, 1998

[24] R. Barzel, A. H. Barr. A Modeling System Based On Dynamic Constraints. *Computer Graphics*, Vol. 22, No. 4, 1988

[25] A. Witkin, K. Fleischer, A. Barr. Energy Constraints On Parameterized Models. *SIGGRAPH '87*

[26] A. Witkin, M. Kass. Spacetime Constraints. *Computer Graphics (SIGGRAPH '88)*, Vol. 22, 1988

[27] M. Gleicher. Motion Editing with Spacetime Constraints. *Interactive 3D Graphics*, 1997

[28] P. J. Besl, N. D. McKay. A Method for Registration of 3-D Shapes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.14, No.2, 1992

[29] D. Chetverikov, D. Stepanov Robust Euclidean alignment of 3D point sets: the Trimmed Iterative Closest Point algorithm

[30] R. M. Haralick, L. G. Shapiro. *Computer and Robot Vision*, vol.2, 1992

[31] J. Noh, U. Neumann. Expression Cloning. *Computer Graphics (SIGGRAPH '01)*, 2001

[32] A. Zell. *Simulation neuronaler Netze.* Addison Wesley

[33] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, W. Stuetzle. Multiresolution Analysis of Arbitrary Meshes, University of Washington, Technical Report, 1995

[34] A. W. F. Lee, D. Dobkin, W. Sweldens, P. Schröder. Multiresolution Mesh Morphing. *SIGGRAPH '99*

[35] D. Zhang, M. Hebert. Harmonic Maps and their Applications in Surface Matching. Carnegie Mellon University

[36] W. T. Vetterling, B. Flannery, S. A. Teukolsky, W.H. Press. *Numerical Recipes in C*, Cambridge University, 1992

[37] B. Choe, H. Lee, H. Koe. Performance-Driven Muscle-Based Facial Animation. *Journal of Visualization and Computer Animation*, 2001

[38] C. Pelachaud, N. Badler, M. Steedman. Linguistic Issues in Facial Animation. *Computer Animation '91*, Springer Verlag, 1991

[39] P. Kalra, N. Magenat-Thalmann, D. Thalmann SMILE: A Multilayered Facial Animation System. *Modeling in Computer Graphics*, Springer Verlag, 1991

[40] D. Terzopoulos, K. Waters Analysis and Synthesis of Facial Image Sequences Using Physical and Anatomical Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 6, 1993

[41] D. W. Massaro *Perceiving Talking Faces*, 1995

[42] B. Guenter, C. Grimm, D. Wood, H. Malvar, F. Pighin. Making Faces. *SIGGRAPH '98*

[43] J. D. Edge, M. Sànchez Lorenzo, S. Maddock. Animating Speech from Motion Fragments.

[44] J. D. Edge, M. Sànchez Lorenzo, S. Maddock, S. A. King. Use and Re-use of Facial Motion Capture Data. *Vision, Video and Graphics*, 2003

[45] K. Mase. Recognition of Facial Expression from Optical Flow. *Special Issue on Computer Vision and its Applications*